éáé

Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Eletrônica e Sistemas
Tel. (081) 271 8210 Fax: (081) 271 8215
Caixa Postal 7800
CEP: 50711-970 Recife-PE

Algoritmo probabilístico paralelo para otimização global

Elaborada por: ***José Ranièri Ribeiro Cavalcante***

Orientador: ***Prof. Fernando Menezes Campello de Souza***

Recife, setembro de 1996

SERVIÇO PÚBLICO FEDERAL

UNIVERSIDADE FEOKRAL DE PERNAMBUCO

CENTRO DE TECNOLOGIA E GEOCIENCIAS
ESCOLA DE ENGENHARIA DE PERNAMBUCO
COORDENAÇÃO DO MES1 O EM ENGENHAI , ELÉTRICA

MESTRADO; ER DA COMISSÃO AMINADORA L ,EFESA DE ENGENHARIA DISSERTAÇÃO DE MESTRADO
ELÉTRICA DE

jj • p p £

JOSÉ RANIERI RIBEIRO CAVALCANTE

TITULO

li **ALGORÍTMO PROBABILÍSTICO PARALELO PARA OTIMIZAÇÃO GLOBAL"**

A Comissão Examinadora composta pelos professores: FERNANDO MENEZES CAMPELLO DE SOUZA, DES/UFPE, MAURO RODRIGUES DOS SANTOS, DES/UFPE e FRANCISCO DE SOUZA RAMOS, CCSA/UFPE, sob a presidência do primeiro, consideram o candidato JOSÉ RANIERI RIBEIRO CAVALCANTE *APROVADO PLENAMENTE.*

Recife, 26 de setembro de 1996.

FERNANDO MENEZES CAMPELLO DE SOUZA

VURO RODRIGUES DOS SANTOS

I

Ficha catalográfica

'-•'' - . . - .'• v. í \ • O i

Cavalcante, José Ranièri Ribeiro

Algoritmo probabilístico para otimização global / José Raniéri Ribeiro Cavalcante . - Recife : O autor, 1996.

516 folhas : il., gráficos.

Dissertação (mestrado) - Universidade Federal de Pernambuco. CTG. Engenharia Elétrica. 1996.

Inclui bibliografia e anexos.

1. Otimização global -programação matemática -pesquisa operacional. 2. Algoritmos probabilísticos. 3. Processamento paralelo (Engenharia Elétrica) I. Título.

519.85 CDU(2.Ed.) UFPE

003 CDD(20.ed.)BC-96-088

"We often classify mathematicians as either pure or applied; hut there are, of course, many other ways of cataloguing them, and the fashions change as the character of mathematics does... These designations are like those commonly used for theoretical and experimental physicists, say; they are independent of whether the objectives are pure or applied, and they do not presuppose that the theoretician sits in bare room before a blank sheet of paper while the experimentalist fiddles with expensive apparatus in a laboratory. Although certain complicated mathematical experiments demand elelronic computers, others call for no more than paper and pencil. The essential difference is that theoreticians deduce conclusions from postulates, whereas experimentalists infer conclusions from observations. It is the difference between deduction and induction. "

Monte Carlo Methods
J. M. Hammersley and D. C. Handscomb

Dedico esse trabalho aos meus pais, meus filhos e a minha esposa Marilene que fazem parte dos meus sonhos e do meu cotidiano.

Resumo

Este trabalho trata da implementação de um algoritmo numérico probabilístico para otimização global. Inicia com uma revisão da área de programação matemática, enfatizando a otimização não linear, em seus aspectos teóricos e práticos. Em seguida, estuda as questões relativas ao processamento paralelo, apresentando algumas taxonomias propostas na literatura e acompanhando a evolução das máquinas de arquitetura paralela (abordagem que vem sendo utilizada para fazer face aos problemas que demandam grande esforço computacional, como é o caso da programação não linear). Na sequencia, aborda o problema da simulação, focalizando as técnicas (o método de Monte Carlo) c aplicações envolvidas, bem como as situações nas quais se deve utilizá-la. Depois, trata da otimização global, procurando enfatizar as abordagens estocásticas e as técnicas que viabilizam a sua utilização. Finalmente, é apresentado o algoritmo proposto, sendo ressaltada a sua natureza inerentemente paralela, bem como focalizada a sua implementação computacional, na qual procura-se enfatizar a simplicidade. Para análise e validação dos resultados obtidos, apresenta-se a implementação do algoritmo em funções de teste com características patológicas de dificuldade. Também é sugerida uma arquitetura paralela inspirada pelo algoritmo.

Abstract

This work deals with the implementation of a probabilistic numeric algorithm for global optimization. It begins with a review of the field of mathematical programming as to its theoretical and practical aspects, emphasizing the nonlinear optimization. Then, it studies issues relative to parallel processing, presenting some of the taxonomy proposed in the literature and following up on the evolution of machines with parallel architectures (which have been used to face problems that demand enormous computational effort, such as is the case with nonlinear programming). Later, the problem of simulation is approached, focusing on the techniques (the Monte Carlo method) and applications involved, as well as on the situations in which it should be used. After that, it deals with global optimization, attempting to emphasize the stocaslic approaches and the techiniqucs that make its use viable. Finally, the proposed algorithm is presented, highlighting its inherently parallel nature, as well as focusing on its computational implementation, where the author strives to emphasize its simplicity. For the analysis and validation of the results obtained, the implementation of the algorithm in test functions with pathologically difficult features is presented. Also, a parallel architecture inspired by the algorithm is suggested.

Agradecimentos

Esse trabalho é fruto de um grande esforço pessoal que foi apoiado por inúmeras pessoas e entidades. Gostaria de agradecer especialmente a EMBRATEL pela minha liberação nesse perióodo de curso e ao pessoal em geral dos seguintes órgãos ESRC, ESRC-MC, DDH01 e da biblioteca da Embratel por toda gentileza e presteza com que me cercaram nesse período. Em particular gostaria de registrar a atenção especial a mim dedicada pelas pessoas que formam a comutação telefônica e o NPD de Maceió e da comutação telefônica do Recife. Meus agradecimentos especiais ao Departamento de Eletrônica e Sistemas da Universidade Federal de Pernambuco pelo apoio e convivência ao longo desses 30 meses, aos meus colegas de mestrado por todos os momentos de esforço conjunto, dedicação, discussão e aprendizado.

Meu mais profundo reconhecimento pela grandeza de alma e generosidade de Ismênia Ribeiro e Maria José de Santana.

Não podia deixar de agradecer aos meus amigos André Marques Cavalcante e André Simon. O primeiro por me fazer crer o tempo todo que "já estava descendo a ladeira" c ao segundo por toda ajuda que me proporcionou, aos dois serei eternamente grato.

Ao meu orientador e amigo professor Fernando Menezes Campello de Souza por sua paciência, disponibilidade c profundo conhecimento do que faz, fatores que transformam um professor em mestre.

Finalmente agradeço a minha família pelo apoio, compreenção c, principalmente, por estarem ao meu lado em todos os momentos e dificuldades.

S u m á r i o

3.5.2 Produtor/consumidor

O problema do produtor/consumidor também envolve um compartilhamento de dados mas nesse caso o objeto compartilhado é especificado como um bufier. O processo produtor cria objetos que são colocados em um buffer, o consumidor espera até o objeto ter sido colocado no buffer, remove e consome-o. Isto pode ser modelado como mostrado na figura 3.5.2.1 (a) abaixo. O lugar B representa o buífer, cada ***token*** representa um item que foi produzido mas não foi ainda consumido.

Uma variante deste problema é o multiplo-produtor/multiplo-consumidor. Nesta variante múltiplos produtores produzem itens que são colocados em um ***buffer*** comum para os múltiplos consumidores, a figura 3.5.2.1 (b) abaixo mostra este problema, modelado por rede de Petri, que tem a mesma solução do problema anterior exceto pelo fato de representar p produtores e i consumidores.

Outra variante é o problema produtor/consumidor com ***buffer*** limitado. Neste caso reconhece-se que o buffer entre o produtor e o consumidor é limitado, isto é, tem n posições, sendo n finito e cnumerável. Deste modo o produtor não pode sempre produzir tão rápido quanto deseje, terá que esperar se o consumidor for menos rápido e o *buffer* estiver cheio. A figura 3.5.2.1 (c) abaixo ilustra este modelo cm termos de rede de Petri. Este limite no tamanho do bufTer é representado por dois lugares: B representa o número de itens que foram produzidos mas ainda não foram consumidos (o número de posições cheias); B' representa o número de posições vazias no *buffer*. Originalmente B' tem n *lokens* e B tem zero. Se o *buffer* está cheio, então B' terá zero *lokens* e B terá n. Neste ponto se o produtor tenta colocar outro item no ***buffer*** ele não poderá faze-lo por causa da inexistência de *loken* em B' para habilitar aquela transição.

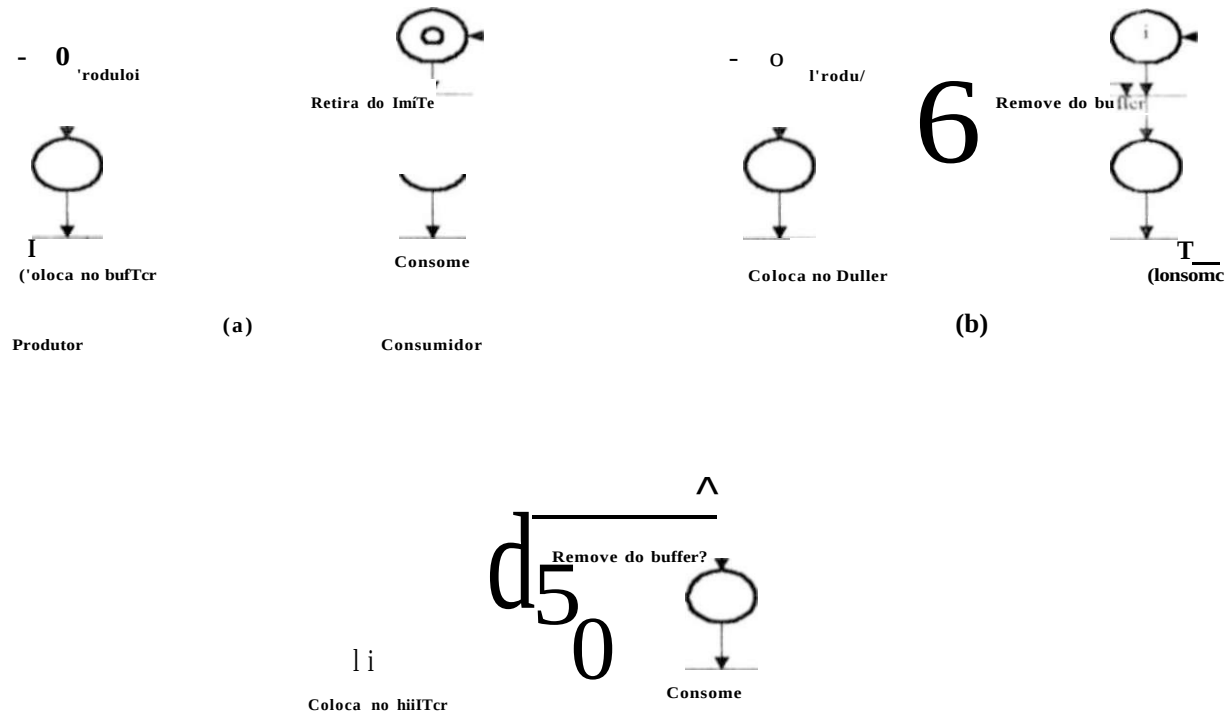


Figura 3.5.2.1 - (a) produtor consumidor, (b) múltiplo consumidor c (c) produtor consumidor com buffer

3.5.3 O jantar dos filósofos

Este problema é relativo a cinco filósofos que alternadamente pensam e comem. Os filósofos estão sentados cm torno de uma mesa sobre a qual estão vários tipos de comidas chinesas para serem consumidas por eles. Entre cada par de filósofos se encontra um pauzinho, porém para comer comida chinesa são necessários dois pauzinhos, então o filósofo para comer deve se apoderar de dois pauzinhos, o da esquerda e o da direita. 0 problema é que se cada filósofo se apoderar do pauzinho à sua direita ele terá que esperar infinitamente para comer e então morrerá de fome (condição de dcadlock). A figura abaixo modela este problema com rede de Petri.

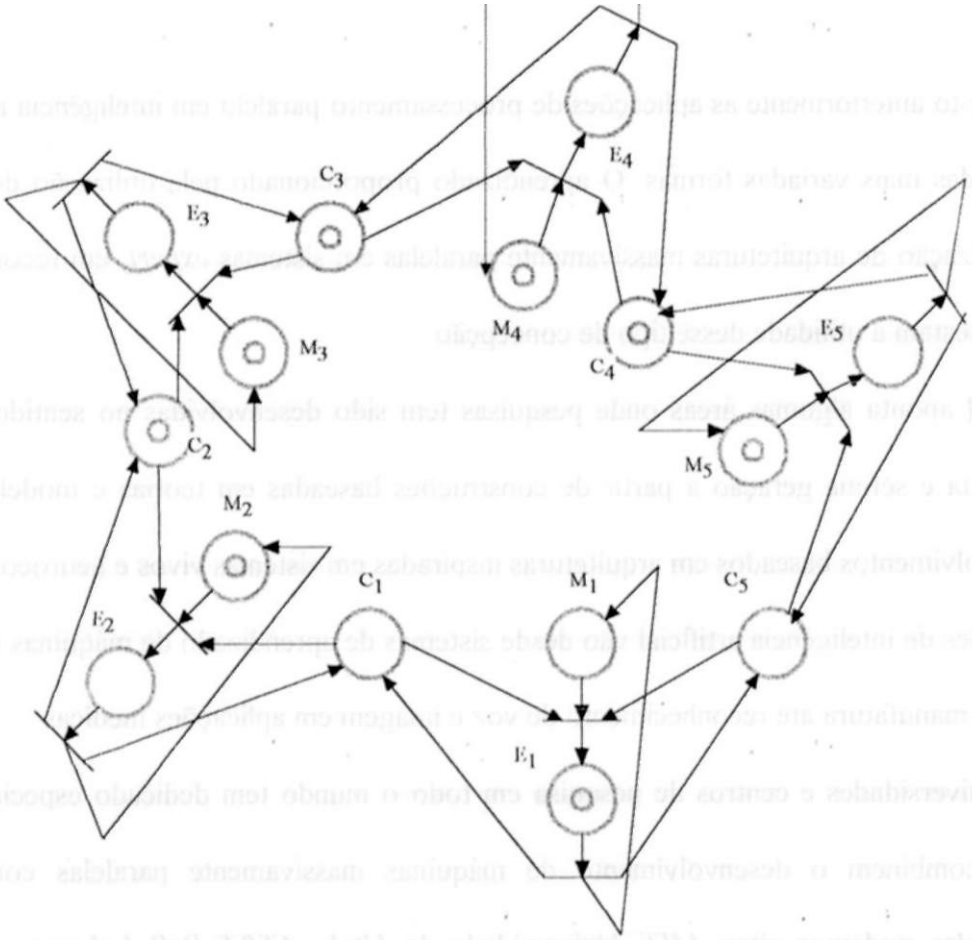


Figura 3.5.3.1 - O jantar dos filósofos, (M_i) meditando e (C_i) comendo.

Os lugares Cj, i =1,...,5, representam os pauzinhos, e desde que cada um deles no início está livre, na marcação inicial cada um deles tem um *token.* Cada filósofo está representado por dois lugares Mj e Ej representando os estados meditando e comendo, respectivamente. Para um filósofo sair do estado de meditação para comer, os dois pauzinhos (da esquerda e da direita) devem estar disponíveis.

3.6 Processamento paralelo c inteligência artificial

Como já visto anteriormente as aplicações de processamento paralelo em inteligência artificial são as mais diversas e das mais variadas formas. O aprendizado proporcionado pela utilização de máquinas de Boltzman, a utilização de arquiteturas massivamente paralelas em sistemas *experl,* em reconhecimento de voz e imagem, atestam a utilidade desse tipo de concepção.

|Soucek| aponta algumas áreas onde pesquisas tem sido desenvolvidas no sentido de obtermos - máquinas de sexta e sétima geração a partir de construções baseadas em teorias e modelos do cérebro humano, desenvolvimentos baseados em arquiteturas inspiradas em sistemas vivos c.neurocomputadores.

As aplicações de inteligência artificial vão desde sistemas de aprendizado de máquinas para utilização em processos de manufatura até reconhecimento de voz e imagem em aplicações médicas.

Várias universidades e centros de pesquisa em todo o mundo tem dedicado especial atenção nas pesquisas que combinem o desenvolvimento de máquinas massivamente paralelas com inteligência artificial, entre elas podemos citar: *MIT,* Universidade de *Utah, AT&T Bell Laboratories,* Califórnia *Institute of Technology,* Universidade do Arizona, Universidade *Camegie-Mellon,* Universidade John Hopkins, Universidade do Sul da California, *National Bureau oj Standard,* Universidade do listado de Ohio entre outros.

3.7 Uma breve discussão sobre desempenho das máquinas paralelas

Jamais o poder de computação das máquinas existentes satisfez a voracidade de usuários de aplicações científicas, de engenharia ou negócios. O problema é que estes usuários têm encontrado um supercomputador ao preço de US$ **20** milhões; no entanto já estão aparecendo novos superminicomputadores com especificações comparáveis aos supercomputadores ao preço de US$

500.000. Arquiteturas tradicionais de computadores estão também atingindo seus limites. As novas máquinas de processamento paralelo trazem a esperança, não somente de resolver problemas científicos c de engenharia rapidamente, mas permitindo a solução de problemas anteriormente intratáveis. Quanto aos problemas em negócios estas novas máquinas irão resolver tais problemas, por exemplo, processamento de transações ou manipulação de bancos de dados, estes problemas podem não parecer importantes, mas eles poderão ser executados em menos tempo, levando a um menor custo efetivo.com máquinas.

Como já dito anteriormente os limites impostos pelas leis da física no desenvolvimento da tecnologia de semicondutores é um forte indicador que a demanda por velocidade de computação da próxima década (século XXI) não poderá ser atendida por máquinas com arquitetura baseadas no modelo de von Neumann, essa demanda só será atendida por máquinas que apresentem uma arquitetura inovadora e que explorem o paralelismo ver [Menassé], [Hindin] e [Lackey et. al.].

Basicamente dois paradigmas tem sido propostos para máquinas paralelas segundo [Menassé]: Multiprocessadores com memória compartilhada c processadores com memória distribuída. O fato dessas máquinas disponibilizarem processadores em paralelo não significa que a performance irá melhorar por um passe de mágica; para se conseguir uma melhora efetiva e substancial de performance é necessário decompor adequadamente a aplicação em tarefas paralelas de tal forma a conseguir extrair dessa decomposição c dos processadores em paralelo a sinergia para atingir a meta desejada.

Algumas questões surgem e têm que ser respondidas quando nos propomos a projetar máquinas paralelas:

i) Qual é a decomposição ótima da aplicação em tarefas paralelas?

ii) Como otimizar a alocação de tarefas entre os processadores?

iii) Qual é a performance esperada da aplicação em uma dada arquitetura?

Vamos analisar algumas propostas de metodologia para avaliação de desempenho de máquinas paralelas.

Uma variedade de tecnologias e arquiteturas está por trás do esforço em pesquisa e produtos. A comunicação entre múltiplos processadores é possível através da troca de mensagens, compartilhamento de memória comum ou alguma mistura dessas duas técnicas. O mesmo sistema operacional pode estar em todos os processadores, ou diferentes sistemas operacionais podem estar presentes (uma máquina IBM com *VM-Virtual machine-* c sobre ele rodando VSB, OS, etc.) ou uma parte de um sistema operacional (por exemplo, o núcleo do sistema) pode estar em um processador.

Uma tarefa específica em um programa de aplicação pode também ser dividido em pequenas partes por um compilador para execução por múltiplos processadores, ou cada processador pode manusear somente tarefas diferentes e específicas.

Para as aplicações a meta de todas as máquinas de processamento paralelo, independente do tipo, é o aumento de velocidade. De que maneira é realizada na prática, contudo, depende de alguns fatores: a natureza da aplicação, a escolha do *hardware/arquilcíurd,* o overhead de *software* que resulta da necessidade de comunicação e sincronização entre os processadores, em que medida as tarefas paralelas e a habilidade do algoritmo paralelo e código se o *software* é originalmente orientado para arquitetura paralela.

Uma visão inicial do processamento paralelo, calcula que um sistema usando N processadores, rodando uma aplicação particionada em N tarefas similares, iria gozar de uma relação linear entre sua performance e o número de processadores - um sistema com N processadores iria ser N vezes mais rápido que um único processador.

Outro ponto de vista é que a relação mais pessimista **log2** N acontece com N grande.

Uma visão mais otimista, admite que se um computador tem duas velocidades de operação (uma para código paralelo e outra para código seqüencial), a menor velocidade domina a performance independente da velocidade do código rápido (lei de ***Amdaht).*** Isto conduz a $N/\log_{()}N$ como relação entre velocidade e número de processadores.

Tem-se tentado melhorar a relação N/logioN desenvolvendo técnicas para fazer o FORTRAN existente paralelo e realizar uma relação de velocidade do processador de 0.3N ou melhor. Alega-se que o controle de concorrência baseado em *hardware* dedicado, combinado com um compilador para geração de código Fortran paralelo a partir dos programas existentes, permite uma abordagem teórica que introduz melhoramentos (N processadores sendo N vezes mais rápido) para algumas aplicações.

De acordo com 11 lindinj usando uma arquitetura VLSI concorrente similar aquela do cubo cósmico colocada no mercado pela divisão de sistemas da INTEL tem realizado um aumento na relação de velocidade do processador acima de 0.94N para sistemas com 64 e 128 processadores.

A lei de *ÂMDAHL* pode ser aplicada em situações comuns onde algum código é executado por processadores paralelos e outros são executados somente por processadores seqüenciais simples. O ganho de velocidade cm um sistema com N processadores é então algum valor entre a alta velocidade (código paralelo) c baixa velocidade (código serial) e depende de quanto do código é paralelo e quanto é serial. O incremento de velocidade para N processadores comparado com um processador único é t/(F t (l-l^/N), onde F é a percentagem de código serial. O ganho possível com processamento paralelo estará limitado se uma pequena quantidade de código serial (F igual a 0.1, por exemplo) está presente por causa deste código não paralelo a eficiência é bastante diminuída em uma máquina paralela.

É claro que melhorias significativas de velocidade requerem software e algoritmo altamente paralelos. Isso não é totalmente claro, contudo, quando não sabemos quantos algoritmos existentes têm paralelismo inerente ou quais novos algoritmos e implementações em software serão necessários. Some-se a isto o fato de que os estimadores da relação de velocidade para o número de processadores podem ser muito otimistas. Estes estimadores são usualmente baseados no fato de assumirmos que as instruções executadas em processadores paralelos são manuseadas da mesma forma daquelas executadas em processadores Seqüenciais. Sistemas multiprocessadores requerem instruções adicionais para processar sincronização da comunicação entre processadores e algoritmos paralelos podem requerer instruções adicionais (overhead).

Cálculos de eficiência lambem assumem que Iodos os processadores paralelos estão trabalhando o tempo inteiro. Isso por sua vez depende da capacidade do compilador para dividir as tarefas a serem realizadas Infelizmente as relações que compõem o ***tradeoff*** entre o número de processadores que devem ser considerados - granulai idade, projeto de compilador, tipo de arquitetura paralela, percentagem de código serial, linguagem de programação - não são ainda de domínio completo para grande parte dos pesquisadores.

3.8 Conclusões

Qualquer que seja o enfoque que queiramos dar ao aumento de demanda por computadores mais e mais velozes um fato é inegável: as aplicações científicas (poderia ser aplicação em prospecção de petróleo, sismologia, meteorologia, etc), por exemplo, função da tecnologia, nível de exigência dos usuários, complexidade crescente dos problemas a resolver, etc, são a cada dia mais sofisticadas e demandam crescentes tempos de CPU para sua realização, [Menassé] define aplicação cientifica como aquelas que gastam muito tempo de execução de CPU e que fazem uso intensivo de operações de ponto flutuante e instruções vetoriais, as máquinas convencionais já de algum tempo não suprem esta demanda, no futuro só as máquinas de arquitetura paralelas poderão se candidatar a prestação deste tipo de serviço Portanto, o desenvolvimento de arquiteturas, organizações, algoritmos e compiladores para estas máquinas são de fundamental importância para o domínio tecnológico de empresas, universidades, governos, etc Em toda essa discussão um aspecto que chama atenção é a pesquisa e desenvolvimento de algoritmos e soflwares que possam tirar o máximo proveito do paralelismo do *hardware,* nesse sentido c que procuramos apresentai esse nosso trabalho em torno do desenvolvimento de um algoritmo para resolver uma classe de problemas que tem importância fundamental em diversas áreas, que é o problema de otimização, ("orno visto ao longo deste capítulo, muitas arquiteturas e variantes delas (nível de *hardware* e

sistema operacional) estão disponíveis, uma vertente forte de pesquisa concentra-se no desenvolvimento de algoritmos que otimizem a utilização dessas máquinas.

Saber quais aplicações continuarão sendo rodadas cm computadores clássicos (seqüenciais) e quais migrarão para máquinas paralelas (multiprocessador, com memória associativa, massivamente paralelo, neurais, genética, etc.) é, do ponto dc vista especulativo, uma questão que assemelha-sc a mesma discussão relativa ao cérebro humano.

0 cérebro humano é dividido em dois hemisférios: esquerdo e direito. A experiência mostra, ver [Soucek], que o hemisfério esquerdo é especializado cm tareias seqüenciais c o direito cm tarefas paralelas. Tarefas seqüenciais utilizam lógica simples e pequena proporção dos dados disponíveis cm qualquer tempo dado. Se um problema pode ser sucessivamente decomposto em partes mais simples e relativamente independentes, ele pode ser resolvido por uma máquina seqüencial similar aos computadores clássicos ou o hemisfério **esquerdo** (tarefas seqüenciais incluem raciocínio lógico, operações matemáticas **seqüenciais,** planejamento, entendimento e produção de linguagem).

Tarefas paralelas usam todos os dados disponíveis ao mesmo tempo. A lógica de tais tarefas não pode ser decomposta em partes independentes mas ao contrário requer uma síntese global Tarefas paralelas demandam um grande número de processadores operando interligados em paralelo, tal como o hemisfério direito do cérebro (tarefas paralelas incluem processamento dc imagens, operações matemáticas **paralelas,** coordenação do corpo ou de um robô, reconhecimento de padrões, raciocínio e aprendizado analógicos).

Uma conclusão importante é que existe lugar em pesquisas e aplicações para ambas as concepções: computadores clássicos seqüenciais e computadores paralelos. Cada grupo é especializado em tarefas distintas similarmente a especialização dos hemisférios direito e **esquerdo** do cérebro humano.

Capítulo 4

Simulação

*"Througtoul history men have employed elaborate rituals to help them reach a decision. They have poured libations, sacrificed animals, read the stars, and watched the flight of birds. They have put their faith in proverbs and rules of thumb devised to take some of the guesswork out of living. Today *s management of decision making employs a new and perhaps more scientific ritual, the use of the computer. "*

David M. Uimmelblau

CAPITULO 4

Simulação

4.1. Introdução

A simulação é uma forma particular dc experiência. Numa experiência comum, normalmente, o ensaísta exerce uma ação sobre o objeto em estudo. Numa simulação este contato não existe, no sentido que o ensaísta não age diretamente sobre o objeto original mas sobre seu modelo, ver [OurmacvJ.

A simulação c a representação do comportamento de um processo físico, industrial, econômico, social, etc. Por meio dc um modelo material, matemático ou lógico cujos parâmetros e variáveis são aqueles do processo cm estudo. Por exemplo, um simulador de vôo de um avião a jato, é um dispositivo composto dc hardware (cópia da cabina de comando e seus instrumentos), e software (os programas que ao receber os comandos do piloto em treinamento acionam todo artefato de hardware envolvido no intuito de gerar no treinando a sensação exata de estar na cabina de comando dc um jato no ar).

Como já dito anteriormente quando falamos cm simulação está subjacente a existência dc um sistema e dc um modelo que o represente. Um modelo pode ser construído para representar todo o sistema ou parte deste. O modelo irá especificar a velocidade ou tamanho dos vários elementos constituintes, a interação lógica entre suas partes constituintes, etc. Qualquer que seja o contexto em que esteja inserido o modelo (industrial, administrativo, econômico, social, dc engenharia, etc), sua construção não obedece nenhuma regra universalmente aceita, ao contrário, esta é a fase da simulação que exige do projetista uma grande dose de bom senso, lógica, arte e, o que é principal, muita criatividade.

Desde os tempos mais remotos o homem procurou construir modelos com os quais pudesse entender o fenômeno real sob estudo. De início os sistemas eram simples e os modelos, normalmente, de fácil construção. A medida que os sistemas de interesse tornaram-se mais e mais complexos os métodos e técnicas de simulação procuraram acompanhar essa tendência.

Como o conceito de modelagem é bastante antigo, o que aconteceu para aumentar tanto o interesse pelos métodos dc simulação de sistemas em anos recentes? a resposta é: a invenção do computador digital (ele permite a implementação- Cálculo - de representações matemáticas - modelos -dc sistemas, determinísticos ou não). Com o advento do computador digital, a simulação tem sido aplicada em muitos problemas nos mais diversos campos da atividade humana. O computador digital é a ferramenta que permite a simulação de qualquer meio ambiente (em sentido amplo) existente ou não Uma das mais poderosas ferramentas que podem ser usadas na estimação dos requisitos de um sistema é a simulação. Problemas em campos tão diversos como negócios, política, engenharia, economia, comportamento social, etc tem sido resolvidos com o uso de simulação.

Quando falamos em simulação as primeiras perguntas que surgem são. simular o que''com que objetivo? qual a necessidade de usar simulação?

Neste capítulo tentaremos, evidentemente sem a pretensão de esgotar o tema, responder a essas perguntas, e, mais ainda, tentaremos justificar o interesse crescente pelo tema simulação, procuraremos mostrar uma nova ferramenta para a solução de problemas e montagem dc cenários: técnicas de simulação determinísticas c probabilísticas, em particular os métodos de Monte Carlo, que cm virtude da ênfase que procuramos imprimir a eles nesse trabalho, serão apresentados em um capítulo a parte.

Em ciências da natureza, fisica experimental, por exemplo, procura-se medir valores de parâmetros a partir de ensaios realizados através de experiências para aferição de estimadores para esses parâmetros. Em engenharia, economia, administração de estoques, montagem de cenários de comportamento social, etc a realização de experimentos práticos torna-se inviável, principalmente, em (unção do custo envolvido (tanto no sentido financeiro quanto no tempo despendido para tal), o que sc faz, normalmente, é montar um modelo matemático que represente o fenômeno cm análise e variar seus parâmetros de forma a gerar cenários ou reações distintos para análise e tomada de decisões.

Imagine, por exemplo, que uma determinada empresa está montando seu planejamento dc longo prazo e quer saber se deve ampliar sua produção, se essa ampliação trará retorno (lucro) maior se feita com aumento da produção das plantas existentes ou se nova(s) planta(s) devc(m) ser construídas. Para responder a essas perguntas e a muitas outras relativas a esse problema, tais como, a influência da taxa de crescimento da demanda ano a ano, regiões onde a demanda deve crescer ou decrescer, cusjo de ampliação das plantas existentes e custo de construção de novas unidades, inflação dc custos, entre outras lançamos mão da simulação. Um modelo, geralmente matemático, e um programa, é escrito em um computador para agir como um modelo do sistema cm questão, líle se comporta em alguns aspectos como o sistema mas muito mais simplificado.

\i fácil ver que, qualquer que seja o modelo matemático adotado a incerteza c inerente ao problema, pois, ao longo do tempo, não há como saber, com certeza absoluta, por exemplo, a demanda no décimo ano, nem o aumento de custos até o quinto ano devido a inllação de custos, nem se na região A demanda diminuirá no segundo ano c na região B aumentará.

Existem ferramentas matemáticas que podem ser utilizadas para suprir essas necessidades. Um modelo de programação linear ou de programação dinâmica, são capazes dc montar cenários para apoio a decisão, no entanto, estes modelos apresentam fragilidades intrínsecas, por exemplo, o modelo de programação linear pode conseguir apreender os elementos dinâmicos da situação porém tratar os aspectos de incerteza usando valores médios, o que gera previsões pobres. Um modelo de programação dinâmica tem a capacidade de tratar problemas de planejamento de múltiplos períodos com incertezas, só que esses modelos são extremamente simplificados e aproximados de maneira a **torná-los** computacionalmente aceitáveis. Uma ferramenta bastante versátil é a simulação, para mais detalhes ver [Wagner], |Naylor| e |Naylor et al.|.

Simulação é uma técnica, que por ser relativamente recente, não se encontra estruturada de tal sorte que possua uma seqüência predeterminada de passos padronizados válidos para qualquer típo de problema.

Em simulação cada modelo demanda determinados cuidados e dispende algumas tarefas específicas. Entretanto não deve-se imaginar que a construção de modelos para simulação dc sistemas seja feita de forma anárquica, pelo contrário, existem algumas linhas mestras e técnicas a serem seguidas e utilizadas para que um modelo eficiente possa ser construído

A simulação é uma valorosa técnica de monitoramento e assistência para a implementação de novos sistemas ou para a modificação do ***modus opetandi*** dc um sistema existente.

Em [Graybeal] são apresentadas algumas vantagens e desvantagens propostas por Adkins e Pooch.

Vantagens:

a) Ele permite o controle do experimento, isso permite que o sistema seja rodado um sem número de ve/es variando os parâmetros de entrada para se conhecer o comportamento do sistema.

b) Permite compressão do tempo

c) Permite análise de sensibilidade pela manipulação das variáveis de entrada

d) Ele não atrapalha o funcionamento do sistema real

e) É uma ferramenta de treinamento.

Desvantagens:

a) A simulação pode ser muito cara em termos de mão de obra e tempo de computação dependendo da magnitude do problema e do modelo proposto.

b) Dependendo da dimensão do problema e do modelo proposto pode levar muito tempo de desenvolvimento.

c) Suposições implícitas no modelo podem levar o modelo a divergir da realidade.

d) Os parâmetros do modelo podem ser dc difícil inicialização.

4.1.1 Definição dc simulação

Shannon, o pai da moderna **teoria** da comunicação, definiu simulação como o processo de **realização** dc experimentos sobre o modelo computacional de um sistema, com o propósito de, ou entender seu comportamento ou de avaliar as várias estratégias para a sua operação. Na literatura especializada encontramos muitas definições diferentes para simulação. A simulação torna possível o estudo sistemático de problemas quando soluções analíticas não são possíveis ou experimentos sobre o sistema são impossíveis ou impraticáveis. Hm |Graybeal| considera-se como fundamental para o estudo da simulação a idéia e o conceito de sistema. Lm |Lewis| é admitido explicitamente que a simulação em computador está intimamente ligada ao conceito de sistema. |Kobayashi| define simulação como uma técnica essencialmente de condução de experimentos sobre amostras de um modelo. [Naylor et. al.| enunciam uma definição proposta por Shubik: A simulação de um sistema ou de um organismo é a operação de um modelo (ou simulador) que representa esse sistema ou organismo. O modelo é passível dc manipulações que seriam difíceis de levar a cabo na entidade que ele representa, quer pelo preço, quer pela impraticabilidade ou impossibilidade de **fazê-las.** As propriedades concernentes ao comportamento de um sistema ou subsistema podem ser inferidas estudando-se a operação do modelo. [Naylor et. al.| formulam sua própria definição como uma técnica numérica para realizar experiências em um computador digital, as quais envolvem certos tipos de modelos lógicos que descrevem o comportamento de um sistema econômico ou de negócios (ou um aspecto parcial de um deles) sobre extensos intervalos de tempo.

4.2. Conceito c taxonomia de sistema

[Lewis] define abstratamente sistema como interações entre componentes, possíveis situações chamadas estados, c regras governando a seleção de estados. O dicionário Aurélio define sistema como o conjunto dc elementos, materiais ou ideais, entre os quais se possa encontrar ou definir alguma relação. Novamente em **[Graybeal]** somos alertados para o fato de que no contexto da simulação o termo sistema

se refere a unia coleção cie objetos com um conjunto bem definido de intei relações entre eles, ou simplesmente como um conjunto de objetos e interações. [Iiertalanfly| define sistema como um complexo de elementos em interação. Essa interação significando que os elementos p estão em relações R, de modo que o comportamento de um elemento p em R é diferente dc seu comportamento em outra relação IC; em [Graybeal] discute-se ainda o ambiente do sistema e classifica as atividades. Quatro possíveis formas de classificação de sistemas são:

a) Quanto ao comportamento de mudanças de estado em relação ao tempo:

- Discretos
- Contínuos

b) Quanto à forma como prever o estado futuro do sistema a partir do estado atual e da realização de uma atividade:

- Determinísticos
- Probabilísticos

c) Quanto ã influência das atividades externas:

- Abertos
- fechados

d) Quanto a influência do fator tempo:

- Estático
- Dinâmico

A simulação e baseada no método cientifico, isto é, o método de construção de modelos que tipicamente desenvolve-sc cm quatro estágios a saber:

- Observação do sistema (que necessariamente admite uma previsão teórica);
- formulação de hipóteses ou teorias que expliquem o comportamento observado;

- Predição do comportamento futuro do sistema baseado nas hipóteses ou teorias c assumindo que estas estão corretas;

- Comparação do comportamento predito com o comportamento real.

O sistema, esse conjunto de objetos e interações já definido anteriormente, quando de seu mapeamento, deve ser construído com um conjunto de limites de tal sorte que se possa saber onde ele começa e onde termina: desse modo tudo que se encontra fora desses limites forma o ambiente do sistema. Então esse conjunto de fatores externos ao sistema, que podem ou não influir nele, forma seu ambiente. Em ICiraybeal | define-se o estado do sistema como o conjunto minimal de informações capa/, de predizer o estado futuro do sistema, de forma única na ausência de incerteza. A introdução do tempo implica em mudança no estado do sistema, devendo, então, existir uma disciplina para isso. lista disciplina é feita a partir da ocorrência de um processo ou evento. Um processo ou evento é definido como uma atividade. As atividades podem ser dc duas classes: endógenas e exógenas. Uma atividade endógena é uma atividade interna ao sistema, e, em oposição, uma atividade c exógena quando é externa ao sistema Aqui cabe uma advertência; nem sempre é possível definir se uma atividade é interna ou externa ao sistema. Para uma discussão mais profunda do conceito e laxonomia de sistemas ver [BertalanHy| e |Churchman|.

A metodologia da engenharia de sistemas é constituída basicamente de quatro fases:

- Planejamento
- Modelagem
- Validação
- Aplicação

Na fase inicial de planejamento ou pré-modelagem está incluída a visão inicial do sistema, o problema a ser solucionado, os fatores pertinentes e o ambiente.

Na fase dc modelagem o analista constrói um modelo do sistema.

O modelo e validado provando-sc que ele é uma **representação** correta do sistema, mostrando-se que ele tem um grau de aderência satisfatório à realidade.

Na última fase verifica-se se não aparecem erros dc implementação, já que, como dito anteriormente, 0 modelo é uma representação do sistema real e daí ajustes serão leitos ou no caso extremo o modelo será totalmente refeito.

[Kobayashi | aponta cinco fases envolvidas na simulação:

- Formulação de um modelo
- Implementação
- Projeto dos experimentos
- Validação
- Hxecução da simulação e análise dos dados

Hm |Wagner| são apontadas três categorias de problemas de decisão gerencial em que o uso de simulação é o mais indicado:

a) Hscolha dc políticas de investimentos para planejamento estratégico;

b) Seleção de instalações no planejamento de operações;

c) Concepção de regras de programação com realimentaçào de informações

Hm **ISchocmakcr] outras três** categorias **são apontadas:**

a) Quando um risco do tipo físico, financeiro, comercial, está em jogo, tais como, o efeito destrutivo de um novo sistema de armamento, os resultados de decisões corporativas, a relação custo-beneficio de um sistema de telecomunicações, ele

b) Quando a escolha entre alternativas é difícil, tal como, problemas na avaliação de decisões sobre projetos de engenharia, etc.

c) Quando a láctibilidade de um projete) é o problema, tal como, escolha de caminho crítico **a** ser utilizado em um projeto, a tolerância de um governo ao aumento da poluição do ar, da água, etc

[Naylor et. al.| aponta os elementos que devem caracterizar um modelo que se presta a simulação, são eles:

1. Muitas variáveis c suas funções;
2. Variáveis aleatórias e suas distribuições;
3. Muitos parâmetros;
4. Muitos interrelaeionamentos entre os elementos do modelo;
5. Não linearidades;
6. Diversas restrições;
7. Uma ou várias respostas que podem ou não ter dependência do tempo.

Essa técnica sofreu um grande impulso de utilização a partir do desenvolvimento dos computadores digitais. Isso significa que a simulação de sistemas de grande porte sem o auxilio dos computadores, digitais hoje existentes seria completamente inviabilizada.

O surgimento da microeletrônica iniciou uma nova discussão sobre arquiteturas de **computadores,** nas novas arquiteturas não somente o aspecto lógico é importante mas também a organização do computador, c um aspecto muito importante do desenvolvimento das arquiteturas é a busca do paralelismo, a esse lema já dedicamos o capítulo 3 desse trabalho.

4.3. Simulação de sistemas discretos

Se um modelo é estático e determinístico, muito provavelmente consegue-se uma solução analítica c essa solução muito provavelmente é melhor que uma simulada. No entanto, quando o sistema é dinâmico, essa solução, só é analiticamente possível com muitas suposições que simplifiquem o modelo de tal forma a tornar a solução analítica computacionalmcnte factível, então, na grande maioria dos casos, uma solução

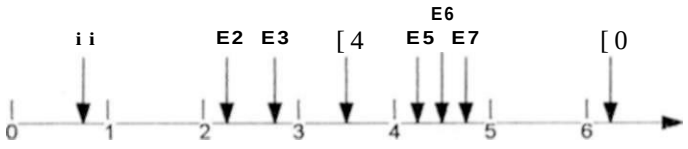
para um sistema dinâmico só é conseguida a partir da simulação de um modelo que o represente de forma adequada.

O estado de um sistema é expresso como uma função do tempo. Dois parâmetros de tempo estão envolvidos quando se utiliza simulação: O tempo dc simulação (*simulation lime)* e o tempo gasto na simulação (*mti lime).* O primeiro representa o horizonte temporal de desenvolvimento da simulação. Quando estamos simulando um modelo para análise da influência de resíduos radioativos no ambiente c na saúde das pessoas, esse horizonte é de centenas ou milhares de anos, ou em um modelo de planejamento de longo prazo para o tráfego telefônico, por exemplo, este horizonte é de 5 anos, enquanto quando estamos simulando o funcionamento dc uma CPU este horizonte passa a ser da ordem de milisegundos. Já o segundo parâmetro dc tempo se refere ao tempo de computador despendido para simular **o** sistema, e esse parâmetro, influencia diretamente no custo da simulação. Visto dessa forma já podemos saber que na grande maioria dos casos o tempo gasto na simulação é menor e, às vezes, muito menor que **o** tempo de simulação. A partir daqui então, verifica-se que o fator tempo em simulação é crucial e dai veremos os métodos de gerência do tempo.

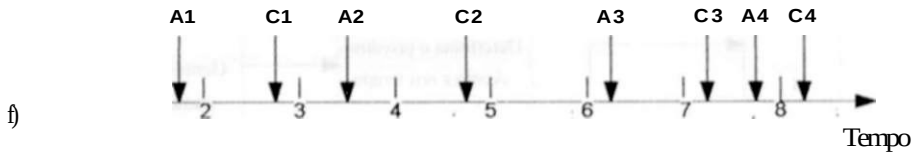
4.3.1 Métodos de gerenciamento do tempo ,

Como já dito anteriormente existem duas classes dc sistema: os estáticos c os dinâmicos. O primeiro não depende do tempo e o segundo depende. Os modelos estáticos evidenciam as inierrelações entre entidades e atributos quando o sistema se encontra em equilíbrio estático e nos permite observar o efeito provocado pela mudança no valor de um atributo sobre o sistema, e não se consegue apreender nenhuma informação sobre a maneira como essa mudança ocorreu. Já o modelo dinâmico permite que a mudança em um atributo possa ser colocado como uma função do tempo c daí o modo pelo qual as mudanças ocorrem podem ser estudados.

("orno os modelos dinâmicos são dependentes do tempo, e podem ser colocados como uma função dele, um simulador utiliza esta função que descreve o comportamento do sistema para descrever as mudanças ocorridas no seu decorrer, daí a denominação de gerenciamento do tempo. **Existem** várias formas de gerência de tempo ou variantes destas, duas das mais importantes são: varredura periódica (*periodic scan*) e varredura de eventos (***evení*** *scan*),|Ciraybcal|. |Kobayashi| utiliza as denominações modo síncrono (*synchronoiis* ***timing*** *or unil-time* advance) e modo assíncrono (*asynchronous* ***timing*** *or cvcnl* advance). Adotaremos a denominação de |Ciraybcal|. A varredura periódica utiliza um esquema de intervalo de tempo fixo. Isto é feito dividindo o intervalo total de simulação em unidades iguais dc tempo, o relógio da simulação é avançado em intervalos regulares de tempo. Durante um determinado intervalo é feita a verificação da ocorrência dc eventos, se ocorre algum evento este é simulado, se.nenhum evento ocorre nada é feito e após decorrido o intervalo básico de relógio este é avançado. A tlgura 4.3.1.1 (a) c (b) abaixo ilustra este esquema.



(a)



(b)

Figura 4.3.1.1 - Abordagem varredura periódica

Por inspeção da figura 4.3.1.1(a) verilíca-se que os eventos l\2 e E3 e os eventos E5,E6 e Ií7 serão tratados como simultâneos erroneamente. Na figura 4.3.1.1 (b) assume-se que o sistema esta vazio no inicio e no fim. Suponha também que os Ax indiquem as chegadas dc eventos e os Cx indiquem suas conclusões, então, por exemplo, os eventos C3 e A4 são considerados como acontecendo no mesmo tempo.

Na abordagem de varredura de evento o avanço do relógio é leito em intervalos de tamanho variável, nessa abordagem os eventos são disparados a partir da consulta a uma lista dc eventos, e o relógio só é avançado após sua simulação. A figura 4.3.1.1 (b) acima poderia ilustrar este esquema se a interpretássemos do seguinte modo: o relógio iniciaria com o evento Al e seria avançado para o evento Cl independentemente do tamanho deste intervalo e assim sucessivamente. A figura 4.3.1.2 abaixo ilustra a utilização da abordagem varredura de eventos.

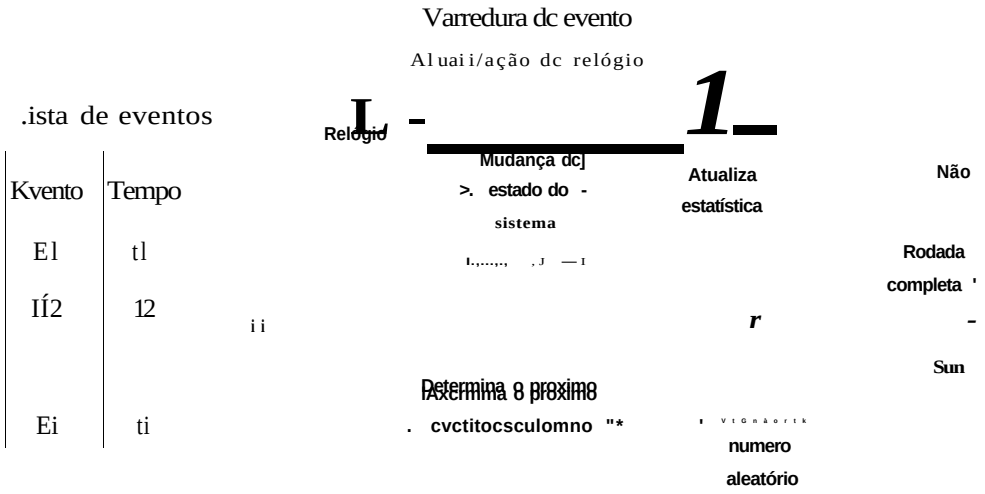


Figura 4.3.2.2 - Abordagem varredura de eventos

.

Quando utiliza-se a abordagem de varredura periódica a primeira providência c a determinação do tamanho ideal de intervalo a ser adotado. Ai alguns cuidados devem ser tomados, por exemplo se o tamanho do intervalo é muito grande; eventos separados no tempo aparecem como tendo acontecido simultaneamente, e assim perde-se informação sobre a operação do sistema. Então o tamanho do intervalo deverá ser pequeno o suficiente para que a probabilidade de ocorrência de dois ou mais eventos cm um mesmo intervalo de tempo seja pequena, lissa discussão nos leva a concluir que o intervalo de tempo deverá ser tão pequeno quanto possível, só que quando isso é feito observando apenas o fato de minimizar a probabilidade de ocorrência de eventos simultâneos, aumenta-se a precisão e consequentemente a necessidade de cálculos. Hsse aumento no número de cálculos implica em um maior tempo para simulação c consequentemente um aumento do custo da simulação. Então existe claramente um *Iradc-ojf* **entre** a necessidade de precisão e o incremento no custo da simulação em computador. Ioda essa análise compete ao estudioso pois determinados modelos não apresentam uma criticidade muito grande em função do tempo exato de ocorrência de um evento; em oposição existem modelos para os quais essa consideração é fundamental sob pena da avaliação do sistema se tornar muito pobre ou mesmo divergir do sistema real.

A outra abordagem requer algum esquema para determinar quando um evento irá ocorrer. Novos eventos são colocados em uma lista ou fila em ordem cronológica e então o avanço do relógio pode ser determinado por uma simples varredura na lista de eventos.

4.3.1.1 Uma comparação entre as duas abordagens

Pelo acima descrito verifica-se que a abordagem varredura de eventos evita alguns dos problemas inerentes á abordagem varredura periódica. Nesta última é exigido que o analista forneça um tamanho

ótimo de intervalo, que é desconhecido e deve ser estimado através de um trade-oíV entre precisão c tempo de processamento (custo). Outra questão que tem que ser vista c que existe a possibilidade de perda de informações pelo fato da ocorrência de eventos cm um mesmo intervalo de tempo o que leva o analista a ter, erroneamente, dois eventos apresentados como simultâneos quando não o são no sistema real Além disso o tempo exato de ocorrência do evento é desconhecido, pelo tato de cada evento ser tratado como se ocorresse no final do intervalo no qual ele acontece; um problema adicional é que nos períodos de inatividade o simulador está trabalhando fazendo nada (simplesmente avançando **o** relógio). Este problema parece pequeno mas se o sistema é caracterizado por longos períodos de inatividade e curtos períodos de intensa atividade, lemos uma significativa parcela do tempo para execução da simulação gasto com a atualização do relógio. Ao contrário a abordagem de varredura de eventos não exige o intervalo artificial de tempo imposto na abordagem anterior, seu esquema de avanço de relógio é implementado pela montagem de uma lista de eventos dc forma bastante simples. O simulador está sempre trabalhando com tarefas dc simulação de eventos. Dois eventos aqui só serão simultâneos na simulação se de fato eles o forem no modelo do sistema real.

4.3.2. Ceração de objetos

Como já definido anteriormente, um sistema é uma coleção dc objetos e interações, logo para a realização de uma simulação, de algum modo, devemos representar os objetos que compõem **o** modelo a ser simulado Nesse sentido alguns requisitos devem ser atendidos de forma a não introduzirim s erros de interpretação dos resultados da simulação:

- Um objeto ou entidade deve ser único cm um modelo (programa) de simulação;
- O estado de um objeto ou entidade em qualquer instante de tempo é representado pelo valor corrente dc seu conjunto dc atributos (vetor de estado).

A unicidade de uma entidade pode ser conseguida numerando adequadamente as entidades que chegam. **Existem** entidades que estão presentes em todo processo de simulação, são denominadas entidades permanentes, outras são introduzidas e retiradas ao longo da simulação, estas são denominadas entidades transitórias. A chegada de uma entidade transitória no sistema é um evento pois altera o estado do sistema pela adição de uma entidade. O estado do sistema pode ser definido como o conjunto dc atributos das entidades presentes no sistema em um determinado instante dc **tempo.** Podemos usar, basicamente, dois tipos de marcação de chegadas de entidades no processo de simulação: em períodos fixos ou amostrando a distribuição entrechegadas. freqüentemente utiliza-se um **esquema** de ***hooístrapping*** para geração dos tempos de chegada para as entidades transitórias sucessivas (O tempo de chegada da primeira entidade é gerado por amostragem da distribuição apropriada e o evento é colocado na lista de eventos, quando o evento é concluído o relógio é avançado para esse ponto, o estado do sistema é atualizado **e** o tempo dc chegada da próxima entidade é gerado). líxiste uma forma alternativa para esse esquema ***bootstrapping,*** gerando todos os tempos de chegada e armazenando-os em uma tabela (esse esquema no entanto exige uma maior disponibilidade de memória).

Uma vez que determinamos o tempo dc chegada para a entidade transitória, devemos atribuir valores para seus atributos. Podemos fazer isso ou no tempo de chegada gerado, ou no instante no qual o evento ocorreu. O fator determinante para adoção de um destes esquemas de atribuição de valor para os atributo é o falo de sabermos se o valor do atributo é dependente ou não do estado. Se o atributo é dependente do estado então eles serão designados no instante da chegada e serão feitos a partir da amostragem simulada de uma dada distribuição, usando geração de números aleatórios.

4.3.3 Eventos e sincronização de eventos

Um evento c caracterizado pela mudança no estado do sistema em algum instante no tempo. Sob esse ponto de vista um evento não tem duração, acontece em um ponto. Alguns aspectos dc muita importância para simulação de sistemas são: encadeamento (***scheduling)*** e sincronização dc eventos.

Quando dois eventos provocam a mesma mudança no estado do sistema eles são ditos ser do mesmo tipo Então uma forma de agrupar eventos c por tipo. Nesse contexto podemos distinguir duas grandes classes de eventos: eventos de sistema e eventos de programa. Um evento de sistema tem um correspondente equivalente no sistema real. Um evento dc programa não tem nada a ver com o sistema real mas unicamente com o programa de simulação (o instante em que a impressão das estatístieas da simulação será realizado é um exemplo de evento de programa). A princípio podem nos parecei artificiais, porém devemos ter o cuidado de encadeá-los da mesma forma que fazemos com os eventos de sistema. Podemos também caracterizar os eventos em decisão e não-decisão; o primeiro caracteriza-se pelo fato de alguma decisão ter de ser tomada sobre o efeito dc sua ocorrência, e o segundo por não implicar na ocorrência dc uma decisão.

('orno já dito anteriormente, quando estamos simulando sistemas dinâmicos a ordem em que os eventos ocorrem no tempo pode ter, e, normalmente têm, importância fundamental no resultado da simulação. Então dois aspectos importantes da simulação são: sequenciamento e sincronização de eventos A cada vez que o relógio da simulação avança, uma rotina é executada para aquele tipo de evento e daí atualiza-se o estado do sistema e, assim, o efeito do evento no modelo espelha a ocorrência do evento no sistema real. O modelo de simulação também deve prover uma disciplina dc encadeamento de eventos de tal forma que, se dois ou mais eventos têm possibilidade de ocorrer simultaneamente, o modelo deve decidir qual deles será executado primeiro, esta decisão deve relletir o comportamento real do sistema No entanto existem casos em que os eventos ocorrem simultaneamente no modelo real e ai a questão de sua representação será função do tipo de arquitetura da máquina que está sendo usada na simulação; se for

uma máquina seqüencial deve-se estabelecer uma disciplina de ordenamento para os eventos, se for uma máquina paralela os eventos deverão ser organizados da forma cm que se apresentam no sistema real.

A questão do sequenciamento de eventos c muito importante porque existem eventos que ao serem simulados geram a ocorrência dc outros, ou podem causar o cancelamento dc eventos presentes na lista de eventos, alterar a ordem de realização dos eventos da lista, etc, portanto eventos não sequenciados corretamente podem ter efeitos devastadores na performance do sistema.

O número dc eventos e a complexidade da execução de rotinas são determinantes tanto do *run lime* quanto do custo da simulação.

4.3.4. Gerenciamento dc fila c processamento de lista

Na grande maioria dos sistemas simulados encontramos filas, isso, naturalmente, ocorre porque em sua grande maioria os sistemas possuem alguns recursos limitados (o número de caixas cm um banco, **o** número de impressoras laser em uma rede, o número de circuitos em uma determinada rola de comutação, etc). As filas então estão presentes por causa da competição existente por esses recursos escassos, c dessa forma é fácil ver que a representação adequada e sua conseqüente manipulação c de fundamental importância na modelagem e desenvolvimento da simulação.

4.3.4.1 Alguns conceitos básicos de teoria das filas

Os fatores básicos para estudo de filas são:

i) A taxa e **o** padrão de chegadas de consumidores;

ii) A distribuição dc probabilidade do tempo de serviço;

iii) A ordem na qual os consumidores são atendidos;

iv) O número de servidores ou canais;

v) A distribuição de probabilidade do tempo de espera dos consumidores.

Os fatores apontados acima devem ser de alguma forma explicitados dc modo a que no processo de construção do modelo se possa introduzi-los de maneira prática e as variações de seus parâmetros possam auxiliar o analista na avaliação e validação do modelo. A partir desse ponto temos que verificar as relações existentes entre, por exemplo:

a) Os parâmetros da distribuição do tempo de serviço e a taxa média de chegada de consumidores,

b) O número de canais e a taxa média de chegadas,

c) A disciplina da fila e o número de canais de serviço.

O propósito de um modelo matemático simbólico é dc fazer predições possíveis sobre o comportamento do processo quando mudamos suas condições de operação ou, as vezes, sobre processos que nem existem ainda.

O instante de tempo no qual um consumidor ou grupo de consumidores entra num sistema de fila é chamado instante de chegada, e o intervalo entre instantes de chegada consecutivos é chamado de tempo cntrcchcgadas. Quando estudamos o processo dc formação de fila, devemos, de algum modo, coletar dados no intervalo cntrcchcgadas de forma adequada para permitir encontrarmos resposta para duas perguntas básicas: os tempos cntrcchcgada são independentes ou existe correlação entre eles? os tempos cntrcchcgadas estão correlacionados com outros aspectos do sistema de fila?

Dois exemplos práticos disso são:

1) No sistema telefónico estes tempos sào independentes.

2) Nas chegadas dc aeronaves cm um aeroporto estes tempos (horários) são planejados. As chegadas e partidas sào escalonadas ao longo do dia, então os tempos entrechegadas não são independentes.

Existem algumas distribuições que são bastante exploradas no processo de chegada deconsumidores, tais como: exponencial {f(t) *Xc*" }, onde X. c a taxa media de chegada por unidade dc tempo, Poisson {P|/• n| ë"" Wn! }(a probabilidade dc n chegadas cm um intervalo de tempo unitário), etc.

O intervalo de tempo entre o instante no qual o consumidor inicia seu atendimento ate que o atendimento é concluído é conhecido como tempo de serviço. É importante distinguir entre o tempo de serviço e o bloco dc tempo do servidor. Este é o tempo decorrido desde o início de atendimento de um consumidor e o inicio dc atendimento do próximo, diz respeito à medida dc performance do Servidor, listes dois tempos não necessariamente são iguais, por exemplo, no caixa de um banco tendo um consumidor iniciado seu atendimento por parte do servidor o tempo de serviço se esgota quando o consumidor sai do caixa, o bloco de tempo do servidor permanecerá sendo contado enquanto este autentica documentos, arruma-os, etc e em seguida inicia o atendimento a outro consumidor.

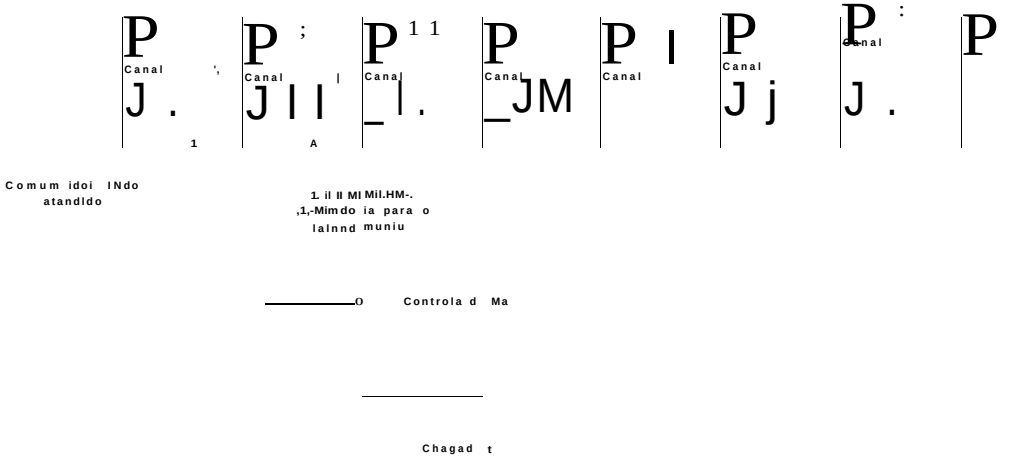


Figura 4.3.4.1.1 - Exemplo de atendimento bancário.

Por simples inspeção da figura acima algumas perguntas nos assaltam e devem ser respondidas de forma mais próxima do sistema de fila real. Utilizando-se simulação consegue-sc algumas, senão todas essas respostas. As perguntas mais importantes são:

I) lodos os consumidores são do mesmo tipo, isto é, seus tempos de serviço tem a mesma distribuição de probabilidade?

2) Todos os servidores são idênticos, islo é, os tempos de serviço que eles provêem pertencem a mesma distribuição dc probabilidade?

3) Os tempos dc serviço estão correlacionados com outros aspectos do sistema?

4) A distribuição do tempo dc serviço é estacionária?

A investigação das respostas a essas perguntas nos levam a formular modelos mais ou menos complexos conforme as respostas que obtivermos.

As disciplinas mais comuns dc gerenciamento de filas sào:

- I M O (lirst-in, first-out) na qual o consumidor que está há mais tempo esperando tem o controle.

- LIFO (last-in, First-out) na qual o consumidor que está há menos tempo esperando tem o controle.

- RAND na qual o consumidor que passará a ter o controle será escolhido por algum mecanismo aleatório.

- PR! neste tipo de disciplina existem inúmeros esquemas, no qual o consumidor a ler o controle será escolhido por um determinado valor de atributo ou por um conjunto dc valores de atributos.

Em uma grande parte dos textos de teoria das íilas utiliza-se uma notação introduzida em 1951 por D G. Kcndall, que apresenta-se da seguinte forma: A/B/s, onde A especifica o padrão dc entrada, B a distribuição do tempo de serviço e s o número de servidores. |Lee| utiliza uma variante desta notação a saber: A/B/s:(d/e), onde A, B c s têm a mesma função da notação de Kcndall c d especifica o número máximo de consumidores que podem ser admitidos no sistema em um mesmo instante de tempo e identifica a disciplina da fila.

Sempre que ocorre uma chegada, uma entidade é gerada c encadeada (***Schedule d***) e os valores de seus atributos sào estabelecidos, ou a partir de uma determinada distribuição ou através de um outro procedimento estabelecido, uma forma conveniente de representar a chegada dc uma entidade no sistema é através da ativação de um registro como mostrado na figura 4.3.4.1 abaixo.

Figura 4.3.4.1 - Exemplo de registro de uma entidade em uma simulação

Os valores dos atributos são ajustados para **refletir** a ocorrência do evento de tal modo que o modelo do sistema se adeque ao comportamento real do sistema. **[Graybeal]** define uma fila como-uma coleção de chegadas de entidades esperando por um recurso comum, uma forma conveniente de representarmos uma fila é através de uma lista. **0** objetivo de utilizarmos uma lista para representar uma fila é o fato de que a lista permite que façamos inscrções/delcçõcs de registros de uma forma simples e eficaz do ponto de vista do programador, além dc permitir toda sorte de consultas tanto a um registro específico quanto a um ou mais atributos dc um ou vários registros.

Como no processamento dc listas objetos são representados em um computador de modo a explicitar as relações entre eles no conjunto ordenado dc entrada. **Este** ordenamento é que toma a lista uma ferramenta conveniente para representar a fila. **Três** considerações básicas têm que ser observadas na construção dc listas:

1) Dimensão da lista;

2) Densidade da lista;

3) Seqüência da lista.

Quando da construção dc uma lista, muitos fatores têm **influencia** sobre a forma operacional como ela irá ser concebida. Vamos aqui enumerar alguns destes fatores.

i) Aplicação;

ii) Técnica dc busca;

iii) forma dc alteração.

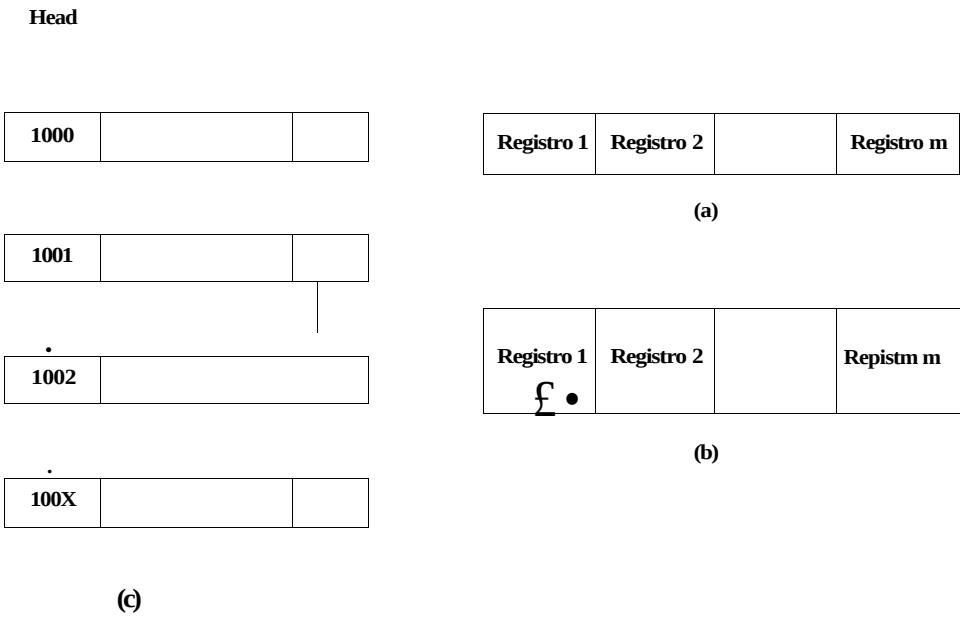
('orno uma lista é uma coleção dc registros estes podem ser alocados dc diversas formas **dependendo** de como a lista foi construída. Três formas básicas dc alocação destes registros são:

a) Serialmente;

b) Diretamente alocado,

c) Alocação indireta

Evidentemente a primeira forma de alocação c a mais simples e também a menos eficiente. A segunda forma utili/.a uma chave cm cada registro dc forma que o acesso é feito através desta chave. Na terceira forma dc alocação os registros são enlaçados usando-sc apontadores, esse esquema é um dos mais eficientes e permite diversos tipos de construção tais como: lista simplesmente enlaçada, lista duplamente enlaçada, lista cíclica, etc, na figura 4.3.4.2 abaixo temos exemplos dessa alocação.



•igura 4.3.4.2 - Exemplo de formas de alocação.

Na figura 4.3.4.2(a) acima temos um exemplo de alocação serial dc registros, em 5 3.4.2(b) alocação direta e finalmente em 4.3.4.2(c) alocação indireta (no caso, com link simples). Quando aumentamos o número de apontadores aumenta na mesma proporção a demanda por memória

4.3.5. Coleta e processamento dos dados

Após a construção do modelo, o passo seguinte a ser dado **é** a simulação propriamente dita, c ai uma tarefa importante **é** a coleta e o registro dos dados, |Naylor et. al.| afirma que c simplesmente impossível formular um problema ou um conjunto dc objetivos experimentais sem acesso a informações (quantitativas ou não) a respeito do sistema que está sendo examinado, isto **é,** uma certa quantidade dc dados deve ter sido coletada e processada antes que qualquer problema possa ser definido, esse fato no entanto **é** irrelevante no que diz respeito a ordem, a questão fundamental c que os dois passos devem ser executados

|Naylor et. al.| apontam cinco razões pelas quais **é** necessário um processamento **eficiente** dos dados para que a simulação possa obter sucesso:

i) Informação (dados) descritivos c quantitativos a respeito do sistema c primordial para a formulação do problema;

ii) Os dados quando colocados em um formato adequado podem sugerir hipóteses sustentáveis que poderão ser utilizadas na construção do modelo;

iii) Os dados por si só podem sugerir melhoramentos/refinamentos para modelos **matemáticos** já existentes;

iv) Os dados após a formatação final podem ser utilizados na estimação de parâmetros das características operacionais que relacionam as variáveis exógenas, endógenas c de estado do sistema;

v) Sem dados seria impossível validar um modelo de simulação.

|Naylor et. al. | identifica seis funções importantes do processamento dos dados que fazem parte do processo de simulação:

a) Coleta

b) Registro »

c) Conversão

d) Transmissão

e) Manipulação

I) Saída

Na verdade a coleta c o registro dos dados ocorrem simultaneamente visto que a coleta dos dados implicam que os dados foram ou estão sendo registrados. A coleta dc dados é o processo de recolhimento dc fatos disponíveis.

A escolha do tamanho da amostra é uma das mais importantes decisões a ser tomada. Em relação a isso |Kobayashi| adverte que a escolha arbitrária do tamanho da amostra é completamente inapropriado. A forma correta de fixarmos um tamanho dc amostra c através de uma análise estatística, esse procedimento garante a adequação entre o tamanho da amostra e a precisão requerida. Três métodos são propostos:

1) Método de replicação independente (***independenl-replicalion)***

2) Método da corrida simples (***single-run)***

3) Método regenerativo

O tipo mais comum de coleta de dados é a totalização (número total dc consumidores em uma fila em um dado instante dc tempo, total dc ocorrências dc um determinado evento, etc). Hsse tipo de dado e facilmente coletado pela utilização de contadores inicializados no início da simulação; estes dados então são manipulados para completar a simulação e provêem um resumo das medidas de performance

Um cuidado que devemos ter é em relação ao local onde estes dados serão coletados, a forma mais óbvia de faze-lo é coletar o dado durante o processamento de um evento que afete a variável de interesse.

Neste aspecto as linguagens dc propósito especiais usadas em simulação apresentam vantagem em relação as linguagens de propósito geral por facilitarem a coleta de dados.

A conversão dos dados normalmente ocorre porque o meio no qual os dados são registrados no primeiro estágio do processamento não são os mais adequados para utilização nos estágios subsequentes, essa é uma função que influencia muito o desempenho relativo a eficiência do processamento dos dados

Um outro problema que eventualmente pode existir é o de transmissão dos dados, isto é, deslocamento de informações de um endereço para outro, esse problema nos dias de hoje já não tem lanta relevância quanto no passado próximo, devido as facilidades de comunicação disponíveis.

Com todas as funções anteriores executadas teremos então que manipular os dados, essa função requer tarefas tais como: classificação, reunião de itens, recuperação de informações bem como operações lógicas c aritméticas, essas operações teoricamente podem ser realizadas com ou sem a ajuda de um computador, na prática o que se verifica é que sem o computador simulações de sistemas de médio ou grande porte são inviáveis.

4.3.6. Análise do resultado da simulação

Como dito em |CiraybealJ a partir do momento cm que o simulador foi projetado, testado c ajustado, a simulação pode ser encerrada e os resultados podem ser interpretados c inferências sobre a operação do sistema real podem .ser extraídas. Numerosas armadilhas esperam o inocente analista na interpretação dos dados da simulação.

O grande objetivo da simulação de um sistema cm computador é verificar, sem colocar cm risco o sistema real, se este já estiver em operação ou não, as reações desse aos mais diversos valores de parâmetros de seus objetos, representando com isso alterações no sistema real quando submetido as mais diversas condições de operação, sem o ónus financeiro de experimentar estas condições de interesse cm um sistema real ou um protótipo deste. Os aspectos do sistema que estão sendo estudados são cm geral medidas de performance c respostas das variáveis de interesse. A tarefa do analista é então verificar as respostas do sistema quando submetido às mais diversas configurações relativas a diferentes valores atribuidos às variáveis do modelo. A questão importante aqui é o fato de que em toda simulação existe um período inicial em que o sistema passa por um transitório e os dados obtidos nessa fase não são dados

típicos da operação normal do sistema. [Kobayashi] chama atenção para um aspecto realmente importante Kxistc uma diferença acentuada em relação a noção de transitório cm sistema estocástico e sistema determinístico (por exemplo, um circuito elétrico), onde o comportamento transitório é completamente caracterizado por sua resposta a função impulso. Isto acontece porque as condições iniciais escolhidas para o modelo de simulação não são características do sistema.

Sc queremos obter medidas dc performance realistas do modelo em um estado estável, a primeira tarefa é eliminar o viés estalislico introduzido no resultado da simulação pelo período do transitório A solução óbvia para o problema é então só realizar a coleta de dados após o sistema simulado atingir um esl ido estável. Nesse ponto então as duas perguntas objetivas que devem ser respondidas são:

- Como reconhecer que o modelo atingiu um estado estável?

- Como eliminar o eleito do viés?

Algumas técnicas têm sido propostas com o objetivo dc determinar o estado estável, por exemplo, podemos computar a média móvel da medida dc performance de tal modo que quando a diferença entre co nputações sucessivas não for significativa admite-se que o estado estável foi alcançado, uma segunda forma seria observar-se uma seqüência dc observações da medida em questão c quando o número de observações acima c abaixo da média for aproximadamente o mesmo admitir que o estado procurado foi en< ontrado Uma terceira e mais sofisticada forma dc faze-lo é utilizando o teste de Kolmogorov-Smirvov que indicará quando uma tendência a estabilização ocorreu c aí admite-se que encontrou-se o estado estável. Em relação a segunda pergunta basicamente existem três formas de se conseguir a eliminação do viés:

- Aumentar o tempo dc simulação

- Reinicializar o sistema com os valores já encontrados para o estado estável

- Introduzir um período de start-up para permitir o simulador atingir o estado estável e ai limpar as estatísticas acumuladas e começar uma nova coleta.

A primeira técnica tem duas grandes desvantagens: a primeira é o fato de não eliminar o viés, apenas minimizando-0 e a segunda é o aumento do custo dc simulação. As outras duas minimizam o tempo de simulação e, se o estado estável de fato for encontrado, eliminar o viés.

Existem alguns sistemas que não admitem um estado estável, neste caso o conjunto dc estatísticas tem valor limitado pois as observações estão muito dispersas cm torno de um valor médio.

Outro problema que temos que enfrentar na análise da simulação diz respeito a escolha da metodologia que nos proporcionam uma amostra estatisticamente válida, pois esse é um requisito básico para que se possam fazer inferências sobre os verdadeiros valores das medidas de performance. A determinação dessa metodologia evidentemente é restrita ao caso em que se trabalha com modelos cstocáslicos. Duas hipóteses têm que ser consideradas: se o modelo for finito c, necessariamente discreto, o total de eventos do conjunto de eventos possíveis poderia até ser, eventualmente, gerado dependendo do tempo de simulação ser suficiente para isso. Se o modelo for contínuo isso fatalmente não ocorrerá. Como trabalha-se com a hipótese de que o tempo de simulação é caro, então o objetivo é, ao mesmo tempo, minimizar o tempo de simulação e obter estimadores confiáveis para as medidas de performance. Logo técnicas dc amostragem estatísticas adequadas tornam-se imprescindíveis para obtenção de estimadores confiáveis.

Deve-se ter em mente que com a popularização e o aumento espantoso de performance obtido no desenvolvimento dos computadores pessoais, a hipótese de custo dc simulação tem perdido muito cie sua força tradicional

I Saltb] propõe uma nova técnica para aumentar a precisão do resultado da simulação, sugere o abandono da abordagem atual, amostragem aleatória simples, por uma nova abordagem que ele denomina amostragem descritiva. Na amostragem aleatória simples tem por objetivo preservar a independência estatística entre os elementos de uma mesma entrada. |Salib| propõe que se abra mão dessa independência.

ele então demonstra que as distorções resultantes são irrelevantes c que o ganho de precisão jus! i fica sua pro >osta.

INaylor et. al. | faz a análise dos dados da simulação a partir dc 1res etapas, quais sejam:

i) Coleta e processamento dos dados da simulação,

ii) Organização da estatística dos testes,

iii) Interpretação dos resultados.

4.3 7. Avaliação do modelo de simulação

Nesse ponto o nosso modelo já foi projetado e codificado, o trabalho do analista agora e validá-lo. Validar um modelo c determinar o quanto o modelo pode predizer tanto o comportamento quanto a pei umance do sistema real. |Graybeal| aponta dois fatores fundamentais para que um modelo possa pie izer a performance dc um sistema com precisão: validade do modelo e a confiabilidade das medidas de pei ormance fornecida por cie. A validade de um modelo de simulação depende da precisão do modelo em rep escutar o sistema real. Para isso o modelo tem que ser suficientemente detalhado para atender a nec ssidade de informações do analista sobre os aspectos relevantes do sistema, então o único método realmente válido para isso é o julgamento de sua performance. Para que se possa assumir que o modelo sob simulação é realmente válido este deve permitir que as inferências feitas a partir dos resultados fornecidos por ele nos leve a conclusões corretas sobre o sistema.

O processo de modelagem é dividido em duas fases: conceituai c dc implementação. Na fase conceituai determina-se o fluxo lógico do sistema e as interrelações entre os vários subsistemas Na fase de implementação serão selecionados e qualificados procedimentos para o modelo e inclui ainda a codificação e a utilização efetiva deste. Essas duas fases têm que ser validadas. Na primeira dois procedimentos podem ser adotados: ter o modelo revisto por um observador qualificado (***a desinleresled qualified observer)***

que pode **confirmar** ou não a decisão tomada pelo analista na formulação do modelo, ou fazer o percurso em ordem inversa, islo é, dc trás para frente. Na segunda procedimentos práticos são utilizados tais como: seguir o desenvolvimento do fluxo lógico do modelo, seguir a codificação do programa de cada subsistema, verificar a integração dos módulos dc subsistemas dentro do modelo completo.

Dc acordo com (NaylorJ validar qualquer tipo dc modelo eqüivale a demonstrar que é verdadeiro; todavia, demonstrar que um modelo é verdadeiro implica em:

1) Que sc tem estabelecido um conjunto dc critérios para diferenciar entre os modelos que são verdadeiros e os que são falsos;

2) Que sc possui a habilidade de aplicar facilmente estes critérios a qualquer modelo dado.

Em função da dificuldade dc se conseguir um consenso sobre um conjunto dc critérios para estabelecer se um modelo está validado ou não, propuseram, por exemplo, que deve-se concentrar no grau dc confirmação de um modelo em lugar de estabelecer se o modelo está validado. [Nayloi | sugere medidas e técnicas específicas para provar a adequação de ajuste de um modelo de simulação (o grau dc conformidade das séries temporais simuladas com os dados observados) devidas a R. M. Cyert, a saber:

a) O número de pontos dc deflexão;

b) O tempo dos pontos dc deflexão;

c) A direção dos pontos de deflexão;

d) A amplitude das flutuações de dados segmentos dc tempo;

e) A simultaneidade dos pontos de deflexão de diferentes variáveis;

0 A igualdade exata dos valores das variáveis.

Algumas técnicas estatísticas disponíveis para comprovar a adequação de ajuste dos modelos dc simulação são: . ,

i) Análise de variância - é um conjunto dc técnicas para analisar dados que se poderá utilizai para comprovar a hipótese de que a média (ou variância) de uma série gerada por um experimento dc simulação

é igual a media (ou variância) da serie observada (pré-supõe) normalidade, independência estatística e mesma variância

ii) Qui-quadrado - é uma prova estatística empregada para comprovar a hipótese de que o conjunto dc dados gerados por um modelo de simulação possui a mesma distribuição de freqüência que o conjunto de dados históricos observados.

iii) leste de Kolmogorov-Smirnov - c do tipo de prova não-paramélrica relacionada com o grau dc concordância que existe entre a distribuição dc um conjunto de valores da mostra (serie simulada) c alguma distribuição teórica especificada (distribuição dc dados reais).

iv) Análise de regressão - possibilidade dc efetuar regressões da série real sobre a série gerada e comprovar se as equações dc regressão resultantes têm interseções.

v) Análise espectral - tem como objetivo principal a qualificação e avaliação dos dados correlacionados uma vez que estes sc transformam dentro do domínio da freqüência.

vi) Coeficiente dc desigualdade dc Theil - proporciona um índice que determina o que um modelo de simulação proporcione predições retrospectivas Pj de dados históricos observados Aj:

Eq. 4.3.7.1 (7 $$\frac{\sqrt{1/n\sum(P_i - A_i)^2}}{\sqrt{1/n\sum P_i^2} + \sqrt{1/n\sum A_i^2}}$$

U varia entre 0 e I. Se U 0, as predições são perfeitas, se, ao contrário, U I, as predições são muito ruins

A discussão sobre avaliação/validação de modelos pode ser mais aprofundada em [Naylor et. al.|, **I** Naylor**I** e | Springer et. al.|, dc qualquer modo não se consegue um conjunto dc pontos dc vista que se possa tomar como conclusivo c consensual, sugerimos cm particular o capítulo 8 de | Naylor et. al.|.

Provavelmente nenhum analista está completamente seguro da validade do modelo de simulação. **Em** muitos casos a validade é assumida até prova em contrário. Isto poderá ajudar muito depois de o analista completar seu trabalho. 0 analista deve, dessa forma, fazer um sério esforço para validar o modelo antes de usa-lo |(iraybeal|

4.3.8 Linuuaacns para simulação de sistemas discretos

A partir do momento cm que temos um modelo conceituai definido, a tarefa agora é de implementação, para isso uma fase importante é a de definição da linguagem, em que o programa que simulará o modelo, será escrito (codificado). Existe uma grande quantidade de linguagens que podem ser utilizadas cm simulação de sistemas discretos, variando desde linguagens de baixo nível até linguagens especializadas em simulação como **GPSS, SIMSCRIPT** c **GASP.** Alguns fatores que **influenciam** na escolha dc uma linguagem são:

- familiaridade do analista com a linguagem;

A facilidade de aprendizado de uma linguagem se o analista não a conhece;

- As linguagens existentes na instalação onde trabalha;
- A complexidade do modelo,
- As facilidades de análise e apresentação dc resultados, etc.

A tareia de escolha de uma linguagem de simulação não c, de forma alguma, uma tareia trivial, c essa escolha tem um papel determinante no desempenho futuro da simulação. Isso significa que essa escolha impaclará diretamente no tempo requerido e consequentemente no seu custo. Uma escolha errada da linguagem pode até inviabilizar o projeto dc simulação. O propósito dc muitos estudos é comparar alternativas ou projetar cenários, então configuração e parâmetros do sistema, políticas dc encadeamento, etc, o programa de simulação deve ser suficientemente flexível para permitir, através de mudanças simples.

que estas alternativas sejam geradas e analisadas. A grande maioria dos modelos de simulação apresentam características comuns tais como:

1) Geração dc variáveis aleatórias;

2) Gerência do tempo de simulação;

3) Manipulação de rotinas para simular execução de eventos,

4) Gerência de filas,

5) ("oleia de dados,

6) Resumo e analise de dados:

7) formatação e impressão dos resultados.

Muitos sistemas são caracterizados por algum comportamento estocástico, seja cie o tempo entrechegadas e requisição de serviços cm um sistema dc filas ou o tempo entre falhas em um equipamento em um sistema dc produção

Km uma linguagem de simulação então alguns requisitos são imprescindíveis. Alguns destes requisitos **são**

- Prover ou permitir fácil desenvolvimento de facilidade para gerar ou transformar variáveis aleatórias,

- Deve permitir fácil representação e manipulação do tempo de simulação;

- Eficiente capacidade dc processamento de listas;

- Facilidade para coleta dc dados;

- facilidade para saida de dados sumarizados para aumentar a eficiência da análise;

- Prover funções estatísticas predefinidas;

- flexibilidade para apresentação e formatação dc relatórios;

- facilidade dc assistência de *debu^ging*

- familiaridade ou conhecimento da linguagem;

- **Facilidade** de aprendizado da linguagem;

- Capacidade dc expansão e adaptação.

Além disso o programador, ou muitas vezes, o próprio analista, quando empenhado na tarefa dc codificar programas para simulação deve estar familiarizado com as sutilezas dos mecanismos, tanto dc hardware quanto dc software dos sistemas que serão simulados, ele deverá garantir que estes são todos tratados dc forma **adequada** no modelo do simulador, **0** eleito dc qualquer suposição para propiciar simplificação pode trazer problemas sérios relativos a capacidade do modelo representar adequadamente o sistema real, e essas suposições equivocadas podem levar o modelo a uma divergência muito grande cai relação ao sistema a ser simulado.

Quando o modelo é construído, validado logicamente e codificado cm forma dc um programa, sua entrada será alimentada por um determinado período de tempo que representará o volume e tipo de tráfego que o sistema real irá comportar. O programa de simulação então imprime certas estatísticas sobre 0 comportamento do modelo, tal como o desenvolvimento dc uma fila, o tempo de resposta obtido, quão lentamente os fatores críticos foram carregados, etc.

Saidas típicas de um programa simulador são:

- O tamanho máximo e médio de fila onde quer que elas ocorram;

- Um histograma do tempo dc trânsito ou tempo de resposta entre pontos em um **sistema;**

- A utilização de facilidades, ou medida do tempo de carga dos vários elementos do sistema;

- A quantidade dc memória requerida para buífer ou fila;

- **0** throughput máximo do modelo, etc.

Os modelos, a partir do momento que foram construídos, podem ser ajustados e experimentados com facilidade, diferentes volumes c, talvez, tipos de throughput podem alimentar o modelo para ver como ele se comporta, os efeitos das mudanças podem ser facilmente investigados, os modelos podem formar uma ferramenta para decisão entre diversos modos dc operação, eles podem ser usados para ajustar

argumentos, parâmetros c variáveis, possibilitando um refinamento cada vez maior tanto dos resultados como também da própria concepção inicial do modelo.

[Kobayashi] apresenta um exemplo didático de simulação de uma fila cíclica implementada em FORTRAN cem GPSS.

4.4 Simulação de sistemas contínuos

Até este ponto temos nos detido cm sistemas discretos, caracterizados por mudanças em seu estado ocorrendo de forma puntual no tempo. Os modelos para sistemas discretos, no qual o interesse principal consiste em detectar a ocorrência ou não de eventos cm um dado instante de tempo, são geralmente estáveis cm termos de equações lógicas. Estas equações delineiam as condições que devem existi: para que um dado evento ocorra.

Agora discutiremos os sistemas contínuos, nos quais as mudanças ocorrem de forma suave e continua ao longo do tempo, estes modelos cm geral são descritos cm forma dc equações diferenciais ou de um conjunto delas. A descrição de um sistema contínuo geralmente envolve a especificação da taxa com a qual certos atributos mudam cm função do tempo.

Exemplos dc sistemas contínuos são

- O movimento de um fluido sendo conduzido cm um duto;
- O vôo dc um avião;
- O movimento de um foguete espacial cm órbita da terra;
- O comportamento dc um circuito elétrico

Muitas das considerações feitas cm projeto e modelagem dc sistemas discretos também precisam ser consideradas na modelagem de sistemas contínuos. Dc fato, 0 planejamento para o projeto dc simulação é aproximadamente idêntico, c existe uma necessidade similar para verificação e validação dos modelos

contínuos. As maiores **diferenças** entre os dois modos dc simulação são a forma na qual o modelo c considerado estável e as técnicas dc implementação utilizadas.

4.4.1 Modelos de sistemas contínuos

('orno já dito anteriormente um sistema contínuo normalmente é descrito cm lermos de equação diferencial ou por um conjunto delas. Alguns exemplos apresentados em [Círaybcal | são:

1) $\frac{dx}{} = kx$, que poderia representar o decaimento radioativo de uma determinada substancia, onde x representaria a quantidade dc substância que não sofreu decaimento c k alguma constante de proporcionalidade.

2) Um outro exemplo físico está ilustrado abaixo na figura 4.4.1.1

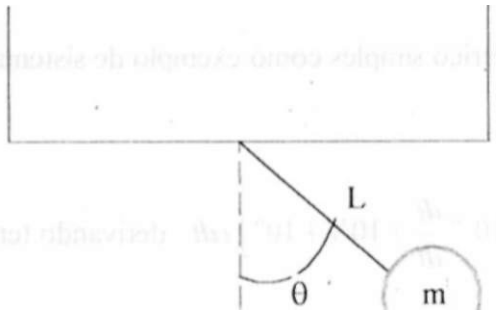


figura 4.4.1.1 - Pêndulo cm movimento.

Nesta figura temos a representação de um movimento realizado por um pêndulo dc massa m, **fixado** por um fio de comprimento L a uma superficie de sustentação. íissa massa é deslocada dc um ângulo 0 c

nesse ponto e solto de modo a movimentar-se livremente, considerando-se g como a força da gravidade a equação que descreve esse movimento c

$$d^2 0 \quad V \quad , \quad .$$

3) O circuito mostrado na figura 4.4.1.2 abaixo teria seu comportamento descrito pela seguinte equação:

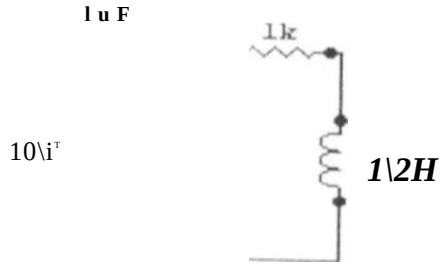


Figura 4.4.1.2 - Circuito elétrico simples como exemplo de sistema contínuo.

/ -~ I *Ri* **i ' f/íll** *e* > I 0 ⁶ ~ i IO⁵/ l 10" f/c// derivando temos

« ' /. 6// /,(F, 6// 6// í// <//

4.4.2 Processamento analógico

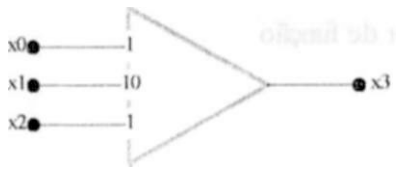
[Ourmaev] registra que os primeiros testemunhos da utilização de procedimentos analógicos paia o cálculo remontam a época dos estados escravagistas da Mesopotâmia. Os **Babilónicos** usavam na agricultura instrumentos analógicos baseados nas propriedades do triângulo rctângulo, **posteriormente** demonstradas por Pitágoras. Os gregos construíram um planetário munido de instrumentos analógicos inspirados no modelo do sistema solar instituído por **Claudius** Ptolomeu.

De acordo com Vichnevetsky a idéia de resolver equações diferenciais utilizando o computador foi introduzida inicialmente por Lord Kelvin cm 1876, essa idéia foi dc fato concretizada cm 1930 no M1 ***i*** quando Bush desenvolveu um analisador diferencial mecânico c durante a segunda grande guerra Lovell c Philbrick desenvolveram um analisador diferencial eletrônico.

O isomorfismo matemático dos diversos sistemas físicos permite estudar certos sistemas com a ajuda de outros. É muito importante que o modelo matemático represente bem o sistema original dc forma que as observações c conclusões extraídas do modelo permitam caracterizar qualitativa e **quantitativamente** o sistema real. O funcionamento das máquinas analógicas é fundamentado no princípio do isomorfismo matemático.

Dc acordo com ((iraybcal] um computador analógico simples c constituído de cinco elementos básicos: somador, atenuador, multiplicador, gerador de função c integrador. Os seguintes símbolos são utilizados para representar estes elementos:

Somador

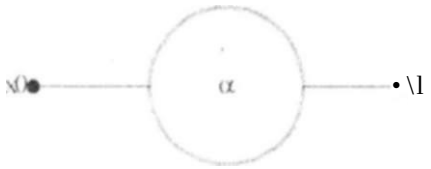


Operação:

x3 xOi 10xl+x2

figura 4.4.2.1 - Circuito somador.

Atcniiador



()pcração: \1 u*A)*

Figura 4.4.2.2- Circuito atenuador.

Multiplicador

Operação: x2 \0..\l

Figura 4.4.2.3 - Circuito multiplicador.

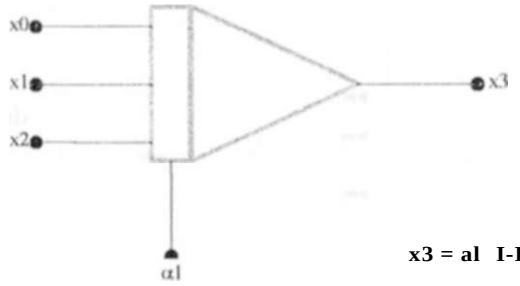
Gerador de função

\

Operação: xl f(>í))

Figura 4.4.2.4 - Circuito gerador dc função

Integrador



Operação:

x3 = al I-Integral(?0+xl+x2)dl

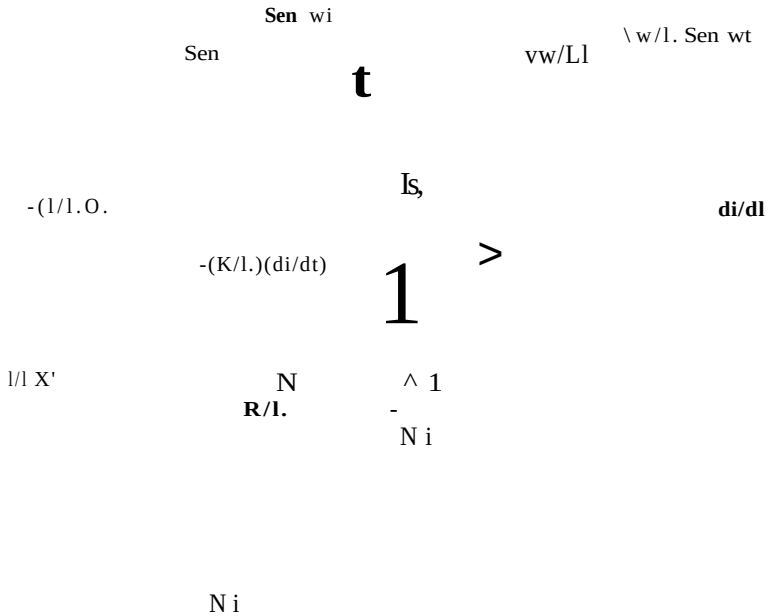
Figura 4.4.2.5 - Circuito integrador.

No processamento em máquinas analógicas a resolução dos problemas é realizada pela combinação destes elementos dc forma a compor a equação ou sistema dc equações, na saida desta combinação de circuitos, como exemplo apresentaremos uma possível solução para o problema do circuito da figura 4.4.1.2, para isso vamos relembrar sua equação básica:

$\frac{d^2i}{dl} + \frac{R}{l}\frac{di}{dl} + \frac{1}{LC}i = \frac{1}{l}\frac{de}{dl}$, considere que $c(t) = v\cos wt$, entao $\frac{de}{dl} = -vw\,\mathrm{Sen}\,wt$, e dai temos:

$L\frac{di}{dl} + R\frac{d/}{dl} + \frac{i}{C} = -v.\ w.Seníwl)$, que pode ser reescrita como:

$\frac{dr}{dr} \quad \frac{R}{l}\frac{di}{dl} + \frac{1}{LC}/ + \frac{VW}{I,}\,Seniwl)$, c então sua representação em lermos dc circuito analógico está mostrada na figura 4.4.2.6 abaixo.



'igura 4.4.2.6 - Exemplo de aplicação

4.4. Sinuilação digital de sistemas contínuos

Existe um enorme interesse em modelagem c simulação dc sistemas contínuos cm computadores digi lis, esse interesse é explicável por dois motivos: a grande difusão dos computadores digitais e o fato de i nputadores analógicos não serem dc grande disponibilidade. Os computadores digitais são capazes de i alizar operações lógicas c aritméticas cm alta velocidade, armazenar uma grande quantidade de dados mantendo um alto grau de confiabilidade, requisitos altamente desejáveis cm uma máquina para simulação.

Relembremos que os modelos matemáticos dc sistemas contínuos geralmente envolvem uma ou mais cqu. ções diferenciais, para que estes possam ser simulados, cm geral, precisamos encontrar solução para esta equações Em uma simulação cm computador analógico, tensões representam valores, os valores (ten õcs) são operados por amplificadores dc corrente de alto ganho e arranjos de circuitos como já visto

anteriormente, quando estas tensões são passadas através, por exemplo, de um integrador, o valor na saida representa a integral em relação ao tempo da tensão de entrada.

Os computadores digitais por sua vez realizam a mesma operação através de processos de aproximação numérica. Sempre que temos solução de equações diferenciais envolvidas no problema, as simulações de sistemas contínuos cm computadores digitais tornam-se lentas por causa do processo interativo necessário para resolver tais equações, de acordo com [Graybcal| essa diferença de velocidade entre computadores digitais e analógicos para solução de equações diferenciais, cm problemas pequenos, é de um fator dc 100. Em computadores analógicos a velocidade é independente do tamanho do problema, ao contrário dos computadores digitais.

Então a solução dc equações diferenciais cm computadores digitais é obtida pelo uso intensivo de métodos numéricos. Deve-se pois procurar os métodos que minimizem o tempo de resolução (em geral aqueles que convergem com o menor número dc iterações) de forma a diminuir o custo dc simulação mantendo um nível adequado de confiabilidade na solução do problema.

A maior desvantagem do uso de computador digital cm simulação dc sistemas contínuos é sua relativa lentidão quando comparado ao computador analógico. Por outro lado, o computador digital oferece um grande número dc vantagens, entre elas podemos citar exatidão, grande capacidade dc armazenamento dc dados, facilidade de debugging, segurança e sua natureza dc propósito geral, isto significa que o computador digital tem várias aplicações inclusive simulação. O computador,analógico é uma máquina dc propósito especial, capaz dc realizar eficientemente as tarefas para as quais foi projetado.

Uma abordagem que tenta combinar as vantagens de ambos, computador digital c analógico, deu origem aos chamados computadores híbridos, uma abordagem aposta na implementação dc máquinas de arquiteturas paralelas dc modo a compensar a necessidade dc grande volume de cálculos exigido pelos computadores digitais.

4.4.4 ypjjgagens para simulação dc .sistemas contínuos

Em simulação dc sistemas contínuos existem várias funções comuns a serem realizadas, tais como. representação do tempo, manipulação dc atrasos, entrada dc condições iniciais, formatação c impressão dos dados dc saída c cálculo dc numerosas funções.

Linguagens dc propósito especial para simulação dc sistemas contínuos foram desenvolvidas pelas mesmas razões que as linguagens dc propósito especial para simulação dc sistemas discretos. Existem dois tipos dc linguagens desta categoria

- Linguagens que utilizam bloco
- Linguagens baseadas em expressões da linguagem comum

Nas linguagens que utilizam formas de bloco, primeiro constroi-se um diagrama dc blocos similar aqueles usados para representação de computador analógico, e então o modelo c implementado através de um conjunto dc declarações dc conexão.

No segundo tipo o modelo é implementado em forma de equações de maneira similar a uma linguagem procedural qualquer.

As linguagens mais largamente aceitas tem sido aquelas baseadas em FORTRAN. Alguns exemplos são: **CSMP** III (**Continuous** system modcl program version III) c DYNAMO (Dynamic model) esta última desenvolvida no MIT.

4.5 I xcmplos de utilização prática da simulação

Nessa seção tentaremos mostrar a importância que a simulação tem adquirido nos últimos tempos e alguns dos campos onde ela tem sido ou estudada ou aplicada com sucesso

4.5.1 Simulação g mminiatura

Lm [Lilegdon] c apresentado um pacote dc software para simulação de processos industriais. FACTOR/AIM c um pacote para modelagem e simulação dc sistemas discretos de **manufatura.**

Est€ software permite a construção, animação c simulação dc modelos industriais permitindo o cru/amento dc alternativas de projeto tanto estatística quanto graficamente. **Utiliza** uma base de dados SQI. para armazenar os valores dc entrada, os modelos e os valores dc saída. Possui uma linguagem para modelar as interações entre processos c pode ser usada para projeto c planejamento e para operação e ' encadeamento ***(schedulling)*** das facilidades dc manufatura.

Modelos desenvolvidos com este software são compostos dc componentes, estes componentes tem características, algumas descrevem suas operações e outras seu ***status*** (estatístico c gráfico). Duas classes dc componentes estão disponíveis:

- Componentes com ambas as características: operacional c gráfica, por exemplo:
- Partes ***(loads)***
- Lotes (grupos de partes)
- Recursos (máquinas, operadores, acessórios)
- Pools (tilas ou ***buffers)***
- Materiais
- Transportadores
- Componentes com característica operacional somente, por exemplo:
- Calendário
- Planejamento de processo (conjunto dc tarefas)
- Ordens
- **Grupos** dc recursos de parada e manutenção

- Turnos

- Regras dc decisão

- Variáveis

A partir da selaçâo de um componente para fazer parte do modelo, mensagens que agem sobre ele estão disponíveis para mudar seu status operacional, suas características gráficas, etc. Mensagens incluídas no pacote são

- Animação (*draw, crase, refresh)*

- Simulação (mudança dc *status* ou característica do componente)

- Seleção do componente

- Manipulação do componente (*highlighí, drag, edit)*

- Mensagens clipboard (*cuí, copy, paste, clear, undo)*

- Database (*save, load)*

- Informação (descrição dc rede e mensagem dc *help)*

Para maiores detalhes ver |Lilcgdon|. Em muitas outras aplicações de manufatura tem sido constantemente utilizada a simulação, ainda nesta seção falaremos dc aprendizado dc máquinas, e deste modo estaremos dando mais ênfase ajunção da simulação com inteligência artificial, entretanto o exemplo do uso dc simulação no aprendizado dc máquinas será aqui utilizado como mais um exemplo de shhulaçào na manufatura

Um outro exemplo c apresentado em |Lu at. all| que aplica técnicas dc aprendizagem para regulagcm c operação dc máquinas que moldam peças (torno) dc forma a aumentar a eficiência destas utilizando uma abordagem aprendizado/estatística

4.5.2 Simtilação no aprendizado dc máquinas

Hm **f** Spclt **I** é apresentado um esquema para aprendizagem de um robô teslado no CESAR ((*'enter for Imgineering Systems Advanced Research*), I IHRMITvS-IlB (*Hoslile llnvironment Rohotic Machine Intelli^ence Kxperiment Series* I1B). liste robô é uma plataforma móvel que se desloca no laboratório sob controle dc dois sistemas de computador nele instalados: um subsistema VMI{ contém um 68020 com 1M dc RAM, *framegrabher* e um *framestore* para visão, c dispositivos de l/O para comunicação com os eíTctors c sensores do robô; um IBM 7532 (PC-AT) com 40M de hard disk, l,2M dc floppy, 2M dc RAM, 4 slots dc expansão do AT com 16 nodos NCUBIÍ, computador paralelo hipercubo dc propósito geral usado para o processamento da visão e para rodar o sistema expert que controla a navegação c aprendizado do robô.

0 robô é constituído basicamente de um par de braços, uma cabeça móvel que contem câmaras dc vídeo e um sonar, c um conjunto de locomoção. O objetivo é fazer o robô se deslocar no laboratório dc forma a encontrar o painel de controle c atuar sobre cie, ou modificando-o ou desligando-o.

1 Spclt **I** justifica que a escolha do aprendizado do robô cm um painel genérico foi feita para ilustrar o potencial c versatilidade do sistema *expert* que controla o robô. A meta principal é ter IIIRMIKS fazendo diagnose automática c reparos cm uma grande variedade dc plantas com componentes similares (por exemplo, controle de processo com válvula, ajuste dc nível em medidores, etc). A idéia é mostrar quc.o robô não colocará cm situação dc perigo uma instalação cm função dc trabalhar por tentativa-e-erro com a mudança no painel dc controle. O uso deste painel dc controle é meramente uma conveniência para leste do sistema geral dc aprendizagem que pode ser aplicado cm uma variedade dc situações não críticas no mundo real.

I Spclt] descreve o experimento c como resultado apresenta um painel onde um gráfico dc barras comparativo entre a performance (medida em número médio de erros cometidos) dc IlERMIES c seres humanos é comparada, ficando claro que o desempenho de HERMIRS é bem melhor.

4.5.3 Simulação c inteligência artificial

Essa c uma combinação dc técnicas que, tudo indica, sofrerá um forte desenvolvimento nos anos próximos, isso porque as técnicas de simulação têm sido largamente empregadas cm treinamento e no processo de reconhecimento de padrões c aprendizagem dc máquinas. Em |Orcn| discute-se a questão da garantia de qualidade de sistemas de alta precisão, onde as técnicas de simulação devem ser empregadas com margens dc erro desprezíveis sob pena da ocorrência de acidentes fatais. |Oren| agrupa as técnicas de modelagem c simulação em duas grandes categorias: modelagem e simulação tradicionais e modelagem c simulação cognitivas.

Simulação cognitiva c a simulação de sistemas que têm habilidades cognitivas tais como: percepção, interpretação, raciocínio, explicação ou aprendizado. A simulação qualitativa já é aceita como técnica na comunidade de inteligência artificial.

Em inteligência artificial um sistema cognitivo c um computador ou um sistema dc rede de computadores que tem habilidades cognitivas tais como: (I) capacidade dc processamento dc conhecimento incluindo aquisição, análise, transformação, geração c disseminação de conhecimento; (2) perguntar e responder questões em uma linguagem de computador ou linguagem natural; (3) monitorar ele mesmo, seu ambiente e seus usuários

A capacidade de processamento do conhecimento é uma meta cm IA. Adaptação para aumentar a habilidade de processamento é realizada através do aprendizado. Existem algumas técnicas e possibilidades para o aprendizado de máquinas, dois importantes grupos dc possibilidades podem ser explorados

- Sistemas com inteligência baseada no conhecimento que têm capacidade para realizar estudos de simulação podem definir cenários com os quais eles podem simular um sistema incrementando seu conhecimento sobre:

I) comportamento do modelo

2) sensibilidade do comportamento do modelo cm relação a variações em seus parâmetros ou condições dc operação.

- Baseado cm dados reais, sistemas inteligentes baseados em conhecimento podem hipotetizar modelos c testar sua validade via simulação. Automaticamente os modelos validados podem ser adicionados a base dc conhecimento do sistema aprendi/., ver [Oren & Zeigler].

lintão a possibilidade dc usar simulação como forma dc ensinar máquinas, através do aprendizado dc sistemas *experí* coloca essa técnica como uma importante aliada do desenvolvimento da inteligência artificial.

4.6 Conclusões

Por tudo que foi dito anteriormente e pelos exemplos aqui expostos, verificamos que a simulação tem sido largamente utilizada nos mais diversos contextos, ou para conseguir um aumento dc eficiência dos sistemas simulados ou para adquirir um *background cm* sistemas que ainda nem existem na prática. Muitos outros exemplos poderiam ler sido apresentados aqui, tais como: simulação no projeto dc redes de comunicação dc dados, ver **[Remes],** simulação dc redes dc comutação dc pacotes, ver [Gicsslcr|, simulação dc protocolo dc comunicação, ver |Wolfinger|, simulação em **teste** dc circuitos integrados, ver **[Ohletz],** simulação cm finanças, ver [Sviokla], para um exemplo didático sobre o assunto ver |Salib|, etc. Concluímos que as vantagens da simulação apresentadas na seção 4.1 realmente existem. E agora podemos verificar o quanto são verdadeiras as desvantagens apontadas na mesma seção.

a) Custo.

Com as ferramentas de software mais c mais amigáveis c dc fácil utilização, o preço dos computadores cm queda c o enorme aumento de performance destes, a tendência natural e que o custo envolvido na simulação venha a diminuir a cada ano.

b) Questões intrínsecas ao problema c ao modelo.

A abordagem sistemática que é utilizada permite a divisão de um sistema de grande porte em vários subsistemas menos complexos, este falo aliado ao fato de que a portabilidade dos softwares atuais permite **0** desenvolvimento de programas modulares, permite que problemas de grande porte ou dc grande complexidade possam ser divididos em programas menores c depois juntados dc forma a representar o sistema todo (tal divisão c o correspondente conjunto dc variáveis que fazem a interface entre os subsistemas c dc fácil c clara determinação *a priori),*

c) Suposições errôneas implícitas no modelo.

Nos parece que, qualquer que seja a ferramenta de pesquisa operacional utilizada, o fato de existirem suposições a respeito do sistema que não correspondem a realidade levará o modelo a divergir da realidade, sendo portanto uma questão que está relacionada com técnicas de modelagem e não a ferramenta dc implementação do modelo resultante. O uso da simulação aliado a estatística nos parece uma poderosa ferramenta de pesquisa operacional.

I Naylor| c |Nayior ct. al.| tratam da simulação cm economia em geral c cm administração de negócios em particular. |NayIor ct. al.| afirma que com o advento dos computadores, a possibilidade de experimentar e verificar diferentes hipóteses concernentes ao comportamento dos sistemas econômicos eslendeu-sc consideravelmente. A principal vantagem de se utilizar simulação em computadores como instrumento de análise econômica é provar não somente um procedimento para a formulação de teorias econômicas, mas também testar essas teorias.

Capítulo 5

Métodos de Monte Carlo

"Thus posed
by a single question the dice promoted:
"Is it possible that these two stones
are loaded? "
To compute the answer in histogram form,
he quickly selected ten trials
as his norm.
Thus proved
his point by hypole sis, you see:
That the answer is definitely "Yes,
maybe."

CAPÍTULO S

Métodos de Moule Carlo

5.1 Introdução

Métodos de solução de problemas que empregam quantidades aleatórias são classificados com a denominação geral de métodos de Monte Carlo. A denominação métodos de Monte Carlo designa uma família dc algoritmos numéricos baseados cm métodos probabilísticos para a resolução de problemas determinísticos tais como: cálculo de integrais, determinação dc máximos e mínimos, solução de sistemas dc equações, dc equações diferenciais, de equações a derivadas parciais, dc equações integrais, etc Para ser mais preciso, os métodos de Monte Carlo abrangem uma coleção de técnicas que permitem obter soluções, por meio da repetição de experimentos, de problemas físicos, matemáticos, dc engenharia, etc.

A denominação Monte Carlo foi utilizada pela primeira vez com o sentido dc designar um método para solução dc problemas numéricos que não admitiam serem resolvidos pêlos métodos até então conhecidos, em um artigo dc Mctropolis c Ulam, porém estes creditaram o nome ao matemático John von Neumann.

A idéia dc utilizar o conhecimento humano e máquinas para simular processos aleatórios é da época do naturalista e escritor francês conde de Buffon (Georges Louis Leclere 1707 - 1788 **)** que criou um método para estimar o valor do número *k.* O procedimento criado por ButTon consiste em desenhar duas retas paralelas separadas por uma distância D e em seguida utilizando uma agulha de tamanho L **(1,** < D), iniciar uma seqüência de jogadas com essa agulha anotando a quantidade total de jogadas c a quantidade dc jogadas cm que a agulha cortou uma das retas, sabemos que a probabilidade dc ocorrência do evento é dada por : $P = \frac{2L}{n\ D}$, a partir desse tato consegue-se um estimador dc *k* bastante preciso. Esse

Capítulo 1

Introdução

"Não adorno esta obra com frases eloquentes, nem com palavras pomposas, nem com esses primores de estilo que muitos empregam para valorizar os seus escritos, pois desejei que, ou não lenha qualquer mérito, ou a tornem grata a gravidade do assunto e a verdade das observações. Também não desejo que se julgue presunção, em homem de humilde qualidade, atrever-se a dar regras de conduta aos príncipes que governam os povos... "

Maquiavel
Dedicatória de 'O príncipe'

experimento, conhecido como a agulha de Buffon pode ser considerado como um precursor dos métodos dc Monte Carlo.

Os métodos de Monte Carlo foram introduzidos durante os últimos anos da segunda grande guerra por von Neumann c Ulam no estudo do comportamento dos ncutrons. As equações diferenciais que modelavam esse comportamento eram dc tratamento analítico extremamente árido ou mesmo inviável, essas pesquisas aconteceram no laboratório dc Los Alamos c delas originou-se a primeira bomba atômica.

Os métodos dc Monte Carlo foram desenvolvidos não só para simular a maioria das distribuições de probabilidades conhecidas mas também para as distribuições empíricas. Sua concepção e implementação é dc uma simplicidade conceituai muito grande e de uma aplicabilidade ainda maior.

Usualmente a solução de problemas é realizada através de uma seqüência de operações (algoritmo) que produz, a quantidade desejada f ou exatamente ou dentro de uma precisão especificada. Sc denotarmos por f|, f_2, ... ,f_n, ... os resultados correspondentes das operações realizadas seqüencialmente, então

-5.1 ' / li **111 /.,**

E no caso do número de operações ser finito, o processo acaba cm algum estágio. Neste caso o procedimento é estritamente determinístico c na ausência dc erros, duas pessoas diferentes reali-zando os cálculos obteriam o mesmo resultado.

Contudo existem problemas para os quais é praticamente impossível construir um algoritmo para sua solução ou o algoritmo é proibitivamente complicado. Lm tais casos, freqüentemente usa-se como recurso para a modelagem dc problemas matemáticos, físicos e de engenharia a lei dos grandes números da teoria da probabilidade. Os estimadores f i , f>, ... X, ... das quantidades desejadas f são obtidas via um tratamento estatístico do material envolvendo os resultados dc certos experimentos aleatórios repetidos muitas vezes. L exigida então a existência de convergência dc f„ para f (em probabilidade) com n > oo, isto é, V e > 0 deveremos ter

Bq. 5.1.2) j u i / ' f l / $f_{\prime\prime}\backslash < c$) I , onde P denota a probabilidade apropriada.

Suponha que existem duas variáveis, x e y, que estão distribuídas com **função** densidade de probabilidade f(x) c g(y) c m O < x < o o e O < y < o o . Queremos obter a densidade dc probabilidade, h(u), de $// = x + y$. A forma direta de fazer isso é por meio da transformada dc Laplace:

/ > H > - " / (* > *

Desse modo h*(z) é a transformada dc Laplace dc h(u) c, teoricamente, pode ser invertida para obter h(u), o problema c que freqüentemente obter a inversa dc h*(z) é extremamente difícil ou mesmo impossível, isto no entanto não inviabiliza o procedimento pois podemos calcular os momentos m_i da distribuição de u a partir de h*(z) através da sua expansão em série de Taylor-Mclaurin e a partir daí aproximar a distribuição dc u através dos seus momentos a uma distribuição familiar c tabelada.

0 procedimento anterior nem sempre é aceitável, e daí uma outra forma de obtermos h(u) é por meio de técnicas dc amostragem aleatória ou métodos dc Monte Carlo.

Para a programação de uma simulação de Monte Carlo em computador os seguintes requisitos são necessários:

a) Uma forma de gerar números aleatórios;

b) Uma forma de produzir amostras a partir de uma distribuição especificada;

c) Técnicas eficientes dc programação. »

[Naylor et. al.| aponta dois diferentes tipos de problemas para os quais é possível utilizar essa técnica para a busca dc uma solução:

I) Os problemas que envolvem alguma forma de processo estocástico (demanda dc consumidores cm planejamento dc vendas c programação da produção, tempo dc produção cm programação dc máquinas e da produção, total investido pela companhia em planejamento estratégico).

2) Certos problemas determinísticos que, ou não admitem uma solução analítico-dctcrninística ou apresentam um nível de dificuldade em sua solução que inviabilizam a utilização dc técnicas analítico-determinísticas para a busca de uma solução (solução dc equações diferenças dc ordem superior - acima de segunda ordem - c problemas de integrais múltiplas).

De acordo com 1**1** lanimersley & Handscomb| Termi, von Neumann e Ulam perceberam a possibilidade dc aplicação dos métodos dc Monte Carlo a problemas determinísticos c populari/.ou-os nos anos posteriores a II guerra mundial. Por volta dc 1948 Fermi, Metropolis e Ulam obtiveram estimadores dos autovalores da equação de Schrödinger usando os métodos dc Monte Cario.

Nesse capítulo vamos procurar mostrar as nuanças destes métodos c suas aplicações.

5.2 [Números aleatórios

Em matemática c estatística existe um conceito muito bem definido e claro, que é o de processo aleatório; entretanto a idéia dc um número aleatório não é tão clara. A idéia dc um número aleatório deriva dc processo aleatório, a questão é saber quando uma seqüência dc números é realmente aleatória pois a princípio a estrutura que a gerou e perfeitamente conhecida.

Um número aleatório entre 0 c 9 significa um dígito retirado desta população, tendo cada dígito a mesma probabilidade de ser escolhido (é como sc cm uma urna tivéssemos bolas numeradas de 0 a 9 e retirássemos uma ao acaso). Agora um número aleatório entre 0 c 1 é outra coisa.

Existem muitos problemas nos quais o elemento aleatório está presente. Por exemplo, se queremos estudar o efeito do número dc atendentes sobre a qualidade dc serviço dc uma agência dos correios

precisamos de um modelo para o padrão de chegada dos usuários na agência. **0** ruído aleatório é outro exemplo. Muitos tipos dc falhas em sistemas de comunicação são muito melhor modelados com efeitos aleatórios.

Outro problema onde a amostra aleatória torna-se muito útil é na avaliação dc volumes de regiões complicadas do espaço euclidiano dc mais de quatro dimensões. Este é um problema de estimação de integrais, claro. Suponha que R é um subconjunto do cubo unitário K em N dimensões. A ideia dc Monte **Carlo** é selecionar um grande número de pontos N em K, aleatórios, com probabilidades iguais dc serem selecionados estando em qualquer parte dc K. **Então** um contador dc números, por exemplo M, que estão em R, isto c, que satislã/.cm as condições dc desigualdade definindo R é ativado, M/N é um estimador do volume dc R. Para quem está familiarizado com a distribuição binomial irá compreender que o estimador da variância dc M/N c grande, assim a precisão do estimador é ruim. **Todavia** uma amostra dc 10.000 pontos iria produzir uma precisão bem melhor do que uma amostra dc 100 pontos, lista precisão é freqüentemente suficiente, sendo muito difícil obter melhor com outros métodos.

Ilistoricamente o primeiro método aritmético para gerar números pseudo-aleatórios cm computador digital foi o ***midsquare,*** no qual cada número da seqüência é obtido tomando-se a metade dos algarismos do quadrado do número precedente, liste método foi proposto por John von Neumann c Mctropolis cm 1946. Contudo , o método do ***midsquare*** foi abandonado c substituído pêlos métodos dc congruência desenvolvidos por Lehmer. |Naylor ct. al.| e |Graybeal| apontam alguns critérios pêlos quais um método de geração dc números aleatórios seria considerado aceitável:

1) Seqüência dc números que seja uniformemente distribuídos;

2) Estatisticamente independentes;

3) Reprodutíveis;

4) Não repetitivos cm qualquer extensão considerada;

5) Gerar os números cm alta velocidade;

6) Utilizar o mínimo dc memória;

Para qualquer destas aplicações cm um computador c preciso produzir uma grande seqüência dc números que se comporte como uma amostra aleatória a partir de uma dada distribuição. Dc acordo com [20fernando] existem três classes dc métodos para produzir tais números aleatórios:

1) Processo físico aleatório - por exemplo, interpretar a emissão de elétrons cm um cátodo aquecido como uma fase em algum ciclo de relógio, tais eleitos são fáceis de produzir, porém sua sincronização é muito difícil sem introduzir um viés. Dc qualquer forma o processo aleatório puro não pode ser reproduzido, é de implementação muito complicada c não permite ***debug,***

2) Gerar estes números ***off-íine*** e armazenar cm um arquivo cm disco - essa idéia foi muito difundida no período anterior a disseminação dos computadores quando inclusive eram publicados livros com estas tabelas de números aleatórios. Um problema que surge então é como acessar a tabela de fornia aleatória para evitar surgimento dc viés;

3) Usar um algoritmo implementado cm computador - estes estão disponíveis, normalmente são dc fácil implementação c podem ser reproduzidos pelo programa. Como estes números são gerados por um algoritmo eles não são completamente aleatórios e são conhecidos como pseudoaleatórios.

De acordo com |Naylor et. al.] existem quatro métodos para geração de números aleatórios

I) Métodos manuais - Mais simples porém menos práticos, são muito lentos para uso generalizado, entretanto oferecem grande interesse didático.

2) Tábuas de números aleatórios - evidentemente, utilizou-se um método dc geração para sua construção, tem a vantagem da reprodutibilidade mas é lento c apresenta o inconveniente dos números serem sempre os mesmos.

3) Método de computadores analógicos - dependem dc algum processo físico aleatório (por exemplo, o comportamento de uma corrente elétrica), são considerados geradores de verdadeiros números

aleatórios, são muito mais rápidos que os métodos manuais ou as tábuas, porém não admitem a reprodutibilidade.

4) Métodos de computadores digitais - são divididos em três classes: provisão externa, geração interna através de um processo físico aleatório e geração interna através de uma relação de recorrência.

O primeiro consiste no registro de tábuas de números em um meio magnético (fita, disco, etc.) para ser utilizado como a entrada em um computador digital c então utiliza-los como dados para o problema.

O segundo envolve a utilização de um dispositivo especial conectado a um computador digital, capaz de registrar os resultados de um processo aleatório qualquer c reduzi-lo a uma seqüência dc dígitos (decaimento radioativo, ruído térmico, etc). Não é reprodutível, exige um conjunto dc dados muito grande e isso influi diretamente na capacidade de memória do computador.

O terceiro envolve a geração dos chamados números pseudo-alcatórios através dc uma transformação dc um grupo dc números arbitrariamente escolhidos. Este método supera os problemas apresentados nos métodos anteriores, uma vez que não temos problema de entrada ou de capacidade de memória alem de admitir uma reprodutibilidade plena pois depende unicamente dc etapas aritméticas.

Nesse trabalho estaremos interessados apenas nesse último tipo dc método.

.

5.2.1 Geração de números aleatórios

Os métodos que serão aqui apresentados são os diretamente desenvolvidos por Lchmcr ou variantes destes, o leitor deve ter algum conhecimento básico da teoria dos números, se não o tiver sugerimos o apêndice A do capítulo 3 dc (Naylor ct. al.|.

0 processo dc geração dc números pseudo-aleatórios é determinístico porque os processos aritméticos envolvidos nos cálculos determinam univocamente cada termo na seqüência de números, ('orno os números da seqüência são calculados a partir dc uma expressão analítica, recursiva c determinística

conhecida, podemos calcular antecipadamente **o** valor do i-ésimo termo desta sequencia dc números (n_0, rii, ... , $n_{„}$, ...).

Então a seqüência **não** c aleatória, poderia-se argumentar, no entanto aceita-sc como aleatória a seqüência que satisfizer um certo número dc testes estatísticos (por exemplo, pode-se mostrar que sc os números de uma seqüência sc apresentam distribuídos uniformemente c são estatisticamente independentes, o processo pode ser considerado como aleatório mesmo que ele seja determinístico) |Naylor cl. al.|.

Os métodos dc congruência são baseados cm uma relação fundamental de congruência, (ver | Naylor ct. al. |, que pode ser expressa como.

Eq. 5.2**.1.1** tv, (aiii i c) MOD m,

onde lê-sc **n**„n congruente a e n„ a, c e m são inteiros não negativos

O significado da operação MOD (módulo):

a MOD b é o resto obtido com a divisão a/b

Por exemplo, 5 MOD 3 = 2, porque 3 divide 5 c resta 2. Para o algoritmo proposto na equação 5.2.1.1 suponha x_0 I, a I, c I, m 5, então teremos:

x, (1 i 1) MOD 5 2

x; (2 i l) MOI) 5 3

x» (3 i l) MOD 5 4

x_4 (4 I I) MOD 5 0

x_s (O U) MOD 5 I

x, (l i 1) MOD *5* *2*

Dados um valor inicial n_0, uma constante multiplicativa a, c uma constante aditiva c, a eq. 5.2.1.1 fornece uma relação dc congruência (módulo m) para qualquer valor de i, através da seqüência ; i i|, ri;, ... ,

n„ . . . } , alem disso, os termos subsequentes de **{ni}** determinados pela cq. **5.2**.1.1 são inteiros formando uma seqüência de resíduos de módulo m, isso implica cm **rij** < m, qualquer que seja **rij.**

Podemos então obter números racionais no intervalo (0,1) formando a seqüência **{fi)** **{tij**/m}.

Aqui então nos voltamos para uma questão de extrema importância que é relativa **a** existência de um período e seu tamanho, isto é, um valor positivo de i que seja o menor possível, i h, tal que nh =ⁱ n« onde h é o período, da seqüência **{nj}.**Suponha agora que esse h existe (e ele existe de fato e pode ser demonstrado como já dissemos anteriormente, através de alguns resultados da teoria dos números, cm particular ver teoremas **5** e 10 do apêndice A do capítulo **3** dc |Naylor et. al.|), então nosso problema passa a ser o seguinte: que condições devem ser impostas as constantes n_0, a, c c m de maneira tal que o valor dc h seja o maior possível?

Três métodos básicos foram desenvolvidos para a geração de números aleatórios, usando diferentes versões da eq. **5.2**.1.1. **O** objetivo comum a esses métodos é gerar seqüências com o maior período possível no menor espaço de tempo. São eles:

i) Método da congruência aditiva

ii) Método da congruência multiplicativa

iii) Método da congruência mista.

O primeiro envolve **K** valores itiiciais onde **K** é um inteiro positivo, c gera uma seqüência através da expressão:

Eq. **5.2**.1.**2** **fijn** - n; **h** n;-**k** (mod m)

lisse é o único método que produz períodos maiores que m.

O segundo gera uma seqüência **{rij}** de inteiros não negativos, todos menores que m através da relação:

Eq. **5.2**.1.**3** $n_{i(}$**i** an; (mod m)

O multiplicador a é escolhido para maximizar o tamanho do ciclo e o número m normalmente c o número de bils do tamanho da palavra do computador. A escolha dc a c m para produzir convenientemente longos ciclos não é trivial.

Esse método é um caso especial dc 5.2.1.1, com c = 0, c estatisticamente é muito bom.

O terceiro tem demonstrado algumas vantagens em relação ao multiplicativo no que tange a velocidade de computação e a ausência de periodicidade dos últimos dígitos, no entanto em alguns poucos casos é inaceitável conforme |Naylor et. al.|.

[Graybeal] apresenta o gerador congrucncial quadrático que pode ser utilizado quando m é potência dc 2, c equivalente ao *miüsquare* de dupla precisão com período longo. A relação recursiva que implementa este método e dada por:

$x_n < i \quad (x_{,,} (x_{n} \; i \; I)) \; MOD \; m$, $n > 0$, a semente $x_{()}$ deve satisfazer a relação $x_{0} \; MOD \; 4 \quad 2$

|DeArmon| propõe um método baseado na utilização dos bits menos significativos como números pseudo-aleatórios, apresenta testes c valida o método c cm [Lee] são apresentados os geradores de números pseudo-aleatórios Tausworthc c o ***shifl-register*** generalizado e em |Aluru, Prabhu & Gustafson] um método para geração de números aleatórios em computadores paralelos

5.2.2 Testes estatísticos dos números pseudo-aleatórios

Como já dito anteriormente, os números pseudo-aleatórios não são aleatórios na exata acepção do termo, no entanto, accita-sc uma seqüência dc números como aleatória quando estes são submetidos a um conjunto de testes e são aprovados como tal. Um teste de aleatoriedade é um algoritmo que computa uma estatística para uma seqüência dc números. Um conjunto de testes apresentados cm |Naylor et. al.| (e aqui apresentados de forma sintética) é constituído de (mas não necessariamente um conjunto de testes é composto dc todos os testes apontados abaixo):

i) leste de freqüência;

ii) 'leste dc série;

iii) leste do produto intervalado;

iv) **Teste** dc encadeamento;

v) leste do intervalo;

vi) Teste do máximo,

vii) leste do poker.

viii) Tcste dc Kolmogorov-Smirnov

ix) **Teste** da distância

0 ***teste de freqüência*** é utilizado para verificar a uniformidade dc uma seqüência dc p conjuntos consecutivos dc N números pscudo-alcatòrios. As duas equações abaixo definem a forma analítica do teste:

Eq. 5.2.2.1 $$\chi = \frac{Z}{N} \sum_{j=1}^{a} \left(f_j - \frac{N}{a} \right)^2$$

Rq. 5.2.2.2 $$= \frac{a}{N} \sum_{j=1}^{a} f_j^2 - N$$

0 procedimento é o seguinte: dividimos o intervalo unitário (0,1) cm a subintcrvalos iguais para cada conjunto dc N (m, ife, .., n) números pseudo-aleatórios. A quantidade esperada dc números aleatórios cm cada subintcrvalo é N/a. Agora seja i] (j = 1, 2, a) a quantidade real dc números n (i 1,2,..,, N) encontrada no subintcrvalo (j-l)/ a < r_i < j/a, então a estatística da eq. 5.2.2.1 tem aproximadamente uma distribuição qui-quadrado com a-1 graus dc liberdade (graus dc liberdade de qualquer estatística é igual ao número de observações independentes usado para calcular a estatística) para uma seqüência dc verdadeiros números aleatórios. Calcula-se essa estatística para todos os p conjuntos dc N números pseudo-aleatórios. Sc representarmos a quantidade dc valores p de Xi que estejam entre o (j-l)-ésimo c o

Capítulo 1

Introdução

1.1 Justificativa

O problema de programação não linear aparece nos mais diversos contextos (engenharia, economia, física, química, administração empresarial, agronomia, etc). O desenvolvimento de técnicas simples que propiciem sua resolução tem importância fundamental.

O desenvolvimento de métodos para otimização numérica de funções de várias variáveis é relativamente recente e data de meados da década de 40, quando surgiram os primeiros computadores eletrônicos.

O objetivo da teoria da otimização é, a partir de um problema expresso em termos matemáticos, encontrar a solução ótima dentre todas as possíveis soluções, quando estas existirem.

A otimização linear sofreu um impulso muito grande (resultado entre outras coisas, de um enorme esforço de guerra), tanto em termos de aplicação quanto em termos de algoritmos específicos, a partir do surgimento do método simplex (algoritmo do transporte, designação, etc). Entretanto, os métodos dc otimização não linear caracterizam-se, nesse contexto, por não possuírem um algoritmo geral, como o simplex na programação linear. Seus métodos são específicos e aplicáveis a determinados modelos matemáticos onde conceitos como convexidade, difcrcnciabilidadc, compacticidadc são vitais na busca dc uma solução. **Esses** algoritmos são, em geral, métodos numéricos interativos que procuram gerar uma sequencia dc soluções convergentes para a solução ótima.

O estudo desse problema permitiu o desenvolvimento de técnicas dc solução bastante sofisticadas ao mesmo tempo em que criou uma taxonomia própria. O problema de otimização é dividido, grosso modo, em otimização linear e não - linear, conforme a função objetivo e as restrições do problema. Também pode

j-ésimo subintervalo de uma distribuição qui-quadrado com ct-l graus dc liberdade (j I, 2, u) por Fj, computamos então a estatística explicitada na eq. 5.2.2.2.

Tendo sido computadas estas estatísticas, a hipótese de que os números pseudo-aleatórios na sequencia de a conjuntos sejam verdadeiros números aleatórios será rejeitada sc XF , com u-1 graus dc liberdade exceder o valor estabelecido pelo nível de significância desejado.

O *Teste de série* c usado para a verificação do grau dc alcatoriedade entre números sucessivos de uma seqüência.

l 5.2.2.3 $x\backslash \{-f\ ) \pm\pm\{/_{jk}$ A/ ;

Eq. 5.2.2.4 $\left| \begin{matrix} u \\ S \\ 1\ 1 \end{matrix} \right| \backslash \quad llJ$

Esse teste normalmente é aplicado a pares dc números. Esses números são considerados como coordenadas dc um ponto em um quadrado unitário dividido cm ***a***' células. Evidentemente essa idéia pode ser estendida a trincas dc números representando pontos em um cubo unitário.

0 teste consiste cm gerar uma seqüência dc p conjuntos dc N números sucessivamente e então calcularmos a estatística da eq. 5.2.2.1 para cada um dos p conjuntos. A partir desse ponto representamos por fj_k a quantidade dc números r, (i I , 2, N-l) que satisfaz a $(j-\backslash)/a < r, < j/a$ e (k-l)/ct < rm < k/a, onde j, k 1,2, a. Aí então calculamos a estatística dada pela eq. 5.2.2.3 acima para cada conjunto de N números. Aqui usa-se o resultado dc Good que mostrou que %/ - Xi distribuição aproximadamente qui-quadrado com a - a graus dc liberdade para uma verdadeira seqüência aleatória. Utilizamos este fato para calcular - Xi para cada conjunto p dc N números c designamos como s o número dos p valores resultantes dc x/ - Xi l estejam entre o (j-l)-ésimo e o j-ésimo subintervalo (j I, 2, u) dc uma distribuição qui-quadrado de ***a*** - a graus de liberdade, e então finalmente calculamos a estatística da cq. 5.2.2.4 acima, que tem u-l graus dc liberdade, aceitaremos a alcatoriedade da seqüência de números cm

um certo nível **de** significância se os valores dc **x?**' c Xs' não forem inconsistentes com a hipótese de que foram extraídos, aleatoriamente dc uma distribuição qui-quadrado com graus dc liberdade **apropriados** (esse teste pode ser estendido para triplas, quádruplas, etc).

0 *Teste do produto intervalado* é uma medida da independência dc números pseudo-alcatórios é conseguida através de um coeficiente de produto intervalado. Seja k o comprimento do intervalo, este coeficiente é definido como:

Um resultado importante aqui é que sc não existe correlação alguma entre r, c^k, os valores dc Cj, serão distribuídos aproximadamente dc forma normal com valor esperado 0,25 e desvio padrão igual a

\2(N —r—, para k > 0 de acordo com Naylorl.
k)

O *teste do encadeamento* serve para verificar a natureza aleatória oscilatória das seqüências dc números pseudo-aleatórios. **Existem** dois tipos diferentes dc testes: teste de encadeamento para cima c para baixo c acima c abaixo da média. No encadeamento para cima c para baixo para uma seqüência de N números n, i ; , r definimos uma seqüência S dc N-l bits binários, cujo o i-ésimo termo é zero sc r, < rj,| e é igual a um sc **n** > fri. Uma subseqüência de k zeros enquadrados pelo algarismo 1 em cada extremidade forma um encadeamento de zeros de comprimento k, da mesma forma compomos encadeamentos com o algarismo I

O teste consiste na contagem do número real dc encadeamentos dc diferentes comprimentos e posterior comparação com os valores teóricos correspondentes. Aqui novamente o teste qui-quadrado de capacidade dc adaptação pode ser utilizado para testar se um gerador de números pseudo-alcatórios é aceitável em um determinado nível dc significância (uma característica comum de seqüências não aleatórias é a presença de longos encadeamentos em excesso).

No encadeamento acima c abaixo da media para uma seqüência de N números pseudo-aleatórios ri, r,, ... , r definimos uma seqüência S com N bits binários, cujo o i-esimo termo c igual a zero sc < *Vz* e é igual a 1 se n > 1/2. O número esperado dc encadeamentos dc comprimento k é — *rr~* e O número total de encadeamentos é (N i l)/2, contamos os encadeamentos em S e um leste qui-quadrado pode ser utili/.ado para verificar se um dado gerador de números pseudo-aleatórios é aceitável.

O *Teste do intervalo* refere-se a aleatoriedade dos dígitos cm uma seqüência dc números, enquanto os anteriores eram relativos a aleatoriedade de seqüências dc números (cada número constituído dc um número fixo dc dígitos) Agora estamos interessados no comprimento do intervalo constituído por dígitos diferentes de "d", para um dado dígito "d". Quando k algarismos diferentes de "d" ocorrem entre dois "d", temos um intervalo dc comprimento k (dois "d" consecutivos produzem um intervalo dc comprimento nulo). Para uma seqüência verdadeiramente aleatória, a probabilidade de se obter um intervalo dc comprimento k c dada por:

Eq. 5.2.2.6 $P(k) - (0,9)^{k} (0,1)$

Para uma dada seqüência dc dígitos, são feitas contagens do número de intervalos que ocorre para cada comprimento, então usamos o teste qui-quadrado dc capacidade de ajustamento para comparar o valor esperado com o número real dc intervalos dc comprimento k (os valores qui-quadrados podem ser tratados de maneira semelhante as equações 5.2.2.1 e 5.2.2.4 para testar a hipótese de que os intervalos são aleatórios.

O *teste do máximo* é um simples teste dc freqüência que pode ser repelido com diversos conjuntos dc N números aleatórios. Para um conjunto de N números aleatórios uniformes independentes no intervalo unitário (0,1), podemos definir uma variável aleatória R max (n, r_2, ... , r_n) que tem uma distribuição dc probabilidades definida por uma estatística tal que R" seja uniformemente distribuída em (0,1).

O teste do máximo dc N números aleatórios uniformes c chamado dc leste da N-upla (n, r_2, ... , r_n) e c considerado um teste mais rigoroso que o teste básico de freqüência.

0 *feste do poker* **é** um teste especial de **freqüência** para combinações de **5** ou mais digilos de maneira aleatória. São contados parcs, trincas, quadras, etc, que são testadas em relação as frequências esperadas para essas ocorrências.

O *teste de Kolmogorov-Smirnov* c utilizado para verificar o grau dc aleatoriedade (a aderência da distribuição dos números observados com a distribuição teórica, nesse caso relativo a distribuição uniforme). Sua utilidade **é** testar a hipótese de que os números vieram dc uma distribuição uniforme.

0 *teste da distância* considera sucessivos pares dc números aleatórios como as coordenadas de pontos cm um quadrado unitário. Por exemplo, se a seqüência aleatória en,, n?, ... , n_i os pontos (ni,n?), (m.n.i), (n_r-i,n_i) será plotada em um plano x-y. O quadrado da distância Euclidiana *Ür* é então calculada entre cada par dc pontos. Sc os pontos são distribuídos aleatoriamente no quadrado unitário (se forem realmente aleatórios), a probabilidade que o valor observado de D'* ser menor ou igual a um valor k c

$$nk \quad \frac{8}{3}k^2 \setminus \frac{k^2}{2}, \text{ para } k < 1.0$$

$$\frac{1}{3}\, i(/r \quad 2)k \;\; i4(A \quad l)\, V - (A \quad l)^{*}$$

$$k \cdot \quad 4k \text{ are sec } Vk, \text{ para } 1.0 < k < 2.0$$

Usando esta função dc distribuição pode-sc calcular as freqüências teóricas para os valores dc I)' em qualquer intervalo preestabelecido; as comparações das freqüências atuais com as freqüências teóricas podem ser feitas com o uso da estatística qui-quadrado.

Um aspecto importante na seleção de testes estatísticos apropriados para os números pseudo-alcatórios c que esta seleção estará sempre limitada por um conjunto dc qualidades desejadas para um certo gerador c para uma determinada aplicação particular.

5.3 Integração por Monte Carlo

Iremos aqui analisar as circunstancias cm que o método de Monte Carlo é equivalente e as vc/.es superior aos métodos clássicos do cálculo numérico, líxistem dois esquemas básicos de integração por Monte Carlo: o método do sucesso ou fracasso e o método da média amostrai. Pelo lato do método do sucesso ou fracasso ter sido muito popular nos primórdios de utilização da integração por Monte Carlo c por sua ineficiência os métodos de Monte Carlo ficaram com uma imagem muito ruim c sendo considerados grosseiramente inferiores aos métodos clássicos.

Todavia devemos verificar que pode-sc melhorar o desempenho dc cada método introduzindo-sc melhorias no sentido de aumentar sua precisão. Hvidentcmentc tendo cm mente que o custo adicional necessário a implementação dessas modificações deve ser bastante considerado e sc justifica única c exclusivamente pela aplicação que sc vai dar ao resultado, por exemplo I lammcrslcy e Morton estudaram **0** uso dc variáveis correlacionadas na experiência de Buffon. Primeiro utilizaram duas agulhas fixadas cm forma de cruz e demonstraram que havia um ganho dc eficiência dc 12.2, isto é, para obter a mesma precisão, gastava-se 1/12 vezes menos tempo que com uma única agulha, fizeram a mesma experiência com três e com quatro agulhas fixadas em torno dc um mesmo ponto c formando ângulos iguais entre si c obtiveram ganhos dc 44**.3** c 107.2 respectivamente, isso por si só nos faz mudar a forma de ver os métodos de Monte Carlo.

Apresentaremos cada método separadamente e, após, faremos uma comparação entre cies e apresentaremos algumas técnicas para a melhoria da performance.

5.3.1 Método dg sucesso / fracasso

O problema geral é o de estimar a integral múltipla

Eq. 5.3.1.1 CD \(/>{x)d\>

onde a variável x representa uni vclor no espaço de coordenadas Çi, Ç^...,^ dv representa um volume elementar do espaço n-dimensional e a integral tem limites fixos c finitos (a suposição dc o limite dc integração ser finito não c, dc forma alguma, uma restrição, pois sempre podemos resolver essa questão com uma adequada transformação dc variáveis), desse modo assume-sc que <^> *é* finita em toda parte c limitado a essa região.

Consideremos aqui o caso bidimensional, seja a função g(x) a ser integrada no intervalo a < x < b c seja c max |g(x)} no mesmo intervalo. **Temos** então um retângulo de lados (b-a) e c. **que** contêm, e é idêntico, **ao** espaço dc probabilidades **Q,**

Q {(x,g(x)): a < x < b, **0** < g(x) < c }.

Sc fizermos então b - c = 1 e a - **0** teremos um quadrado unitário como mostrado abaixo

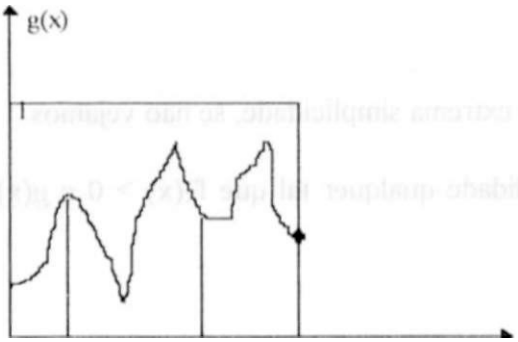


figura 6.2.1.1- Um exemplo dc função.

O conjunto dc todos os pontos (x,y) G *Q* tais que y < g(x) forma então a área sob a curva g(x) c então temos

^ í *tt(ˣ)dx* /, como a área de *Í1* c(b-a), temos que dado um ponto dc coordenadas (x,y) c Cl, a probabilidade dc (x,y) c A c

Eq. **5.3**.1.**2** $\int_{arcada\,A} f_x(x)\,Ux$, **arei/ Jcii ((/) í/j** C(/J *a)*

O método de Monte Carlo de sucesso ou fracasso se apoia nesse fato simples para gerar um estimador para I, a partir dc uma seqüência aleatória dc pares (x_J, v_J) tais que $y_y < g(\cdot)$, contabilizados como n,\. Então a probabilidade dc, cm n pontos, (xj,yj) e A c estimada como

Eq. **5.3.**1.**3** *p* , e então podemos estimar o valor de I como

Eq. **5.3**.1.4 / $\frac{c(b-a)}{n}$.

E fácil ver que n_A c o número de acertos da hipótese (x„Vj) Í A, e, evidentemente, nu n - n_A, é o número de pontos que não atende a hipótese, ou seja, o número de erros.

5.3.2 **Método da média amostrai**

Nesse caso, como no anterior, o raciocínio c dc extrema simplicidade, se não vejamos.

Seja f\(x) uma função densidade dc probabilidade qualquer tal que $f_x(x) > 0$ c g(x) / 0, sc X U(a,b), então

f\(x) l/(b-a) sc a < x < b e f\(x) 0 cm outros casos, então

Eq. **5.3.2**.1 / J ; / $\int J\backslash(x)dx$ *li* $\frac{(b-a)K[g(x)\backslash}{f_x(x)}$ $f_x(X)$, e daí se gerarmos uma seqüência X (xi, X2, x_n) dc variáveis aleatórias uniformemente distribuída no intervalo a x, b, teremos

I r-**i**

Eq. **5.3.2.2** /«/[^f**(jr)]** *zLg(x),* e então nosso estimador será dado por:

Eq. **5.3.2.3** / *(b a)* •

5.3.3 Uma compararão entre os dois métodos

Vamos então calcular o valor esperado c a variância dos estimadores dados pelas equações 5.3 1.4 t 5.3.2.3.

$$/,(/) = /J\ c\{b\text{-}a\} \quad \frac{n}{n} \quad \frac{c(b\text{-}a)}{n}$$

mas, como podemos observar, cada experimento constitui um experimento dc Bcrnoulli com probabilidade p, então

$K(n_A)$ np, e dai temos que

Eq: 5.3.3.1 $ic\{i)\text{-}c(b\ \ a)p=I,$

esse e o valor esperado para a equação 5.3.1.4. Agora vamos calcular o valor esperado de 5.3.2.3, como

$$b\text{-}a^{\wedge} \qquad b\text{-}a \qquad h \qquad b\text{-}a^{\wedge}\text{-}$$
$$11 \qquad '' \quad /1$$

logo,

Eq. 5.3.3.2 $/\bullet:[/] = (b\ \ a)/í[«M] = /\ .$

Vamos então, agora, calcular as variâncias das mesmas equações. Para a equação 5.3.1.4 temos:

$$Var\backslash I\backslash \qquad Var^{\wedge}c(b\text{-}a) \qquad \frac{c(b\ \ a)f}{n} \qquad \boldsymbol{Var\backslash n}$$

novamente usando o lalo dc que os experimentos são de Bernoulli, temos que

$Var|n_A|$ np(l-p), daí

Eq. 5.3.3.3 Kíir[/]: $\backslash c(b\ \ a)\ /\ ]$, consequenlemcnlc lemos

.

Eq. 5.3.3.4 $a[i]$ t M c f a 'j

Vamos então efetuar os mesmos cálculos agora para a equação 5.3.2.3.

Karl/ *Vat* *íL* *\2* *lai*
I' , I s //)

(f> a) **r** *I* **vi** *Varhix)]*

porém sabemos que

Var|g(x)| E[g²(x)] - E²[g(x)], temos então:

Eq. 5.3.3.5 Kar[/] = -[(A ***a)fg²(x)dx-T***

Eq. 5.3.3.6 *o* [/] ^ [(/ ; ^ (xh/.v /'

Vamos lazer uso da desigualdade de Chebyshev para que possamos fazer uma comparação entre estes dois métodos

Aplicando então a desigualdade de Chebyshev na equação 5.3.3.3 temos:

/ / > *Va,[l]* /[,(/, ₀) *l* **- *a***
ne

Então,

/'[(/ /[<*?]> a se ***a***<\ *l[c(b a) l*
*h i:*²

logo, para dados ***a*** c e, lemos:

n> ***í[c(b a) /}***
(1 ***alé*** , pelo teorema central do limite, c para n suficientemente

grande, temos

/' / / . o(/J , tem distribuição N((), I), dessa forma

/ / *k{* ' - *A i* * */'[k* (/'<*] C|)(A) (|)(*k)* I 2d)(A)
a(l)

c então o intervalo de confiança com nível l-2<I>(k) para I será:

/ i *kcr(í).*

Utilizando o mesmo procedimento com a equação 5.3.3.5, obtemos resultado idêntico ao anterior. Sejam *l* e 1, respectivamente os estimadores do método dc sucesso fracasso c do método da média amostrai, já vimos que /•,'[/,] /','[/,] / . Vamos então subtrair as equações 5.3.3.3 e 5.3.3.5 para obter

Var{l,)-Var{l,) $\frac{h \quad a}{ii}$ *cl* \ $g^{2}(x)dx$, como sabemos g(x) < c, Vx t [a,b],

então

Ktf/*]/,] > Kar[/,] , portanto considerando-se o mesmo tempo dc cpu utilizado em ambos os métodos, o método da média amostrai é mais eficiente.

.

5.4 Passeio aleatório

0 passeio aleatório está entre os construtos probabilísticos conhecidos como processos estocásticos. Um processo estocástico é uma família dc variáveis aleatórias indexadas por um conjunto dc números reais c satisfazendo certas condições de consistência.

|l'cllcr| suponho o seguinte jogo: o jogador ganha um dólar em caso dc vitória c perde um dólar cm caso dc derrota (experimentos dc licrnoulli com probabilidades p e q, respectivamente, onde q **I** - p), suponhe ainda que o montante total dc dinheiro em jogo é a, sendo que o jogador A tem z. dólares c o jogador **B** tem a - z dólares, o jogo seguirá seqüencialmente até o capital do jogador A ser dc a dólares ou dc 0 (zero) dólares, isto é, até que o jogador A arruine **B** ou vice-versa. **0** nosso interesse é relativo a probabilidade dc ruína do jogador e a distribuição dc probabilidade da duração do jogo.

1 Feller] propõe ainda que o passeio aleatório seja encarado como o resultado do movimento de uma partícula sobre o eixo dos x, essa analogia física tem a vantagem de proporcionar uma maior facilidade dc

ser colocada como otimização vinculada ou não - vinculada conforme apresente ou não restrições, respectivamente.

Uma taxonomia para otimização é apresentada cm (Shang & Wah| c c mostrada com algumas modificações na figura 1.1.1 abaixo.

Os métodos *covering* detectam sub-regiões onde o mínimo global não está e as excluem Em geral esta abordagem é útil para problemas que requerem soluções com precisão garantida. Estes métodos podem ser computacionalmente muito caros pois o tempo dc processamento aumenta dramaticamente com o tamanho do problema.

Os métodos *descenl* generalizados modificam a trajetória cm busca do mínimo. Sua maior desvantagem é o grande número dc avaliação dc funções gasto cm regiões não promissoras. O método dc penalidades previne múltiplas determinações dc um mesmo mínimo local pela modificação dá função objetivo, isto é, pela introdução dc um termo de penalidade em cada mínimo local para uma função auxiliar; o problema c que, como muitos mínimos locais são encontrados, a função auxiliar torna-sc muito solicitada (tornando o processo tedioso), e a função objetivo modificada torna-se mais difícil dc minimizar.

Os métodos ***clusíering*** realizam uma análise ***clustering*** para prevenir redeterminação dc um já conhecido mínimo local. Hxislem duas estratégias para o grupamento dos pontos em torno do mínimo local, reter somente pontos com valores relativamente baixos da função ou empurrar cada ponto para um mínimo local realizando uns poucos passos de uma busca local. Hstcs métodos não têm boa performance com uma função que apresente muita irregularidade, isto é, que apresente muitos mínimos locais

Os métodos de busca aleatória são simples e funcionam muito bem cm algumas aplicações, contudo, cm geral, tem baixa eficiência quando impropriamente aplicados.

Os métodos baseados em modelos estocásticos em sua maioria utilizam variáveis aleatórias para modelar os valores desconhecidos da função objetivo. Os métodos bayesianos, por exemplo, são baseados cm funções estocásticas c minimizam o desvio esperado do estimador do mínimo real global.

visualização, se não vejamos, suponha que os experimentos sejam realizados nos tempos i 1.2, no (empo I 0 esta partícula está na posição x z, e nos tempos t 1,2, 3,...ela se move em passos unitários para a esquerda c para a direita de acordo com o resultado dos experimentos (vitória/derrota, sucesso/fracasso), desta forma a posição da partícula no tempo t n representa o capital do jogador ao final do n-ésimo experimento. O jogo acaba quando a partícula alcança ou x 0 ou x a. Diz-se então que a partícula realizou um passeio aleatório. As posições x 0 e x a são denominadas barreiras absorventes; sc p q ***Vi*** o passeio aleatório é dito ser simétrico. Se, dc outro modo, considerarmos outras posições limites, como por exemplo, uma barreira de reflexão em x ----- ***V2*** com a propriedade de que se a partícula inicia em x I c move-se para a esquerda ela é rclletida cm x ***Vi*** c retorna para x 1 ao invés dc prosseguir até x 0, esta é a chamada barreira de retlexão. Uma outra barreira é a barreira elástica que são parcialmente absorvente e parcialmente reflectiva, |I'ellcr| trata essa questão com minúncia de detalhes.

5.4.1 A ruína do jogador

Como já visto anteriormente, a ruína do jogador ocorre quando um dos jogadores está com uma quantidade de a dólares c o outro com 0 (zero) dólares. O que nos interessa agora é saber a probabilidade de ocorrência desse evento.

|lcller| demonstra que p, 1 q, I, onde *p,* é a probabilidade dc vitória e q, é a probabilidade dc derrota, isto é, q, é a probabilidade da particula ser absorvida em x = 0 c p_x a probabilidade de absorção cm x a.

Depois da primeira jogada o capital do jogador é ou z - I ou z I I, c então temos

Eq. 5.4.1.1 q_z p $q_{/M}$ 1 qq_y.,

onde I • z • a-1. Para z I a primeira jogada pode leva-lo a ruína c temos

Eq. 5.4.1.2 q, pq_2 1 q

similarmente para /. a-l a primeira jogada pode leva-lo a vitória e assim

líq. 5.4.1.3 $q_{a}.|\ qq_{a}._{2}$

para unificarmos as equações 5.4. l i a 5.4.1.3 acima façamos

qo I e q., 0

A equação 5.4.1.1 c uma equação diferença, supondo p / q a equação diferença 5.4.1.1 admite duas soluções particulares: q_{z} = l e q_{z} - (q/p/, a solução da equação 5,4.1.1 c da forma

q, = A i B(q/p/

A i H I

resolvendo o sistema de equações lineares

temos.

Eq.. 5.4.1.4 $\boldsymbol{q_{2}}$ =

A equação 5.4.1.4 c a solução formal da equação diferença 5.4.1.1 satisfazendo as condições limites. Contudo devemos observar que uma suposição básica para obtermos a cq 5.4.1.4 acima foi que p / q. pois se p q a equação 5.4.1.4 liça sem sentido, neste caso as duas soluções particulares q, I e q, (q/p/ são idênticas, contudo existe uma segunda solução formal em q, = z, e desse modo

q„ A + Bz é solução dc 5.4.1.1.

Para satisfazer as condições limites devemos ter $[A$ i Ha 0, então

•q. 5.4.1.5 q_{z} = 1 - (z/a)

[Feller] demonstra também a unicidade dessa solução. Então a probabilidade de ruína do jogador se p / q é dada pela equação 5.4.1.4 e se p q *Vi* é dada pela equação 5.4.1.5.

5.4.2 **Durarão esperada do jogo**

[Feller] deduz, a distribuição de probabilidade da duração esperada do jogo supondo (c demonstrando) que esta esperança é finita, lúitão temos

Eq. 5.4.2.1 *I).* $z \quad a \quad \sqrt{\frac{P/}{}}$ sep / q.
$<t \quad P \quad <l \quad P$ i $(q \quad * \quad / P$

Eq. 5.4.2.2 *I) z(a z)* se p q.

[Feller] observa que se dois jogadores, cada um com 500 dólares jogam uma moeda ate um deles estai arruinado, a duração média do jogo é de 250.000 experimentos (jogadas). Sc um jogador tem um dólar e seu adversário 1.000 a duração media é dc 1.000 jogadas.

p	q	z.	a	Probabilidade dc ruína	Probabilidade de sucesso	Ganho esperado	Duração esperada
0.5	0.5	9	10	o.I	0.9	0	9
0.5	0.5	90	100	0.1	0.9	0	900
0.5	0.5	900	1000	0.1	0.9	0	90000

0.5	**0.5**	**950**	**1000**	**0.05**	**0.95**	**0**	**47500**
0.5	**0.5**	**8000**	**10000**	**0.2**	**0.8**	**0**	**16000000**
0.45	**0.55**	**9**	**10**	**0.210**	**0.790**	**-1.1**	**11**
0.45	**0.55**	**90**	**100**	**0.866**	**0.134**	**-76.6**	**765.6**
0.45	**0.55**	**99**	**100**	**0.182**	**0.818**	**-17.2**	**171.8**
0.40	**0.60**	**90**	**100**	**0.983**	**0.017**	**-88.3**	**441.3**
0.40	**0.60**	**99**	**100**	**0.333**	**0.667**	**-32.3**	**161.7**

Tabela 5.4.2.1 - Resultados obtidos por [Feller].

Se passarmos ao limite com a --> a> (significa jogar com um adversário infinitamente rico), para p q a duração tem esperança infinita. Sc q < p (jogo favorável) a probabilidade dc ruína c (q/p)'. Sc q>p a probabilidade dc ruína c um.

|l'cllcr| apresenta a tabela 5.4.2.1 acima para ilustrar alguns resultados curiosos.

5.4.3 Cadeias dc **Markov**

Seja li um conjunto enumerável ou finito, os elementos dc li serão chamados estados e li o espaço dc estados. Seja {**p,**},. i uma sucessão de números tais que p, > 0 e **V** p_i I .
Tm

Seja p uma matriz dc números p_{ij}, ij e E, tais que pij > 0 e Vie B $\sum_j P_{ij}$ I Unia tal matriz c chamada matriz, markoviana.

Por definição uma sucessão dc variáveis aleatórias $\{X_n\}_n$ \;r.>... c chamada uma cadeia dc Markov com distribuição inicial {pi},, b c matriz, de transição P se:

a) V i e li P|X„ i| p,

b) Vn; V $i_{()}i|$. $i_{„}i_{„}$ **i** í E lai que

HXo $i_{„},...,X_{n}$ $i_{„}|$ 0

/ , J ^ - /. . . . .

Significa que a distribuição da variável $X_{()}$ é determinada pela sucessão {p,};, \s e que além disso a distribuição condicional de X_{n} i dado XoX|....$X_{„}$, não depende nem da sucessão io.i,,, nem do instante n, só depende do estado i_{n} ocupado no tempo n (Vn).

O passeio aleatório tal como enunciado anteriormente é uma subclasse da cadeia de Markov. O problema da ruína do jogador pode ser modelado matematicamente como uma cadeia de Markov com estados 0, I, 2, a é a matriz, de transição:

1 0 0 . . . 0
q 0 ***p*** . . . 0

• • • <***i*** **0** ***p***
. . . . 0 0 I

O capital inicial z indica o estado do jogo, mais detalhes podem ser encontrados cm [Fernandez] e [Feller].

5.5 Métodos de redução da variância

lixiste uma diferença entre o problema usual de estimativas estatísticas e estimação por Monte Carlo Essa diferença consiste cm que no primeiro caso tanto a distribuição dc probabilidade como os parâmetros a serem estimados são considerados fixos (dada uma amostra dc n valores da distribuição podemos calcular o estimador dc variância mínima); no segundo caso a resposta é lixa c o problema é conseguir uma amostragem da distribuição que forneça um valor mínimo de variância. Além disso, uma diferença adicional é a que se encontra na direção da inferência, isto é, no método de Monte Carlo <\ inferência é

realizada das observações amostrais para o espaço dos parâmetros, enquanto no easo de estimativas padrão a inferência c feita do espaço de parâmetros para o espaço de amostras.

Uma forma dc melhorar a precisão dos resultados obtidos através do uso das técnicas de Monte Carlo é diminuindo a variância. Os métodos dc Monte Carlo têm grande aplicação c conseguiram um enorme desenvolvimento cm física da partícula, c é desse contexto que têm surgido algumas boas técnicas dc redução dc variância.

Colocaremos este problema da seguinte forma:

Seja S o espaço de amostras, onde:

S = **{s** *(r* **S: s** Fn **(si,** s?, ... , s„)}, é uma variável aleatória uniformemente distribuída com n lixo e finito, S então é o espaço dc todos os vetores dc dimensão n, cujos componentes são números aleatórios.

lodos os pontos **s** c **S** estão relacionados com uma função densidade dc probabilidade f**(s).** Suponha que cada componente do vetor dc números aleatórios é independente e identicamente distribuído na forma:

$$/(\bullet V,) \quad \begin{cases} 1 \text{ se } 0 < s, < 1 \\ 0 \text{ em outros casos} \end{cases}$$

A função densidade de probabilidade conjunta é dada por:

$$\begin{cases} 1 \text{ sc } 0 < s, < \backslash, \textit{para } i \quad 1,\ldots, n \\ \mathbf{0} \text{ cm outros casos} \end{cases}$$

Suponha agora que para qualquer que seja o vetor s e S um programa (função) o transforme em um valor escalar y c Y, então Y é uma variável aleatória (Y também poderia ser vetorial). **0** objetivo então é estimar o valor esperado dc Y obtendo um número tão próximo quanto possível do valor real, essa esperança é dada por:

$$l>:(Y) \quad \mathbf{I}_{\mathbf{II}}^{\mathbf{i}} \ldots \mathbf{J}_{o}^{\mathbf{i}} \; Y(s)f(s)ds$$

O procedimento aqui normalmente é gerar uma amostra de n vetores aleatórios da fdp f(s), de forma a calcular o valor de Y para cada um desses vetores e obter a media dessas n observações de Y como o estimador do valor esperado.

Em muito poucos casos conhecemos explicitamente Y, porém se temos certa liberdade ao especificar sua forma e se se reconhece e aproveita detalhes do problema, pode-se obter um rendimento melhor das técnicas de Monte Carlo. Nesse ponto é importante observar que a existência dessa liberdade adicional é uma das principais diferenças entre experimentos com sistemas lisicos c a experimentação com os modelos dc simulação associados a esses sistemas.

Uma condição importante é que cm todos os modelos dc simulação a correlação entre cada número aleatório utilizado na simulação e o resultado que provoque o evento no sistema simulado, gerado por esse número aleatório, seja positiva c tão grande quanto possível.

Então se se empregam números aleatórios para gerar tempos de serviço durante a e.ˇecução do programa, é exigido que os números aleatórios grandes gerem tempos de serviço também grandes . Da mesma forma, ao gerar tempos de chegada conjunto dos clientes, é necessário que os números aleatórios grandes gerem tempos de chegadas conjuntos pequenos. Isto garante que Y(0) seja mínimo e que Y(l) seja máximo

As técnicas de redução dc variância foram introduzidas também por von Neumann e Ulam como forma de obter um refinamento da simulação direta, basicamente utilizaram as técnicas conhci idas como roleta russa e ***"Splittinjf*** ((racionamento), das quais falaremos mais adiante. ,

O fator crucial ao decidir quando usar as técnicas dc redução de variância é se, na realidade, uma dada abordagem diminui a variância das estimativas e se assim for, sc a redução é suficiente para justificar os cálculos adicionais requeridos.

5.5.1 Método da amostragem estratificada

Seja ^ i , Ç„ números aleatórios uniformemente distribuídos entre 0 c I, então os valores

Eq. 5.5.2.1 f i - ^ i)

São variáveis aleatórias independentes com esperança dada por K(Q. Por isso

Eq. 5.5.2.2 / - £ /

c um estimador não-viesado dc E(fj), c sua variância c

Eq. 5.5.2.3 ¹ £ (/ (*) ***E{ffdx='***

e o desvio padrão de / c dada por

liq 5.5.2.4 *a* °

No método da amostragem estratificada a idéia é dividir o limite de integração cm algumas partes, por exemplo, ***cx,*** , < ***x*** < ***a,*** onde 0 a₀ < ai < ***a₂*** < ... < otk = I, c aplicamos o método primitivo dc Monte **Carlo** a cada parle separadamente. **0** estimador da cq. 5.3.1.1 é então da forma

Eq. 5.5.2.5 ô - **£ t** (« y - ***a,)^f{a,*** , ₊ (« , ***-a, ,)*„)***
J I ' I

onde **iij** representa a quantidade de números aleatórios gerados para o j-ésimo intervalo.

Afirmamos que O é um estimador não viesado da integral <P (a demonstração desse fato pode ser encontrada cm |Shimi/.u|), c sua variância é dada por

Eq. 5.5.2.6 ***%** " ' ^ f ' / (*) ' * ± ¹ I f **f(x)uX***

lista variância pode ser menor que cr^2 , com // V / / , se a estratificação for bem feita de maneira que as diferenças entre **OS** valores médios de f nas várias partes sejam maiores que as variações de f em cada parte. Quando os pontos da estratificação são prescritos a melhor forma de distribuir os pontos das amostras entre as partes é dc maneira que n j seja proporcional a

$$[(\ll, \ll.)\pounds;, / w * \{í:;,/w\}'| \quad \_ \ ,$$

Existem várias formas cie escolher otj, a mais simples é dividir o intervalo original cm k intervalos iguais, $(Xj \quad j / ^$, entretanto a melhor forma é escolher a, de maneira que a variação dc f seja a mesma em cada intervalo (parte), cm |l lammersley & I landscomb] este resultado é demonstrado.

|Naylor| aponta duas hipóteses que devem ser atendidas (não simultaneamente) para que se justifique a adoção da amostragem estratificada: a) a variância do valor estimado do parâmetro obtido por meio dessa abordagem deve ser muito menor que a determinada mediante a amostragem aleatória para um custo total dado; b) que para uma variância lixa específica do valor estimado do parâmetro, o custo correspondente a amostragem estratificada seja menor que o da amostra aleatória e sugere alguns cuidados que se deve adotar quando ulilixa-se a amostragem estratificada com o intuito de evitai surpresas desagradáveis, tais como o aumento da variância, são os seguintes:

1) Hspecificar uma variável de estratificação,

2) Conhecer a função dc probabilidade da variável dc estratificação;

3) Iispecificar o número de estratos;

4) Os limites dos estratos devem ser especificados cm função da variável utilizada no critério dc esl ratificação,

5) Deve-se especificar o tamanho total da amostra e o tamanho da amostra de cada estrato: •

o) li necessário indicar um método para obter aleatoriamente a amostra de cada estrato

5.5.2 **Método da ordem dc importância**

Suponha que lemos

Eq. 5.5.2., <, *Íf(^ i ^ i ^ i ^ ^)*

para qualquer função g c (í satisfazendo

Eq. 5.5,2.2 *G{x)* £ *g(y)Jy*

agora se restringirmos g a uma função dc valores positivos tais que

Êq. 5.5.2.3 *[x{y)dy-*

então G(x) é uma função distribuição para 0 < x < **1,** e, se t] c um número aleatório amostrado da distribuição G, então a equação 5.5.2.**1** mostra que f(r|)/g(T|) tem esperança ϕ e variância

Eq. 5.5.2.4 *a f y* */ (*) [f f i * "*(«)*

0 objetivo na amostragem por importância c concentrar a distribuição dos pontos da amostra nas partes dos intervalos que são mais importantes em vez de espalha-los na totalidade do intervalo. Para não viesar o resultado compensamos usando a distribuição f/g no lugar dc fcomo estimador.

Observamos que se f é também valorado positivo podemos pegar g proporcional a f; g cf, por exemplo. Então

Eq. 5.5.2.5 O **f** ^^-*dCÁx)* **f** ^**d(i(x)* ¹ , isso implica cm *c* ' * *g(x)* ᵛ ' ʲ ⁽ ⁾ ç / 0) • ᵂ *C* • O

isso nos leva a

Eq. 5.5.2.6 o ' f| ^ <D *dG(x)-0*

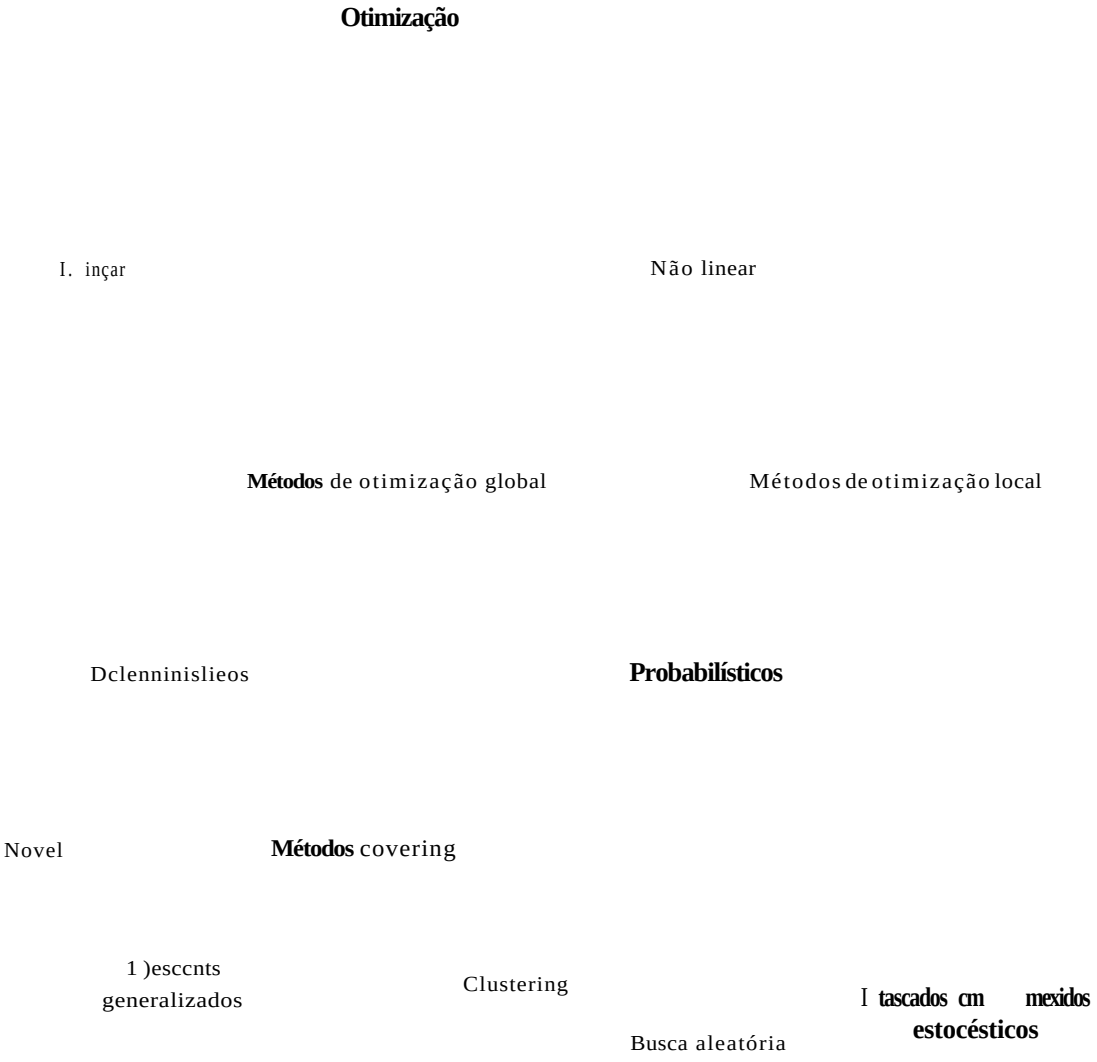


Figura 1.1.1 **- Taxonomia** para otimização apresentada em **[Shang** & Wah|.

Muito dos problemas existentes no mundo real admitem várias ou infinitas soluções, a teoria da otimização procura encontrar a melhor dessas soluções, lim [Himmelblau! c apresentada, uma outra forma de classificar os métodos dc otimização:

- Métodos analíticos

2 - Métodos numéricos

3 - Métodos gráficos

Perfeito!

Infelizmente, para que possamos amostrar f/g, devemos conhecer g, c para conhecermos g (f/O) devemos conhecer Q>, e se conhecermos O, não precisamos dc Monte Carlo para estima-lo'

Contudo o raciocínio não está completamente invalidado. Observamos que sempre pegamos um estimador não viesado de <**1**\ qualquer que seja a função positiva g que usarmos, ver [Hammersley & Handscombj. Nosso objetivo e selecionar algum g para redu/.ir o desvio padrão do nosso estimador, sendo este estimador a media dos valores observados dc f/g; c, dc antemão sabemos que teremos um pequeno valor da variância da amostra sc t/g é uma constante, evidentemente não conseguiremos que esse valor seja de fato constante pelas razões já mencionadas, queremos portanto que g imite f, por um lado, deste modo a taxa dc f/g varia pouco, por outro lado lemos que restringir nossa escolha de g a funções que teoricamente podemos integrar para satisfazer a equação 5.5.2.3. Estes requisitos são claramente conflitantes: g deve ser suficientemente simples para que possamos ter sua solução teórica enquanto f deve ser tão complicada para exigir a utilização de uma solução por Monte Carlo. Deve então existir um compromisso entre estes requisitos e uma boa avaliação desse compromisso produzirá um estimador de <I> com desvio padrão substancialmente menor do que o estimador conseguido com a aplicação do método de Monte Carlo sem a utilização de técnicas dc redução de variância, em | Hammersley & Handscomb] é apresentado um exemplo comparativo da aplicação dessa técnica.

5.5.3 Método das variáveis de controle

Uma outra técnica que reduz a variação de fi» é a técnica das variáveis de controle que consiste basicamente em dividir a integral 5.3.1.1 em duas partes, como mostrado abaixo

Eq. 5.5.4.1 $\int_a^b f(x)p(x)dx = \int_a^b [f(x)-g(x)]p(x)dx + \int_a^b g(x)p(x)dx$

c integramos separadamente, a primeira matematicamente c a segunda por Monte Carlo. Portanto tp deve ser uma função bem simples mas deve ter um comportamento semelhante a f, dc modo a absorver sua variação, a variável (p(x) é denominada variável de controle dc f(x).

Existem várias formas de ver este método. Por exemplo, quando estimamos um parâmetro desconhecido O por meio dc um estimador t, podemos buscar outro estimador f que tenha forte correlação positiva com t, c cuja esperança é uma quantidade numericamente conhecida <f>\ Amostramos t et' simultaneamente e usamos t - 1 ' t como estimador dc O.

5.5.4 Método das variáveis antiteticas

Ao contrário do método das variáveis dc controle este método procura um estimador t" que lenha uma forte correlação negativa (com esperança desconhecida). Então ' (/ i /") será um estimador não viesado de O (ver |Hammersley & Handscomb|), c a variância da amostra é dada por

Eq. 5.5.3.1 $$Var\ (/1\ r) \quad \frac{1}{4}Var/ + \frac{1}{4}Varl'' + \frac{1}{2}cov(/,/'')$$

na qual cov (t , t") é negativa pode as vc/.cs conseguir valores menores que Var t por uma escolha adequada dc t".

Por exemplo, 1 - Ç é distribuído uniformemente sempre que Ç o é, assim f(Q c f(l-Q são ambos estimadores não viesados de O. Quando f é monotônica, f(Q c ({\-Q serão negativamente corrclacionadas,c assim podemos usar

Eq. 5.5.3.2 $$\frac{1}{2}(< + = \frac{1}{2} f(<;) + \frac{1}{2} -f(\text{-}Ç)$$

como estimador dc O. Esse método segundo |Hammersley & Handscomb| foi introduzido por Hammersley c Mortoncm 1956, e é baseado no seguinte teorema.

Teorema: *Se I denota o in/imo de lar {£,] quando Ioda possível dependência eslocásíica ou funcional entre os § são consideradas, sujeito a cada § estar uniformemente distribuído entre 0 e I, então, sob a condição de que g, são funções limitadas,*

onde ***x*** *denota a classe de funções x(z) com as propriedades (i) x(z) c um mapeamento* ***(I,*** *l) d< intervalo (0,1) nele mesmo, e (ii) exceto quando muito em um número finito de pontos z, dx dz l.*

Muito mais detalhes a respeito deste método de redução da variância encontra-se em 11 lammersley & Handscomb], [Nayloret. al.J, [Yakowitz].

5.5.5 Método da roleta russa e fracioiiaiiiento

Tanto a idéia quanto o nome dessa técnica são devidos a von Neumann e Ulam. A amostragem é feita cm estágios, sendo possível examinar a amostra cm cada estágio c classiliea-la como sendo de alguma forma interessante ou não interessante.

0 que desejamos é que um computador efetuando cálculos a partir das amostras, gaste maior parcela de esforço com as amostras interessantes e menor parcela com as não interessantes I ;so pode ser feito lracionando-se *(splitíing)* as amostras interessantes cm partes independentes c assim aumentar o número de amostras interessantes e, por outro lado, eliminando-se certa porcentagem das amostras não interessantes

0 primeiro procedimento é o fracionamento e o segundo a roleta russa. A eliminação é feita através de um jogo dc azar suplementar. Perdendo-se o jogo a amostra é descartada e ganhando-se o jogo ela é considerada acresccntando-lhc um peso para compensar o fato dc algumas amostras terem sido eliminadas Por exemplo, nos problemas originais para os quais o método foi concebido, a difusão de partículas,

partículas que caem cm regiões interessante são divididas cm k subparticulas independentes, cada qual com l/k do peso da partícula original c então são consideradas no processo. Partículas que caem cm regiões não interessantes são fundidas cm um número menor de partículas mais pesadas.

5.5.6 Outros métodos de redução da variância

Existem outros métodos para redução da variância lais como: método da análise multivariada, das funções ortonormais, da amostragem regressiva que podem ser encontrados cm [Hammersley & I landsComb] c outras referencias na nossa bibliografia, além destes temos o método sugerido por |Salib].

5.6 Aplicações

Uma pergunta que fatalmente aflora quando nos debruçamos sobre os métodos de Monte Carlo é: aonde aplicar tais métodos? Muitos críticos encontrariam poucas aplicações para estes, tentando, talvez, justificar seu parecer com argumentos como: os métodos analíticos c numéricos determinísticos nos fornecem respostas melhores que as de Monte Carlo ou a precisão obtida com o emprego destes é pobre. Contudo existem situações nas quais não se consegue uma resposta adequada com a utilização dos métodos numéricos determinísticos para o problema (as vezes sequer sc consegue uma resposta), ou a estrutura do problema é dc natureza probabilística c a utilização dos métodos de Monte Carlo consegue incorporar esse aspecto dc forma efetiva, ou, como é parte dc sua própria história , quando não existe outra forma de modelar o fenómeno e aí sua utilização é inevitável.

J. von Neumann e Ulam propuseram uma forma dc resolver sistemas de equações lineares fazendo uso dc uma cadeia dc Markov estacionária. Para o sistema $\sum_{j=1} a_{ij} x_j = b_i \,,\ 1 \le i \le n$

onde os ci_{ij} c bj são constantes e os x, os números a serem encontrados. Usando a notação matricial temos Ax b. Suponhc-sc então que A pode ser escrita como

A I - **11**

onde I é a matriz, identidade e todos os autovalores de **11** tem magnitude menor que I. Então

x A 'b (l-ll) 'b lb i llb) ll'b l ... I U"b I ...

que converge se a suposição sobre os autovalores dc **11** for verdadeira. Sendo X (t) uma sequencia (t c I) X(l)...X(t), onde os X(i) tomam valores no conjunto { I, 2, ... , n}, sendo n a ordem do sistema. P uma função de probabilidades que designa uma probabilidade positiva a todo caminho possível X(') de todo tamanho possível t 0, I, 2, ... P(X(t)) > 0, V l > 0 c V X(l). Os caminhos X(t) são simulados de acordo com a distribuição P(), e a quantidade

ÍT'V0 l),A'(0

/>(*(/))

onde hjj são as coordenadas da matriz II. X(0) c um valor inicial fixo I < X(0) < n. esta simulação é repetida m vezes para obter Vi(l,„;X(())), ... , V,„(l„;X(0)) e a média amostrai

'»'(°(°)) $\frac{1}{/}$ y]K/(/;A(0)) é calculada. Verifica-se que E| V(l;X(ü))| (A 'b)x(0) xx_{o},, onde, o índice indica a coordenada do vetor x, para mais detalhes ver | Yakowitzf

0 método dc Monte Carlo pode ser utilizado também para solução de sistemas de equações diferenciais, solução do problema do valor limite envolvendo a equação dc Laplace, equação de Poisson e equação do calor As equações diferenciais podem ser vistas como modelos para fenômenos físicos microscopicamente aleatórios. As aplicações iniciais deste método foram em estudo de transporte de neutrons, projeto de reatores nucleares e outros problemas de física da partícula. [Yakowitz] ilustra a solução dc um problema de valor limite bidimensional envolvendo equação de Laplace (conhecido como problema de Dirichlet), nesse caso novamente lança-se mão do construto passeio aleatório para a solução.

Uma outra aplicação c abordada cm [Goldfeld & Dubi] c envolve a engenharia dc segurança, manutenibilidade e disponibilidade dc sistemas.

Uma aplicação clássica do problema da ruína do jogador c em controle dc estoque. Muitos trabalhos utilizando métodos probabilísticos tem sido publicados cm várias áreas, em particular cm otimização global da qual lalaremos no capítulo 6.

5.7 Conclusões

Nos últimos anos os métodos de Monte Carlo têm sido consideravelmente evidenciados. Isto ocorreu principalmente devido a um maior reconhecimento daqueles problemas nos quais eles são melhores e, muitas vezes, a única técnica disponível. Tais problemas têm aumentado (em quantidade e complexidade), particularmente porque as técnicas dc redução dc variância tem transformado os métodos dc Monte Carlo em eficientes onde anteriormente eram considerados ineficientes.

I Yakowitzl simulou a integração por Monte Carlo de algumas funções (método sucesso/fracasso e média amostrai) sem a incorporação dc técnicas de redução de variância c com sua incorporação (variáveis dc controle e variáveis antiléticas) e os resultados são expressivos.

De qualquer sorte é necessário reconhecer a importância desses métodos, no mínimo em problemas onde os outros métodos numéricos são proibitivos do ponto dc vista computacional ou simplesmente não se aplicam.

Capítulo 6

Otimização global

"Para ser considerado quebra-cabeças, um problema deverá ser caracterizado por algo mais do que uma solução certa. Devem também existir regras que delimitem a natureza das soluções aceitáveis, bem como as passagens através das quais se devem obter tais soluções."

T. S. Kuhn

Capítulo 6

Otimização global

6.1 Introdução

Em muitas aplicações dc engenharia os problemas **a** serem resolvidos só podem ser formulados como problemas dc otimização com funções não lineares, fim muitos casos procura-sc um mínimo local tio qual a função admite seu menor valor (em lodo seu domínio de definição), isto é, um **mínimo** global

O problema de projetar algoritmos que possam distinguir entre o mínimo global c os numerosos mínimos locais que por venlura venham a existir na função é conhecido como o problema de otimização global, [Cetin et. al.|. l-xislem diferenças notáveis entre otimização local c global (técnicas dc otimização global tem sido desenvolvidas para resolver problemas não lineares c não convexos).

Os algoritmos desenvolvidos para otimização global podem ser divididos grosso modo cm duas classes: probabilísticos **e** determinísticos.

Discutiremos aqui basicamente os algoritmos probabilísticos para otimização global. As aplicações de otimização global incluem projetos dc sistemas dc comunicações, sistemas de controle não lineares, circuitos eletrônicos c filtros óticos, ver |Hassoun|.

O problema de otimização global permaneceu marginalizado por um longo período de **tempo em** um campo de atividade e pesquisa chamado programação matemática. Existem algumas razões para isso. A primeira é simplesmente que a otimização global é de difícil solução. Uma segunda razão diz respeito a qual classe dc algoritmo (necessariamente diferente daqueles utilizados cm **otimização** local) pode ser utilizado para tentar resolver o problema numericamente? (uma observação freqüentemente feita é que tentar resolve-lo é um absurdo dc acordo com 1**1** liriart-Urruty|).

Porque otimizar globalmente[7] cm termos práticos, todo problema dc otimização que é proposto é um problema de otimização global, a questão é que muitas vezes nos conformamos com uma otimização local.

seja porque o problema assim nos permite seja pelo fato de não contarmos com algoritmos adequados para soluciona-lo globalmente. Qual a situação aluai? existem atualmente notáveis diferenças 'entre as características de otimização local e global, na primeira algumas idéias básicas governam os aspectos teóricos de interesse (aproximações dc primeira e segunda ordem, linearização, penalização, etc.), alem disso estão disponíveis numerosos algoritmos numéricos bem documentados em bibliotecas, empresas e centros de pesquisas. Em otimização global os seguintes aspectos ficam evidentes: uma incrível quantidade dc idéias (algumas inclusive muito ingénuas segundo 1**1** liriart-Urruty|), de variadas origens e naturezas, dificuldade de acesso a algoritmos numéricos (| **1** liriart-Urruty | especula inclusive quanto a confidencialidade destes ou sobre sua inadequação a outros contextos que não sejam aqueles para os quais foram concebidos).

6.2 Algoritmos para procura de extremo local e global

Consideremos o problema de otimização de encontrar o ponto extremo de uma função velorial multidimensional da forma y: I) < R" > R, onde o espaço de procura **1)** é um subconjunto compacto de R" Um ponto extremo é um ponto x* (I) tal que f(x*) é um valor máximo (ou mínimo) dc f(x). O máximo é local se f(x*) é máximo cm |x* - x| < e, c fiV) é máximo global se f(x*) > f(x), Vx t *D* c R"

Assumimos também que lodo mínimo local x* de f(x*) cm I) satisfaz as condições:

Eq. 6.2.1 $\frac{'^{1/(/)}}{/?X}$ **. o**

Eq. 6.2.2 y^{i} $\frac{\backslash}{Sx^{2}}$ ' *v* o. v> « R^{n}

6.3 Condições para otimalidade global

O que **nos** interessa determinar em um problema de otimização global é x € D c R" tal que f(x*) < l{x), V x f I), x / x\ ou ainda x* € D c R" satisfazendo f(x') < f(x), V x c I) o N, onde N é alguma vizinhança de x*, em otimização global a vizinhança N passa a ser o espaço total definido em I)

Veremos alguns resultados para otimização global baseados cm integração, li óbvio que sc imaginarmos o mundo dos problemas de otimização especialmente estruturados com função objetivo continua e um conjunto viável compacto verificaremos imediatamente que as condições práticas reais dificilmente poderão ser enquadradas em tal contexto.

Assumindo que o conjunto S de restrições é o fecho de um conjunto aberto limitado não vazio (um exemplo c um conjunto convexo com interior não vazio) c f:I) > R c uma função continua. Um resultado que segundo |Hiriart-Unuty| vem dc Laplacc c que hoje cm dia é explorado e utilizado no contexto dc grandes desvios em probabilidade e estatística, afirma que

Eq-[6]-3.i $\quad \underset{v<D}{Max\,A')} \quad \underset{A>KC}{iim} \; Nj/^{H/X}$

A questão agora colocada é: como podemos aproximar **eficientemente** a integral dada na eq. 6.3.1 **acima?** Quando k > **+00** o procedimento c estável[7] questiona **11** liriarl-Uiruty|. A integral pode ser aproximada rapidamente usando o método de Monte Carlo da média amostrai.

Se uma função g contínua, é estritamente positiva sobre D, a transformação f **Log** g caracterizada na eq. 6.3.1 acima declara que na seqüência

Eq. 6.3.2 $\quad r_k(x^*) \quad \mathbf{J} \quad \left(f \quad \backslash \right) \quad cJx \;\; x^* \, G \, D$

x* é um máximo global dc g em D se somente sc a seqüência {r_k(x)} é limitada.

4 - Métodos experimentais

5 - Métodos usando estudo de casos

Os métodos analíticos baseados no cálculo diferencial c no cálculo das variações para que possam ser aplicados em um determinado problema este deve estar descrito em termos matemáticos. Desse modo, as funções c variáveis podem ser manipuladas pelas conhecidas regras algébricas. Entretanto quando tratamos com problemas dc médio ou grande porte sua utilização torna-se inviável c aí outros métodos analíticos devem ser utilizados.

Os métodos numéricos usam a informação sobre o passado para gerar uma solução melhor por meio de procedimentos interativos; estes são os métodos utilizados quando a solução analítica não é possível.

Métodos gráficos usam, evidentemente, os gráficos da função objetivo e das restrições de forma a apontar a solução do problema por simples inspeção. O grande inconveniente aqui é que com problemas dc mais dc duas variáveis independentes tornam-se impraticáveis.

Nos métodos experimentais procura-se o extremo da função em estudo por experimentação direta sobre as variáveis do processo ou por manipulação dc uma descrição matemática do processo. Os resultados dc um experimento são usados para decidir onde deve-sc alocar o próximo experimento dc forma a obter um resultado melhor.

Os métodos de estudo de caso envolvem a estimação de um certo número dc soluções para o mesmo problema dc maneira que sc possa determinar a melhor solução entre cias. A solução obtida desse modo pertence ao conjunto dc pontos viáveis, mas, não necessariamente, é a solução ótima.

Qualquer que seja o problema entretanto sua solução é, na grande maioria dos casos, não trivial, o que tem levado diversos pesquisadores a estudar c propor técnicas c algoritmos para a sua solução.

Nosso objetivo ao longo desse trabalho é no desenvolvimento de um algoritmo numérico (dc base analítica, portanto), que resolva o caso geral do problema, isto é, quando a função objetivo c as restrições são funções reais dc variáveis vetoriais c não lineares. A idéia é desenvolver um algoritmo utilizando

Teorema 6.3.1 - *Assuma que I) ê o fecho de um conjunto aberto limitado nào vazio e /:/) > l< é contínua. Suponha ainda que f é globalmente maximizada em /) em um único ponto x . Então a sequência*

$$m \text{ — } . \text{ k } \mathbf{1.2....}$$

converge para x quando k > v.

6.4 Algumas abordagens em otimização global

Como já visto anteriormente os métodos apresentados no capítulo 2 nos leva, em geral, a,extremos locais. Uma forma de procura do extremo global é estender esses processos a diversos pontos do domínio da função de interesse. Por exemplo, podemos aplicar o método de Cauchy (steepest desceul) em diversos pontos do intervalo de interesse e ir registrando os valores de mínimo (ou máximo) encontrados, de tal sorte que ao final do processo teremos uma coleção de pontos

$$x_i \setminus x_2 \setminus \ldots, Xp^*$$

e o processo se resume em comparar os x, / Xj de tal forma a descobrir $f(x_k^*) > f(x,*)$, $Vx_{\text{„}}$ i 1,2, k-l, ki I, .., p. Dessa forma tendo-se determinado o provável mínimo (máximo) minimorum da função, lisse processo, aparentemente eficiente de otimização global tem muitos inconvenientes, tais como:

1) Como cm geral não tem-se idéia da topologia da função objetivo a ser otimizada pode-se (c cm geral isto ocorre) perder pontos extremos, isto é, ao final do processo não constarão da lista $\| , X; , \ldots, x_p^*$, e portanto poderemos ter construído uma lista incompleta de pontos candidatos;

2) A determinação de quantos pontos devem ser procurados e a partir de que pontos iniciais não e trivial (lembre que nào conhecemos a topologia da função).

Bsse método apresenta lodos os inconvenientes já apontados para si na otimização local,

4) lísse procedimento é coinputacionalmenlc ineficiente.

6.4.1 fundamento

Uma outra forma dc procura do extremo global da função é o tunelamento introduzido por A. V. Levy c A. Montalvo cm um trabalho dc 1985 e que [Cethl, Barhen & Burdick] modificaram No tunelamento de Levy e Montalvo **o** método era composto de duas fases a cada ciclo de busca: uma fase de minimização local c uma fase de tunelamento. Na primeira, algoritmos de otimização tais como o método do gradiente ou **o** método dc Newton são empregados para minimizar f(x). Assume-se que iniciando no ponto xi, a minimização converge para **o** primeiro mínimo local Xj , que satisfaz as condições dadas pelas equações 6.2.1 e 6.2.2. Na segunda, a função de tunelamento é definida como

Eq. 6.4.1 $T(x.x\backslash)$ —. ondef(x) » f(x)-f(xÍ)

A fase dc tunelamento bouça encontrar zeros dc T**(x,X|*),** isto é, T**(x,xi*)** 0 é resolvido para qualquer $\backslash_2$ tal que $x_{(}$* z x_2, mas f(x;) f(xi*). O denominador é um polo dc tamanho (X, localizado no mínimo local Xj previamente determinado, dessa forma prevenindo o algoritmo para encontrar zero a partir dc xj pois este é polo da função de tunelamento. **0** novo zero é utilizado como novo ponto inicial dc procura e o processo é repelido seqüencialmente até que o critério dc parada seja atendido, por exemplo, falha na procura do zero dentro de um determinado intervalo dc tempo da CPU. **0** último mínimo encontrado é assumido como mínimo global.

[Cetin, Barhen & Burdick | apontam algumas desvantagens para este método:

I) **0** tamanho do polo depende do problema;

2) O algoritmo pode achar outro mínimo local x_2 tal que f($x_{\cdot}$) = f(x_2);

3) Divisão por zero de f(x) com x -> QO $(t\backslash x)$ --,> 0 com x -> oo);

4) O algoritmo para encontrar o zero é baseado no método de Newton modificado que requer encontrar raízes de uma função escalar com múltiplas variáveis (este procedimento pode ser computacionalmente muito caro e como ainda não existem algoritmos para encontrar /.eros globalmente convergentes o critério de parada **fica** indefinido). Uma observação importante aqui é que [Cetin, Barhen & Burdick| propuseram e implementaram esse algoritmo em hardware analógico VLSI.

6.4.2 Métodos de duas fases

i) Busca aleatória pura

Lsse algoritmo consiste na geração de uma seqüência dc pontos uniformes independentes e identicamente distribuídos numa região factível S, e conseqüente seleção dos melhores pontos encontrados

Passo 0: Laça n I, $y_{(1}$ - <

Passo I: Pegue um ponto x da distribuição uniforme cm S.

Passo 2: Se **f**(x) y„ **i,** então faça y„ **f**(x) e x„ x. ('aso contrário, faça y„ y_{n}.| e x_{n} x„.j.

Passo 3: Incremente n e volte ao passo I.

ii) Multistarl

É um método mais eficiente que o anterior, com a desvantagem de inevitavelmente encontrar o mesmo máximo local mais dc uma vez. Consiste em aplicar um procedimento de busca loca! L antes do' global

Passo 0: Laça n I, $y_{(1}$ - co

Passo I: Pegue um ponto x da distribuição uniforme sobre S e aplique a x o procedimento local L obtendo x .

Passo 2: Se i(x') > y„.|, enlão faça y„ - f(x') e x„ - x\ caso **contrário** laça y„ y„.i e x„ x„-i.

Passo 3: Incremente n e retorne ao passo I.

6.4.3 Métodos de grupamento

A idéia básica que dá suporte a esse tipo de método é iniciar a partir dc amostras uniformes sobre S c criar grupos fechados dc pontos c aí aplicar L (procedimento local) aqueles grupos. Duas formas diferentes de gerar esses grupos têm sido propostas:

- Redução - Consiste cm reter uma fração y dc pontos da amostra consistindo de pontos com alto valor da função;

- Concentração - Consiste na transformação da amostra inicial aplicando alguns passos *steepest ascent* em lodos os pontos e escolhendo os pontos adequados. Segundo [Boender & Romeijn] o grupamento conseguido em ambos os casos é sempre o mesmo.

i) *Density clustering*

A idéia nesse caso é que o conjunto dc níveis da função f na vizinhança dc um máximo local é aproximado por um elipsóide, ou, em outras palavras, a função é localmente aproximada por uma função quadrática. **0** êxito em otimizar globalmente uma função depende, segundo |Boender & Romeijn], do quanto essa aproximação é boa. No esquema de agrupamento esse método utiliza uma distancia crítica que faz uso do cálculo da matriz I lessiana no ponto.

ii) Grupamento com ligação simples

Aqui os grupos são formados seqüencialmente. Cada grupo novamente é iniciado a partir dc um ponto procurado, depois que um grupo C é iniciado, verifica-se os pontos não agrupados através do critério

</(x.C) minix v|, seja mínima. Este ponto então é adicionado a C, depois que o procedimento é repetido até d(x,C) exceder algum valor crítico r_k. |IJoender & Romeijn] afirmam que o grupamento com ligação simples aproxima o conjunto dc níveis com mais precisão que o *densily cluslering.*

iii) Ligação simples multiníveis

Lsse método, segundo [Boender & Romeijn], combina a eficiência computacional do método de grupamento com as virtudes teóricas do nuiltistart. O procedimento dc busca local L é aplicado a todo ponto da amostra, exceto sc existir outro ponto da amostra com alguma distancia crítica que tem grande valor de função.

6.4.4 Métodos de busca aleatória

A classe de métodos de busca aleatória consiste dc algoritmos que geram uma seqüência de pontos na região viável seguindo alguma distribuição de probabilidade preespecificada, ou seqüência de distribuições de probabilidade. Os algoritmos mais básicos dessa classe são operacionalizados pela geração de pontos de uma distribuição de probabilidade, isto e, os pontos são variáveis aleatórias independentes e identicamente distribuídas. Alternativamente a distribuição a partir da qual a seqüência é gerada pode ser atualizada adaptativamente, isto é, dependendo do número de iterações e dos pontos já iterados. [Boender &

RomeijnI Irisam que não existe implementação eficiente destes algoritmos, contudo os resultados teóricos que podem ser obtidos por esses algoritmos são muito interessantes.

i) Busca aleatória pura

risse algoritmo, como já visto anteriormente, é o mais simples da classe dos métodos dc busca aleatória.

lisse algoritmo oferece uma garantia probabilística no sentido que o máximo global será encontrado com probabilidade I quando o tamanho da amostra aleatória tende para o infinito. Uma questão importante aqui é saber se esse método é melhor do que os métodos determinísticos que fazem a busca cm grades fixas, nos quais a função é calculada em cada ponto da grade sobre S. [Boendcr & Romeijn] apontam como uma vantagem óbvia desse método o fato de poder ser implementado adaptativamente.

ii) Busca aleatória

Seja {Un}n o' uma seqüência de distribuição dc probabilidade sobre R^d.

Passo 0: Faça n - O e escolha um xo e S

Passo 1: Gere y„, i a partir da distribuição u_n

Passo 2: Faça x„, i - D(x_n, y_{n+1} i), incremente n e retorne ao passo 1.

O mapeamento D com domínio S x R^d e ***range*** S satisfaz a condição:

/ (" M M *) » * » ' ˢ - '(ᵖ(".y))*/W

essa condição assegura que a seqüência {f(X_n)}ₙ o" é monotônica não decrescente com probabilidade 1. Se

n o t*(A))-o
/i=o

l'ara lodo conjunto A tal que <p (A) > 0, esta seqüência convergirá para f com probabilidade I, de acordo com [Boender& Romeijn],

Dois outros métodos dessa classe são a busca adaptativa pura e a busca adaptativa comentados c devidamente explicados em | Boender & Romeijn], inclusive mostrando que o suporte teórico (teoremas de ZABINSKY & SMITH e dc ROMKIJN & SMITH).

6.4.5 Reçozjmento simulado (simulaled anncaling)

lisse método não usa explicitamente a informação do gradiente e dessa forma é aplicável a uma grande classe dc funções (especificamente funções cujo gradiente torna o processo de otimização computacionalmente muito caro ou funções não diferenciáveis).

O recozimento simulado é análogo ao comportamento do processo físico do recozimento do metal fundido. Acima da sua temperatura de fusão um metal entra numa fase onde os átomos (partículas) são posicionadas dc forma aleatória de acordo com a mecânica estatística. Como em todos os sistemas físicos, as partículas do metal fundido irão buscar uma configuração (estado) dc energia mínima no resfriamento. Uma configuração dc energia mínima significa um estado altamente ordenado, como por exemplo, como cm um metal livre de defeito, para alcançar esse estado o metal é recozido: primeiro, o metal é aquecido acima do seu ponto dc fusão e então resfriado lentamente até ele solidificar cm uma estrutura cristalina perfeita (a lente do telescópio dc monte Palomar levou 2 (dois) anos nesse processo). O resfriamento lento é necessário para prevenir deslocamentos desordenados de átomos. O estado do cristal livre de defeito corresponde á configuração global mínima de energia.

[Hassoun] apresenta alguns resultados da mecânica estatística (objeto central do estudo da física da matéria condensada). Um resultado fundamental da física é que em equilíbrio térmico cada um dos possíveis estados x ocorre com probabilidade

H(x)

.te.V

onde X é o conjunto dos possíveis estados e k é a constante dc Boltzmann. A equação acima é conhecida como distribuição de Boltzmann-Gibbs. Além disso define-se as seguintes probabilidades de transição W(x->x') e VV(x'-vx) c daí a condição suficiente para manter o equilíbrio térmico é

P(x) W(x-»x') = P(V) W(x'->x) o que dá

, $\sim^{*x}$ '. - -^-^ = e w , onde AE = E(x') - E(x). A escolha mais comum para

W^x^x') é o algoritmo de Metropolis

$$lV(x\text{->}x') = \begin{cases} \text{I sc Al-} < 0 \\ \text{AI?} \\ \text{c * I cm oulros casos} \end{cases}$$

Esse algoritmo tem a vantagem de fazer mais transições para estados de menor energia que o anterior e assim alcançar o equilíbrio mais rapidamente. Deve-se notar que transição de estados de baixa energia para alta energia é possível exceto quando T = 0.

A otimização por recozimento simulado consiste em encontrar o mínimo global de uma função ***í[x)*** usando esses resultados; mais informações podem ser obtidas em [Hassoun], [Boender & Romeijn] e também em [Floudas & Pardalos].

6.5 Critérios para a comparação de algoritmos de otimização global

[Stuckman & Easorn] aponta dois critérios para estudo comparativo das técnicas de otimização global:

1) Convergência

2) Performance sobre uma série de funções de teste

Quando utilizamos a convergência, as técnicas (algoritmos) de otimização são avaliados em relação a suas capacidades dc convergir para uma solução ótima. Hssa abordagem pressupõe a prova teórica de convergência c implica cm um grande número de suposições para obtenção da solução. Além disso a convergência pode ser provada somente, em geral, para uma limitada classe de funções.

O segundo método consiste em comparar as diferentes técnicas de otimização global pela implementação destas em algumas funções de teste (dizer, uma dezena). Estas funções são importantes porque simulam muitos dos atributos de aplicações reais, os resultados obtidos indicam como os métodos de otimização global se comportam.

Nesse trabalho não apresentaremos nenhuma comparação do algoritmo aqui proposto com outros já existentes pois nosso objetivo nesse momento não é fazer um estudo comparativo do algoritmo mas sim mostrar (pie ele funciona adequadamente.

6.6 Conclusões

l)Algumas dificuldades típicas surgem da própria construção do modelo matemático, de acordo com [llimmclblauj, algumas delas são:

i) A função objetivo a ser otimizada pode ser insensível a mudanças nas variáveis de decisão independentes;

ii) A função objetivo ou uma ou mais restrições pode ser ilimitada no intervalo de busca, ou as derivadas parciais das funções do modelo pode vir a ser ilimitada (um exemplo é um modelo com polinómio no denominador).

iii) Uma diferença de escala entre as variáveis, isso ocorre quando os termos da função objetivo são de ordem de magnitude muito diferentes (por exemplo, = lOOxi² - 0.0lx₂²).

iv) A interação entre variáveis cm um modelo pobre do ponto de vista de projeto (por exemplo, *í\\)* $2x,x_2$ i 10, nesse caso os valores de xi e x_2 variam em intervalos muito grandes para um mesmo valor de X1X2)

v) O chamado efeito nulo pode existir no modelo, por exemplo, suponha a seguinte função objetivo

$f(xi,x_2)$ Ki^2 » $2x,x_2$ I x_2^2 I 2 $(x, I x_2)^2$ I 2

Realizando a seguinte transformação xi + x_2^m x, e assim fl^x) = x + **2,** precisamos então procurar o extremo variando apenas x.

Outras dificuldades surgem relativas agora as técnicas numéricas de solução, por exemplo:

2) Como podemos obter suposições iniciais adequadas para as variáveis independentes? porque **quando o** problema contem funções não lineares, pode existir mais de um extremo, um aspecto ausente na análise linear. **Consequentemente,** se as suposições iniciais para as variáveis estão muito longe do extremo global, a otimização pode terminar em um outro extremo que não o global;

3) Como podemos manusear os aspectos estocásticos da variável real?

4) ('orno podemos reduzir os erros numéricos computacionais? erros de arredondamento reduzem a efetividade de muitos algoritmos. Aspectos relativos a estabilidade concernentes a resposta cm questão: se a solução aproximada do problema de programação não linear converge no limite para a solução original do problema

5) ('orno mencionado na introdução, não podemos esperar encontrar condições necessárias e suficientes para otimização global para todas as classes de problemas de otimização, contudo, especialmente para os casos estruturados (e eles podem formar uma grande classe) podem ser manuseados para obter-sc resultados úteis A mensagem todavia é clara: para lidar com otimização global precisamos

técnicas probabilísticas (transformando o problema dc otimização cm um problema de integração) que, como qualquer método numérico, gere uma seqüência convergente para sua solução.

Esse trabalho tem a intenção de permitir que os interessados em resolver problemas dc otimização cm geral passem a contar com um algoritmo rápido, eficiente, e, principalmente, que possa ser implementado cm ambientes com processamento paralelo. Desse modo procuramos estrutura-lo dc modo a torna-lo, tanto quanto possível, aulocontido. Essa característica se rcíletc também na organização dos capítulos. Os capítulos de I a 6 tratam de temas distintos e aparentemente desconexos contudo ao inteirar-se do conteúdo dos capítulos 7 a 9 o leitor entenderá o inter-relacionamento existentes entre cada um deles. A idéia básica nos capítulos dc I a 6 foi dc realizar um *survey* pelas diversas técnicas c temas dc forma a fornecer uma revisão bibliográfica atualizada destes temas.

O capítulo 2 apresenta alguns conceitos básicos necessários ao entendimento de alguns conceitos de otimização não linear, traz as condições de otimalidade e alguns algoritmos determinísticos utilizados cm otimização.

O capítulo 3 apresenta conceitos, técnicas c arquiteturas dc máquinas paralelas. A idéia aí é realizar levantamento do ponto de vista evolutivo desse tipo de máquina apresentando o estado da arte. Procuramos, tanto quanto possível, obedecer a um critério cronológico de surgimento dc novas implementações, melhorias técnicas (tanto dc *hardware* quanto dc *software)* c mudanças conceituais com o intuito dc conseguir aumento de performance cm geral c dc velocidade cm particular.

O capítulo 4 apresenta a teoria, técnicas e aplicações de simulação dc modo a deixar o leitor inteirado das dificuldades e formas dc utilização da simulação.

O capítulo 5 apresenta os métodos de Monte Carlo, ilustrando as técnicas dc geração de números aleatórios, teste dc seqüências dc números aleatórios, métodos dc redução dc variância, passeio aleatório c aplicações.

O capítulo 6 trata dc otimização global cuja solução atrai o interesse dc um número crescente de

de um maior desenvolvimento da matemática, possivelmente mudar nossa abordagem, e criar novas ferramentas matemáticas, dc acordo com 111 iriart-Urruty|;

6) **[Boender & Romeijn] afirmam que o problema de otimização global é inerentemente insolúvel em um número finito de passos;**

7) **lisse é um campo vasto e aberto a pesquisas e contribuições além de ser de enorme interesse em vários contextos;**

8) As abordagens probabilísticas para a solução desse problema têm evoluído e **tornado-se sofisticadas além dc procurar resultados teóricos em que possa assentar-se de forma mais efetiva;**

9) **|Arnold| trata desse problema a luz da teoria da catástrofe. Os problemas de máximos e mínimos fazem aparecer singularidades, bifurcações e catástrofes. No prisma da teoria da catástrofe em um sistema de controle no espaço dc fases, não temos apenas um vetor velocidade como nos sistemas evolutivos usuais, mas um conjunto completo de vetores denominado a indicatriz das velocidades permissíveis. O problema do controle é escolher, a cada instante, um vetor velocidade da indicatriz e assim alcançar um alvo. |Arnold| comenta c estuda estas questões e afirma que muitas singularidades novas e interessantes surgem no estudo dc problemas dc otimização com vínculos, como, por exemplo, no problema de contornar obstáculos. Seu estudo conduziu a novos resultados em uma das áreas mais clássicas da matemática - a geometria das superlicies suaves no espaço tridimensional.**

10) **[Hassoun] afirma que o sucesso de um método de busca global em localizar uma solução globalmente ótima de uma dada função fi» é dado pelo equilíbrio existente entre o processo de exploração, o processo dc direcionamento** *(guidance process)* **e o processo de indução a convergência. O processo de exploração é usualmente de natureza estocástica. O processo de direcionamento é um processo implícito ou explícito que avalia a qualidade relativa do ponto buscado (sua função é conduzir o algoritmo para regiões dc alta probabilidade de solução), finalmente, o processo de indução a convergência assegura a convergência definitiva da busca para uma solução x*.**

II) IStrongin & Scrgcycv| fornecem um algoritmo paralelo para transformar uma função multidimensional cm uma unidimensional utilizando divisão do espaço de busca tipo Peano e resolver o problema dc otimização em uma única dimensão. Esse algoritmo exige que a função objetivo satisfaça as condições de l.ipschitz para alguma constante I.

Capítulo 7

O algoritmo proposto

// is almost always gambling that enables one to form a fairly clear idea of a manifestation of chance; it is gambling that gave birth to calculus of probability; it is to gambling that this calculus owes its first faltering utterances and its most recent developments; it is gambling that enables us to conceive of this Calculus in the most general way; it is, therefore, gambling that one must strive to understand. But one should understand it in a philosophic sense, free from all vulgar ideas. "

Louis Bachelier

Capítulo 7

O algoritmo proposto

7.1 Introdução

O dicionário Aurelio define algoritmo como o processo de cálculo, ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, regras formais para a obtenção do resultado ou da solução do problema A grande enciclopédia Laroussc cultural define como um conjunto de regras de operação cuja aplicação permite resolver um problema enunciado por meio de um número finito dc operações. [Contei define algoritmo como um complexo e inequívoco conjunto de ações conduzindo a solução de um problema matemático. [MaheyJ define como um mapeamento A associando pontos de um espaço X a subconjuntos de X.

Os métodos numéricos utilizados cm programação matemática em geral, e em particular, em otimização, são algoritmos interativos que procuram gerar uma seqüência de pontos que convergem para a solução do problema

Nesse trabalho supõe-se que o modelo matemático (em termos de expressão analítica) e o problema dc otimização está formulado. A idéia é fornecer um algoritmo que nos leve a solução do problema proposto. C) algoritmo aqui proposto é de natureza probabilística por isso apresentaremos uma síntese da base teórica necessária a compreensão bem como várias referências bibliográficas importantes tais como:[Bath], [Feller], |llammersley & llandscombj e [Loève|, principalmente.

7.2 Base teórica

7.2.1 Erros e aproximação dc números

A palavra erro comporta diferentes significados. Com o sentido de erro absoluto (definido a seguir), com o sentido geral de incerteza (que c uma indicação dc quanto pode ser o erro) ou no sentido dc erro padrão (que c uma maneira particular dc indicar a incerteza). O grande objetivo da teoria de erros é extrair o máximo de informação possível sobre o erro.

lixistem basicamente dois tipos de erros segundo [VuoloJ, são eles:

1) Erros estatísticos;

2) Erros sistemáticos.

Firro estatístico é um erro tal que as n medidas **Xj** de uma determinada grandeza estão distribuídas de maneira aleatória cm torno do valor verdadeiro de x (demonstra-se que se n > **oo,** o valor médio x_m das medidas tende para o valor verdadeiro de x).

lirro sistemático é um erro tal que as n medidas x_i são iguais, mas diferem do valor verdadeiro x dc uma quantidade constante a.

Como o erro sistemático é exatamente o mesmo quando se repete a medida, seu efeito não pode ser avaliado simplesmente repetindo medidas e por isso as incertezas sistemáticas são normalmente mais difíceis dc serem avaliadas que as incertezas estatísticas.

Quando utilizamos um algoritmo para gerar uma seqüência Xj de números convergente para um valor $x_{„„}$ dois conceitos são de muita importância, precisão e acurácia.

Precisão é uma indicação de quanto as medidas são reprodutíveis, sempre está relacionada aos erros estatísticos (quanto menor for o erro estatístico mais preciso é o resultado da medida).

Acurácia é uma palavra usada para descrever quanto o valor calculado está próximo do valor verdadeiro da grandeza (quanto menor for o erro total, tanto maior é a acurácia do resultado). [Vuolo]

afirma que a acurácia está relacionada com erros sistemáticos e estatísticos, outros autores relacionam acurácia unicamente aos erros sistemáticos.

A acurácia dc um número aproximado não depende do número de dígitos significativos, mas do número dc dígitos significativos corretos.

Um número aproximado a é um número que difere muito pouco do número exato A c é utilizado em seu lugar em cálculos. Se sabemos que a < A, então a é chamada uma aproximação menor de A, sc a > A então a c uma aproximação maior dc A. Por exemplo, uma aproximação menor de -*Jl* é 1.41, enquanto 1.42 é uma aproximação maior (1.41 < V2 < 1.42).

O erro absoluto A c dado por

Eq. 7.2.1.1 $A - I A - a |$

Dois casos devem ser evidenciados:

i) Quando o número A é conhecido (nesse caso o erro é prontamente determinado);

ii) Quando o número A não é conhecido, que é o caso mais freqüente, e então o erro absoluto não pode ser obtido pela equação 7.2.1.1. Quando isso ocorre é útil em lugar do erro absoluto teórico desconhecido, utilizar um estimador superior chamado erro absoluto limite, que c definido como qualquer número não inferior ao erro absoluto daquele número. Como podemos facilmente verificar essa definição é muito vaga e temos infinitos números que são candidatos a erro absoluto limite.

Erro relativo 6 de um número aproximado a c a taxa do erro absoluto A pelo módulo do valor exato

A / A »v A
A (A / 0). Assim

Eq. 7.2.1.2 $S = \sim = \succ s l A \equiv I\{Ais$

Como no caso de erro absoluto existem duas situações possíveis: i) A conhecido e ii) A desconhecido, similarmente introduzimos aqui o erro relativo limite, definido como qualquer número não inferior ao erro relativo daquele número. Por definição temos 5 < ô„, isto é,

—r < ÍV , , então
Ml

Eq. 7.2.1.3 A s |A|ô.

Em situações práticas, A a, e na equação 7.2.1.3 temos

Eq. 7.2.1.4 A, |a|ô

O número exato está entre a(1 -ô) e a(I 16), ou seja

Eq. 7.2.1.5 A a(I i ô.)

Uma característica importante dos erros é que normalmente o valor verdadeiro de A é desconhecido, por conseguinte o valor do erro também é desconhecido; por isso o erro A dado pela equação 7.2.1.1 só pode ser conhecido em termos de probabilidades. O erro então tem sua distribuição própria, e é caracterizado por uma função densidade de probabilidade **M**(A)' ll(A-a).

Geralmente o erro A pode ser escrito como uma soma de erros, por exemplo:

A A, i A; i A., i . . . i A

Os erros Aj podem ter distribuições de probabilidades as mais diversas, resultantes, por exemplo de : arredondamentos, critérios dc parada do algoritmo, aproximações implícitas no algoritmo, etc.

[Demidovich] demonstra a relação entre erro relativo dc números aproximados e o número de dígitos corretos, c classifica em cinco tipos as fontes básicas de erro, a saber.

a) Erros envolvidos na formulação do problema;

b) Erros originados de processos infinitos;

c) Erros devidos a parâmetros numéricos (cujos valores podem ser determinados aproximadamente);

d) Erros associados com o sistema dc numeração;

e) Erros devidos a operações envolvendo números aproximados (erros de operação).

Nosso interesse contudo vai além das fontes de geração de erros, queremos acompanhar sua propagação e os mecanismos desta propagação, fazemos isso com a análise abaixo.

1) O erro de uma soma - aqui dois resultados (teoremas) serão enunciados e a prova está disponível cm l Demidovich],

Teorema 7.2.1.1 - 0 erro absoluto de uma soma algébrica de alguns números aproximados não excede a soma dos erros absolutos dos números.

Teorema 7.2.1.2 - Se os termos tem o mesmo sinal, o erro relativo limite dc sua soma não excede o erro relativo limite máximo de qualquer dos termos.

2) O erro de uma diferença - suponha a diferença de dois números aproximados x = X| - x_2. Como já dito anteriormente A_x • A*i * A_{x2}, isto c, o limite do erro absoluto de uma diferença é igual a soma do limite do erro absoluto do diminuendo c o subtraendo.

3) O erro de um produto - também aqui usaremos os seguintes resultados:

'Teorema 7.2.1.3 - O erro relativo de um produto de alguns números aproximados não zero não excede a soma dos erros relativos dos números.

Corolário 7.2. II - O erro relativo limite de um produto é igual a soma dos erros relativos limite dos fatores

$$S_x = \hat{o}xl + 0*2 + \ldots + \hat{o}_{xn}$$

4) O erro de um quociente

Teorema 7.2.1.4 - O erro relativo de um quociente não excede a soma dos erros relativos do dividendo e divisor.

Corolário 7.2.1.2 - Se u ^ x/y, então $\hat{o}_u = \hat{o}_x + \hat{o}_y$.

5) O erro de uma potência - suponha u = x^m (m qualquer número natural), então

Ln u mLn x e.

$$\left|\frac{AM}{}\right| = m\left|X\right|$$

a partir disso temos

$\hat{o}_u = m\hat{O}_u$

ou seja, o limite do erro relativo da m-ésima potência de um número é m vezes o limite do erro relativo do número

6) O erro de uma raiz - suponha $u = tfx$, então $u_m = x$, onde

m

isto é, o erro relativo de uma raiz de grau m é m vezes menor que o erro relativo limite do radicando.

7.2.2 **Probabilidades**

Um evento aleatório é um evento que tem uma chance de ocorrer, e probabilidade é a medida numérica dessa chance. A probabilidade é um número entre 0 e I, inclusive, e quanto mais próximo este número estiver de I mais provável de ocorrer é o evento (o contrário é verdadeiro também). P(A) indica a probabilidade de ocorrência do evento A, P(A+B+C+...) indica a probabilidade de ocorrência de no mínimo um dos eventos A, B, C, P(ABC) indica a ocorrência de todos os eventos A, B, C,... e P(A|B) indica a probabilidade de que o evento A ocorra quando sabemos que B ocorreu (P(A|B) é denominada probabilidade condicional).

Os dois axiomas mais importantes que governam a probabilidade são:

Eq. 7.2.1 P(A+B+C+...) £ P(A) l P(B) t- P(C) + ...

e

Eq. 7 2 2 P(AB) P(A|B)P(B)

Dados dois eventos A e B, se somente um deles pode ocorrer, eles são chamados exclusivos e aí a igualdade da equação 7.2.1 vale. Se P(A|B) = P(A), dizemos que A e B são independentes, isto significa que a ocorrência de B não influencia a ocorrência de A.

O complemento de um evento A, denotado A, é o conjunto de elementos que estão em O mas não pertencem a A.(Í2 denota o universo de eventos e O denota o conjunto vazio).

Uma propriedade importante é:

P(Q) I eP(4>) = 0.

Supondo que um experimento aleatório tem N resultados igualmente prováveis e que um evento A é composto de n resultados, a probabilidade de ocorrência de A é dada por:

P(A) n/N, a probabilidade de $\tilde{A}$ é dada por:

?{2) - 1 - P(A) = I - n/N = (N - n)/N

Uma variável aleatória é uma função que associa um valor numérico com cada evento aleatório de um experimento. Uma variável aleatória pode ser discreta ou contínua. Ela é discreta quando assume um número discreto de valores (finito ou contável), de outra forma é contínua.

Uma distribuição discreta de probabilidade é a associação de uma variável aleatória discreta com uma função probabilidade.

Uma propriedade importante é a seguinte: seja X uma variável aleatória discreta que toma valores xi, x_2,..., x_n e seja P uma função probabilidade com p(xj) = P(X^Xj). Então:

i) P(xi)X), Vxi,i= 1,2,... ,n

/ I

Outra função útil é a função distribuição cumulativa, esta função normalmente denotada F(x), mede a probabilidade que a variável aleatória X assuma um valor menor ou igual a x, isto é, F(x) = P(X < x).

A seguinte propriedade é muito importante: seja X uma variável aleatória discreta, e seja F a função distribuição cumulativa associada, com F(x) = P(X < x). Então:

i) 0<F(x)< 1, -oo<x<oo

ii) Sc xi < x_2, então F(xi) < F(x_2), V$x_к$$x_2$ (F é não decrescente)

pesquisadores c, no momento, o nosso próprio.

O capítulo 7 apresenta o algoritmo proposto, com os conceitos básicos necessários à sua compreensão, apresenta a classe de funções em que se aplica c discute aspectos ligados à convergência e às vantagens dc sua utilização.

O capítulo 8 apresenta um dicionário de funções dc teste utilizadas, a forma como o algoritmo foi implementado para estas funções e os resultados obtidos.

0 capítulo 9 traz algumas conclusões sobre as técnicas envolvidas nesse trabalho bem como algumas sugestões dc temas e assuntos para pesquisas futuras.

Ao final apresentamos um apêndice com as definições c resultados que julgamos dc maior interesse para quem fizer uso futuro deste nosso trabalho.

1.2 Deliuição do problema

1.2.1 Q problema vinculado

0 problema geral dc otimização é encontrar extremo dc funções com restrições de igualdade/desigualdade ou não. Enunciaremos o problema tendo em mente que a função objetivo e as restrições são funções reais não lineares dc variáveis vetoriais

Sejam as funções f: **R" -> R,** g: **R" -> R** c h: **R" > R**^{p}**,** sendo f, g, h continuamente diferenciáveis, encontrar, se existir, um vetor **x" = (xi,** x_2, x_n) no conjunto **V** de pontos viáveis, **V** definido como o conjunto dc todos os pontos **x c R"** tais que g(x) > b e h(x) = 0, tal que **Vx** c V, *í[x')* < f(x).

Esse problema pode ser rcescrito de forma mais compacta como:

Minimize *f[x)*

V

Sujeito a g**(x)** > b

iii) lim/••(*)- /••(*>) I, Jim/"(*) = /«(-QO)-0

Por definição a esperança da função g(x) = X', r = 1, 2, ... , é chamada o r-ésimo momento da variável aleatória X relativo a zero e é denotado por | i . Aplicando a definição de esperança podemos calcular o r-ésimo momento da variável X acima relativamente a zero como

u_r -]T] *,'/'(*,) se X for discreta, ou como

u_r - **J** *x'/(x)Jx* se X for continua.

Obs : *Daqui por diante não mais nos referiremos as variáveis aleatórias continuas pois na construção do algoritmo é utilizado apenas a noção de variáveis aleatórias discretas, islo no entanto não nos leva a nenhuma perda de generalidade.*

O grande interesse relativo aos momentos de uma variável aleatória se concentra nos momentos relativos a média da distribuição, isso nos leva a

Mn. £<^x< O'/'(*,)

I

Os momentos mais importantes são o primeiro (média), o segundo (variância), o terceiro (assimetria) e o quarto (curtosc), vamos então deduzir suas expressões analíticas bem como dar uma interpretação a eles

O primeiro momento e o valor esperado ou a média da distribuição que aqui será denotado por x_m, e é dado por:

Eq. 7.2.1 x_m £>, $x_m)l'(x,)$

i

0 segundo momento é a variância da distribuição. E um indicador da dispersão amostrai em torno da média, é denotada por Var(x) e é dada por:

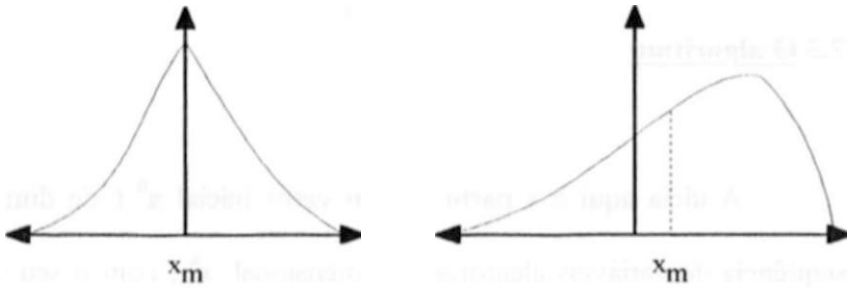
Eq. 7.2.2 *lar(x)* ***Y.****(x, x_m)'l'(x,)*

I

O terceiro momento representa a assimetria da distribuição e será denotado por A(x) e é dada por:

Eq. 7.2.3 $A(x) = (x_{\hat{e}} \quad x_{m})^{3} />(*,)$

A assimetria nos fornece uma indicação do formato da distribuição. Se ela for positiva a distribuição é assimétrica e positivamente inclinada (o pico da distribuição está inclinado para a esquerda), se é zero a distribuição é simétrica e se for negativa a distribuição é assimétrica e negativamente inclinada. A figura 7.2.1 abaixo ilustra estas configurações.



A (x) > ( )

figura 7.2.1 - Comportamento do terceiro momento (assimetria).

O quarto momento é a medida de curtose, ou seja, é um indicador de o quanto a distribuição é mais ou menos pontiaguda e é calculado como mostrado abaixo. Para esclarecer mais este conceito, observe a ligura 7.2 2 abaixo

Eq 7.2.4 $* J > (* ,)$

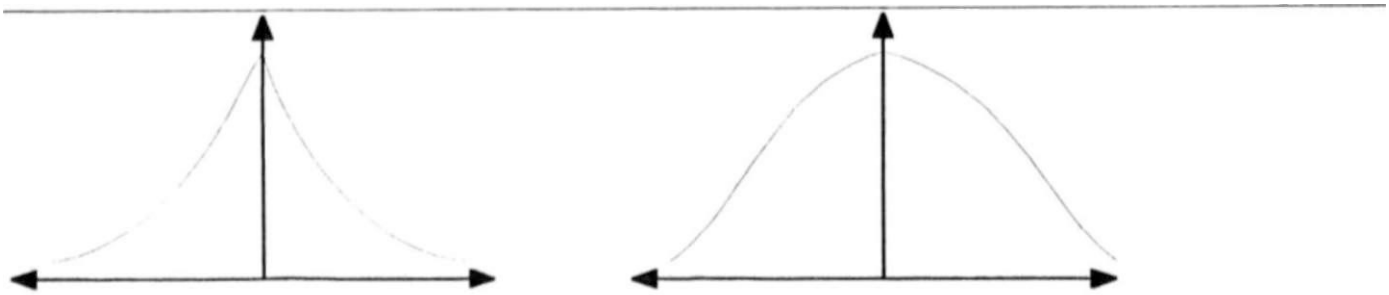


Figura 7.2.2 - O quarto momento

7.3 O algoritmo

A ideia aqui c a partir de um vetor inicial **x° (** de dimensão n ainda desconhecida), gerar uma seqüência de variáveis aleatórias n-dimcnsional, $\mathbf{x}^{k}$, com o seu valor médio convergindo para o máximo maximorum da função *\\x).*

Sem perda de generalidade c com o intuito dc tornar clara c inteligível nossa descrição do funcionamento do algoritmo descreveremos o caso unidimensional onde estaremos procurando um máximo dc fi» que seja máximo em todo o domínio de definição de f\x).

0 primeiro passo nessa caminhada é a interpretação da função *f{x)* a ser maximizada como uma função densidade de probabilidades da variável x. Isso nos permite obter os momentos dessa variável (média, desvio-padrão, assimetria e curtose), e utilizar estas informações para alcançar nosso objetivo.

O ponto máximo maximorum procurado é a moda da distribuição, então trabalharemos no sentido de conduzir o algoritmo a convergir para esse ponto

Para aplicação do algoritmo suponha-se que são dados: a expressão analítica da função *\{x)* a ser maximizada, um intervalo de busca | Limlnux] c a precisão desejada para o valor de x* ou **fiV)** ou ambos.

0 intervalo de busca deve ser finito e a precisão é utilizada para compor o critério de parada, isto é, podemos nos dar por satisfeitos com o resultado quando

i) A 11

|./(v,^") /(*,*)!<*

iü) **<s c |/(r*⁺') /(*,*)!<***

O objetivo é: dado um vetor inicial **(xi,** x_2,..., x_n) produzirmos transformações tais que como conseqüência obtenhamos a resposta para o problema de otimização global, isto é, o vetor solução (x***,fvx*));** a figura 7.3.**1** abaixo ilustra esse esquema lógico.

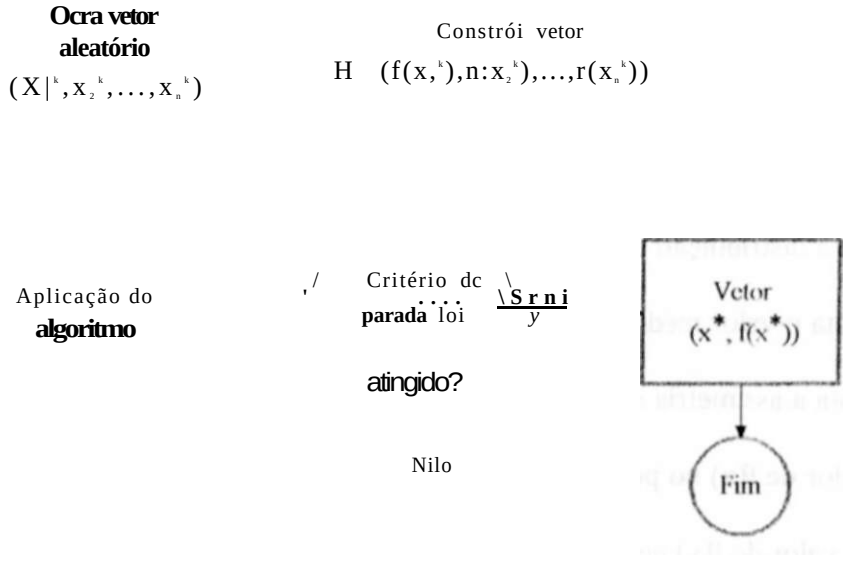


figura 7.3.1 - Esquema lógico do algoritmo.

0 algoritmo é composto das expressões dadas pelas equações 7.3.**1** a 7.3.7 abaixo

i*. 7.3.1 *-*S*Wr'-K**

Eq. 7.3.2 $\sum_{i=1}^{n} (\wedge) \wedge \wedge$ **i l L**

Eq. 7.3.3 *Cianho do Passo y\c **

, 5 '' 7 5 4 **[/«r(*i*)-/(*?)j**

Eq. 7.3.5 *Sinal* $r\{4\ ^{\circ k})-f(*i)$ **(4 m**

Eq. **7.3.6** Diferença - $x,.^{k} - a^{k} - x;^{k}$

Eq **7.3.7** $\mathbf{X\,i}^{k}$ ' ' x,^{k} I Ganho do Passo Curare. Sinal . Diferença

Onde i é o índice indicativo da componente do vetor aleatório

k é o índice indicativo da iteração

Xj^{k} c o valor da i-ésima componente na k-ésima iteração do vetor aleatório

0» é a função a ser otimizada

$l'^{k}(xj^{k})$ é a distribuição gerada a partir de fi»

x/ denota o valor médio na k-ésima iteração da distribuição $l^{:k}(xj^{k})$

a^{k} denota a assimetria da distribuição na k-ésima iteração $F^{k}(xj^{k})$

H>j^{k}**)** valor dc no ponto x^{k}

fin.ix**(xj)** valor dc u» em **Xj**^{k} que torna fi» máxima na k-ésima iteração

ôi, **Ô2, 51,** Yi, **Y2,** são parâmetros para simulação do algoritmo

7.4 Descrevendo o funcionamento do algoritmo

Desde o início desse nosso trabalho o objetivo tem sido tornar sua leitura clara e de fácil compreensão; com esse intuito então iremos fragmentar as equações **7.3.1** a **7.3.7** de forma a tornar a análise e a descrição o mais detalhada possível (solicitamos algum esforço por parte do leitor no sentido de compreender que nos casos de mais de uma variável o algoritmo se comporta como se estivéssemos otimizando separadamente na direção de cada variável).

O algoritmo funciona a partir do ponto em que temos uma expressão analítica de uma função a ser otimizada, uma dada precisão c uma nuvem de pontos gerada de forma aleatória formando o vetor aleatório inicial (xi°, x_2°, ... , x„°) - onde cada Xj^k **é um ponto da nuvem inicialmente gerada - e a partir desse vetor encontrar (x , *í{x*)) que resolve o problema proposto.**

Como questão inicial a ser respondida colocamos o seguinte: como o vetor inicial (xi, x_2,..., x_n) deve ser gerado? optamos por gera-lo de forma aleatória com uma distribuição uniforme no intervalo de interesse (isso é um elemento que contribui para tornar o algoritmo paralelo). A função fi» deverá ser aproximada por seus valores calculados no intervalo de busca dado em um número finito m de iterações realizadas sobre esse vetor aleatório n-dimensional.

Dado que o vetor aleatório inicial foi gerado prosseguimos construindo um outro vetor aleatório (f{xi"), fíx$_2$°), ... , fTx„^{n})), e de suas coordenadas escolhemos um ponto candidato a máximo global $f_{,_{na}}x$**(xi) que será utilizado nas equações 7.3.1 c 7.3.4.** $f_{ma}x$**(xj°) c nosso primeiro candidato a máximo maximorum.**

0 passo seguinte é gerar a distribuição P(XJ)**; pela expressão 7.3.1 verificamos que para** $\hat{O}_2 = 1$ **=>** $l^{:k}$**(xj^k) tem uma distribuição normal centrada cm** $x_{max}{}^k$ **(onde** $x_{max}{}^k$ **c o valor de** Xj^K **que proporciona o maior valor de fm«x(Xj^k)), e que quanto mais próximo fi>j^k) estiver de** $f_{ma}.x$**(xi^k) maior será o valor de F^k(x;^{k}) para aquele** Xj^k**, decorre daí então que os valores próximos do ponto de sintonia (**f_{max}**(xj^k)) terão probabilidades altas e o contrário ocorrerá com os pontos que estiverem distantes.**

0 fator $e^{\acute{o}}$***^{k} tem a função de tornar o valor de F^k(xj^k) dependente do número de iterações de tal sorte que quando k for pequeno, isto é, no início do processo, a distribuição vai depender basicamente da diferença de** ft>j^k) **e** $f_{„_{ux}}$**(xj^k) que no início terá uma faixa de variação muito grande (os** Xj° **estão distribuídos uniformemente), na medida em que k cresce as diferenças entre os f{xj) diminuem consideravelmente pois o algoritmo leva os** Xj^k **para as regiões tais que xj.^{k} -> x* e então** $e^{\acute{o}}$**^ garante a sintonia em torno de** f_{max}**(xj).**

Muito bem, agora temos a distribuição F (x;), então o nosso próximo passo é calcular o valor médio X|.^{k} que queremos que venha a convergir para x\ atribuímos inicialmente uma probabilidade a cada Xj^{k} dada

por III** I c então calculamos a média pela equação 7.3.2. Ksse valor médio nos permite

/ i

calcular a assimetria da distribuição.

Nesse ponto um problema que surge é a instabilidade de f$_{m}$a.$_{x}$(x;^{k}) e aí introduzimos um termo que denominamos curare (nome comum a substâncias de origem vegetal que possuem uma potente ação paralizante, em medicina a ação fisiológica dos curares sobre os músculos manifesta-se por vários estágios - hipotonia, atonia - até chegar a paralisia, que se acompanha de inexcitabilidade elétrica As substâncias curarizantes são usadas em anestesiologia para obter um bom relaxamento muscular, particularmente útil em cirurgia abdominal) dado pela equação 7.3.4. Na equação 7.3.7 verificamos que o valor Xj$^{k''}$ é calculado a partir de Xj^{k} adicionando-se a esse termo um produto de quatro termos um dos quais o curare. Analisando a equação 7.3.4 constatamos facilmente que quando *f[x-)* = f$_{max}$**(x$_{i}^{k}$)** o curare vale zero e Xj$^{k''1}$ = Xj^{k} e que se fl**[xj^{k})** / f$_{n,ax}$**(x^{k})** => x;$^{k''1}$ / Xj\ cm outras palavras, o valor de x que gera f„u$_{x}$**(xi^{k})** não muda até que surja um novo valor máximo no vetor, isto é, ^«(x,^{1}"') > f$_{max}$(xj^{k}) e aí a distribuição F^{k}(xj^{k}) muda bruscamente seu centro (passando a ser o novo valor x$_{max}^{k1}$).

Novamente vamos verificar o que já temos

1) Temos o vetor **(xi^{k},** x$_{2}$\ ..., **x„)**

2) Temos o vetor (u>,^{k}), fi>$_{2}^{k}$)____n>$_{n}^{k}$))

3) Geramos F*(xi^{k})

4) Calculamos X|.^{k}

5) Calculamos a^{k}

Verificamos então que não sabemos, ainda, para que lado da distribuição iremos andar com cada ponto Xj\ essa informação será fornecida pelo sinal através da equação 7.3.5.

0 termo diferença nos diz o quanto iremos andar na direção fornecida pelo sinal. Nesse ponto é preciso que tenhamos cm mente que, no começo (k pequeno), ainda não conhecemos nada ou quase nada sobre nossa função c, por precaução, gostaríamos de andar devagar. Isso é necessário e deve ser feito com o claro objetivo de minimizar a probabilidade de perder algum máximo ao longo da nossa caminhada. E a partir do momento em que já estamos à vontade quanto a nossa função (k grande) tornar os passos mais largos na direção de x* dado por cada **Xj^k,** essa função é realizada pelo Ganho Do Passo. Finalmente deveremos entender o papel dos fatores **61,** $\hat{o}_2$, ôj, **Yi** e **Y2.** Estes fatores foram introduzidos como forma de facilitar o processo de simulação do funcionamento do algoritmo, isto é, eles nos permitem verificar o comportamento do algoritmo para diversos valores que a eles forem atribuídos. O parâmetro ***yi*** tem a clara função de evitar ***overflow*** ($f_{max}(xj^k)$ - í{xj^k) = 0), um estudo detalhado para determinação dos valores ótimos desses parâmetros é sugerido no capítulo 9.

A aplicação do algoritmo continua até que os critérios de parada sejam atingidos.

Um cuidado adicional a ser verificado é no sentido de evitar que os novos pontos iterados sejam colocados fora do intervalo dc buscas, para isso então a partir da segunda iteração devemos retornar o valor l_{min} se $x_j^{k+1} < l_{min}$ c l_{max} se $Xj^{k+1} > l_{max}$.

Um aspecto importante a ser observado é que o tamanho da amostra (comprimento do vetor aleatório) n não é utilizado de forma explícita no algoritmo induzindo a falsa idéia de que o algoritmo converge para qualquer valor de n. Isso absolutamente não é verdade e intuitivamente isso não é aceitável Afirmamos que n influencia diretamente na performance do algoritmo, o que queremos esclarecer é que o valor absoluto de n não é tão relevante nessa discussão quanto a densidade de pontos no intervalo de interesse, isto é, $\frac{n}{'max \quad «min}$, esse fator combinado com o desvio-padrão amostrai na geração do vetor inicial devem ser bastante considerados, no estágio atual de desenvolvimento do algoritmo não entramos

nesse detalhe, contudo esse é um aspecto que exige estudos posteriores. Nosso objetivo imediato é mostrar que o algoritmo funciona

No capitulo 8 a seguir veremos o comportamento do algoritmo em funcionamento c poderemos analisar seus resultados mais pormenorizadamente.

Para o caso bidimensional:

Eq. 7.3.8 F*(*S.4)-#"

Eq. 7.3.9 *í444)-íAl*

Eq. 7.3.10

Eq.7.3.11 Í V ^ . U ^ , *) W (^ . 4)

Eq. 7 3.10 *(lanho do posso y^c*

Eq.7.3.11 *Sinal*

Eq. 7.3.12 *Sinal2* -

Eq. 7.3.13 *Diferença*

Eq. 7.3.14 *Diferenço 2*

Para o caso geral introduziremos o índice m para indicar a variável e temos:

Eq. 7 3 15 ***l'(4ü¹ mi)=***

Eq- 7.3.16 $$\mathbf{X}\ *\ ,\ ,\ ,\ S\ *>(**,)-\ Z\ V-\ \dot{V}\ mj\ \ mj$$

Eq. 7.3.17 *Curare* =

Eq. 7.3.18 Í f«wfir> ífr> /?í7.v.w /1 Í' ^

Eq. 7.3.19 *Sinal* m - — y -

Eq. 7.3.20 ***Diferença*** m 13c ... - **A**^' - jr* .

7.5 Convergência

O algoritmo gera algumas seqüências tais como:

\fLc\ {A\ (/(•))• { x L , }**

A função $F^k(x^k)$ tem um valor máximo quando $x_i^k = x^*$, onde x_{max}^k é o valor de x_i^k (i = 1, ... ,n) responsável pelo valor f_{max}^k. Noutras palavras *ffam)* = f_{nux}^k, como já dito anteriormente a distribuição de probabilidade gerada a partir da $F^k(x^k)$ será uma guassiana (se $\hat{o}_2 = 1$), cuja variância diminui a medida que

h**(x)** 0 , ou ainda como

(Prob. 1.2.1.1) Minimize f**(x)**

x

Sujeito a gj**(x)** > bj, i 1 , 2, 3, m

hj**(x)**-0 , j= 1,2,3, p

onde

f**(x)** é a função objetivo

g**(x)** representa as restrições de desigualdade

h**(x)** representa as restrições de igualdade

V é o conjunto de soluções viáveis

gi : **R**ⁿ **- > R** é a i - ésima componente de g

hj! Rⁿ **—> R** é a j - ésima componente de h

m é o número de restrições de desigualdade

p é **o** número de restrições de igualdade

[n - (m + p)] > 0 é o número dc graus dc liberdade

O vetor coluna \ que resolve o problema 1.2.1.1 acima é denominado ponto de ótimo da função objetivo c **o** correspondente valor dc f**(x")** é denominado o valor ótimo da função objetivo, **o** par dc valores **(x**",f**(x"))** formam uma solução ótima do problema. Uma solução globalmente ótima representa o menor valor de f(x) entre todos os possíveis valores da função, enquanto que uma solução localmente ótima representa o menor valor de f(x) apenas na vizinhança dc algum vetor x.

1.2.2 O problema não vinculado

Seja f: **R**ⁿ **—> R,** continuamente diferenciável, encontrar, se existir, um vetor **x"** e- **R**ⁿ**,** tal que **Vx** c **R"**

k aumenta , c estará sempre centrada em x_{max}^{k}. O valor médio dessa distribuição, $\mathbf{X}|.^{k}$, convergirá para $x^{\wedge}x$ à medida que k aumenta (isso vem do fato de f(xp) ~> f_{max} <^> **X**F -* x_{max}).

O curare garante que a seqüência $\{f_{max}^{k}\}$ é monotônica não decrescente ($f_{max}^{k} < f_{max}^{k}$", V k). Excetuando-se os casos de descontinuidade de segunda espécie, como o domínio de f é limitado, seu contradominio também é limitado. Então existe / tal que \\x) < / , V x e D (onde I) é o domínio de \\x)). Como toda seqüência monotônica limitada é convergente , conclui-se que $\{f_{max}^{k}\}$ é convergente. A questão a partir daqui é saber para que valor a seqüência $\{f_{n}u_{x}^{k}\}$ converge. Como dito anteriormente / é o máximo maximorum (fi» </ , V x c D) então ***f*** **=*f*** como $f_{max}(xj^{k})$ > / = ***f*** o valor n>,.^{k}) -> / = f* .

7.6 Classe de funções em que se aplica

funções mensuráveis.

7.7 Conclusões

1) A grande vantagem desse algoritmo é o paralelismo inerente a sua construção, o que nos coloca frente a uma expectativa muito positiva em relação a sua performance quando implementado em uma máquina que possa tirar proveito das suas características probabilística e paralela.

2) A minimização de uma função *i\x)* qualquer pode ser realizada maximizando - n » , desse modo toda discussão anterior continua válida;

3) A minimização também pode ser realizada com modificações apropriadas das equações 7.3.1 a 7.3.6 no caso unidimensional, 7 3 8 a 7 3 12 no caso bidimensional e 7.3.15 a 7.3.20 no caso geral. Esse aspecto é importante c deve ser ressaltado porque ele nos leva aos pontos de sela que o algoritmo calcula apenas maximizando cm uma direção e minimizando na outra (caso bidimensional).

4) Uma conjectura em relação ao valor inicial de n nos leva a especular quanto a sua relação com o teorema da amostragem.

5) Observa-se que o algoritmo converge muito rapidamente para a região ótima, isso sugere uma utilização combinada desse algoritmo com um algoritmo de otimização local eficiente para aumento da velocidade com **que** se alcança a precisão requerida.

6) 0 problema de propagação de erros fica minimizado nesse caso pois o algoritmo funciona por integração.

7) Nenhuma exigência quanto a aspectos topológicos ou algébricos foi feita para aplicação do algoritmo;

8) ("orno visto nos conjuntos dc equações 7.3.1 a 7.3.7, 7.3.8 a 7.3.14 e 7.3.15 a 7.3.20 a complexidade do algoritmo de um fator inferior a n com o aumento da dimensão do espaço;

9) A implementação computacional do algoritmo é bastante simples;

10) Não utilizamos gradiente c matriz I lessiana o que evita instabilidade numérica cm torno do ponto dc ótimo

Capítulo 8

Implementação computacional

"A teoria do azar consiste em reduzir todos os acontecimentos do mesmo gênero a um certo mimem de casos igualmente possíveis, ou seja, tais que estamos igualmente inseguros sobre sua existência, e em determinar o número de casos favoráveis ao acontecimento cuja probabilidade é buscada. A relação deste número com o de todos os casos jwssíveis é a medida dessa probabilidade, que não é assim mais que uma fração cujo numerador éode casos favoráveis e cujo denominador é o número de todos os casos possíveis."

Ensaio Filosófico Sobre as Probabilidades
Pierre Simon Uiplace

Capítulo 8

Implementação computacional

8.1 Introdução

Nossa ideia quando iniciamos esse trabalho era empreender um esforço adicional e implementar o algoritmo com a ajuda de uma linguagem compilada, procurando utilizar as técnicas mais apropriadas de programação. Isso no entanto traria dois inconvenientes que procuramos evitar: primeiro, qualquer que fosse o nível de sofisticação c elegância que conseguíssemos imprimir a implementação computacional estaríamos utilizando uma máquina seqüencial para isso o que de certa forma iria mascarar os resultados que estávamos querendo apresentar. Segundo isso não daria a idéia exata da simplicidade computacional e da força do algoritmo Nosso ideal então passou a ser implementa-lo em uma calculadora de bolso de forma que o meio empregado justificasse o fim (demonstrar a sua simplicidade e performance). Contudo resolvemos utilizar uma ferramenta que está ao alcance da maioria das pessoas diretamente interessadas no problema que procuramos solucionar, uma planilha eletrônica. Entre todas as planilhas existentes optamos pela ***Quatro Pro*** da Borland cm sua versão 6.0. Essa decisão foi influenciada principalmente por conveniência pessoal

Após a escolha da ferramenta computacional procuramos selecionar algumas funções de teste com o objetivo dc medir a performance e o desempenho do algoritmo. Escolhemos as funções apresentadas na seção 8.2.

Essas funções não foram escolhidas ao acaso, procuramos selecionar funções que apresentassem dificuldades patológicas dc soluções (funções de I a 6) e outras que além de apresentar estas mesmas características fossem de uso clássico c generalizado nesse tipo de literatura funções de 7 a 20 dentre elas as funções de Koscnbrock e dc Goldstcin-Price.

8.2 Funções dc teste utilizadas

Função universal dc teste unidimensional

Função I

1)1	**c l**	Mi 1	**1)2**	**c2**	I H **2**	**b3**	**1)4**
28	**10**	1	**-12**	**1**	1	**0**	**10**
b5	**c5**	**m5**	**n5**	**d5**	**b6**	**c6**	**d6**
10	**5**	**5**	**4**	**-5**	**10**	**0.5**	**0.1**

Função 2

hl	**11**	> i 1	1)2	c2	in2	1)3	1)4
10	**c**	1	-12	**1**	1	0	10
b5	**c5**	**m5**	**n5**	**d5**	**b6**	**c6**	**dó**
10	**5**	**5**	**4**	**-5**	10	0.1	0.1

Função 3

III	1 1	Mil	1»^	< 2	in 2	|».1	l>|
2X	10	1	-12	1	1	0	10
l>5	c3	m5	n5	d5	b6	c6	d6
10	5	5	4	-5	10	0.5	0.1

l limão 4

M	t 1	ml	h i	(2	in2	IÜ	1)4
10	5	1	-12	1	1	0.2	10
1)5	c5	m5	n5	d5	1)6	c6	d6
10	5	5	4	-5	10	0.5	0.1

Função 5

III		m l	l>2	(2	Ml2	l u	h i
10	5	1	-12	1	1	0	10
1)5	c5	ni5	n5	d5	1)6	c6	d6
10	5	5	4	-5	10	0 5	0.1

Função 6

	. I	Ml I			III?.	I1j	
30	**5**		**-12**			**0.25**	**10**
b5	**c5**	**ni5**	**n5**	**d 5**	**b6**	**c6**	**dó**
10	**5**		**4**	**-5**	**0.35**	**2**	**2**

Função 7

Maximize l{x) - x Sen(l/x)

X r [0.05,0 5]

x' 0 223

11 lassoun J pai; 418 e 423

Função 8

Maximize *I\x)* **x - x**^{7}

Solução X* - 0.5 - > ílx*) - 0.25

Função 9 **(Kosenbrock)**

$$\text{Minimizar} \quad f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

Proposta cm |llimmclblau| pag. **394 problem 2**

Ponto inicial: x" |-I 2 ; I |

\\ x") 24 20

Solução x I I I l'

fiV) o

Função 10

Minimizar *f(x)* = **4**(jC, - **5)**² 4- *(x₂* - *ôf*

Ponlo inicial:x" |8 9|'

Hx°)«45

Solução x [5 6J

MV) - 0

Problem 25 pag. 426 de |Himmelblau]

Função 11

Minimize $f(x_1, X_2)$ = (jC,X₂)²(I -Jt,)^I -JC, - X_2(\ ~^|)⁵J

Ponto inicial: x° | - l 2 l | '

M>°) 26.656

Solução: x* |l - l ' ou|0-|' ou|-()|' u > V o

Problem 27 pag 427 de 11 limmelblau|

Função l2((;ol(lslciii-Piicc)

minimize

I I • (r, i x_2 **0 (I I f 2 • (>**2 3*1) J- (I -.ijr) (I»-32x! +I2*f I 4**KJT$_2$ **- líix|**JT$_2$ **+ 27.t**

$-2 < x_j < 2$

I x . W l |0,-l|

l(x,',x$_2$*)-3

[Stuckman & Easomj pag. 1024

Função 13

Minimize $f(x_1,x_2) = (xf + X_2 - \mathbf{I}\ \mathbf{I})^2 + (jC, + x\ - lf$

Ponto inicial: x" |l l | '

l(x") 106

Solução: x* |3.58443 -1.84813]' c [3 2J^T

Hx*) 0

Obs Todos os **testes** a partir **d e x** 11 11' produzem |3 2|'

Problem 28 pag 428 |l limmclblauj

Função 14

Maximize

$f(x_1, x_2, X),x_4) = (x, + I (bf_2)^2 + 5(JC_3 - *_4)^2 + (*_2 - 2*3)^4 + I ü(*, - x_4)^4$

Ponto inicial: x° « [3-1 O l f x ° - | l 1 I 1]

ltx°) 215 IIx°) 125

Solução: x' [OOOOf

uV) * 0

Problem 26 pag. 427 [Himmelblau]

Função 15

Minimize /(.v|.x_2) = OJ**JT*** I *0JÍ* -C2**M(2XI)]+XJ**

Alguns mínimos locais

I mínimo global

(x,\$x_2$V(0,0)

Função 16

Minimizar /"(*!,*2)~í^x 2 ~^\) ^0 ^x\)

(X|,X_2) = (I D

ft>,\$x_2$Vo

|l limmelblauj pag 195

Função 17

Minimizar /(*|,.t_2) (*2 ^1 ^m *Í* ^/ ^x\f

, fiY) < fi».

Esse problema também pode ser rcescrito de forma compacta como:

(Prob. 1.2.1.2) Minimize f(x)

E imediata a constatação que o problema 1.2.1.2 é um caso particular do problema 1.2.1.1 com m

P 0.

Alguns casos particulares importantes são:

i) quando as variáveis estão restritas a valores inteiros - programação inteira;

ii) quando as restrições envolvem o parâmetro tempo - programação dinâmica;

iii) quando as restrições são equações diferenciais - controle ótimo;

iv) quando temos valores aleatórios nos parâmetros da matriz dc projeto - programação probabilística;

v) quando a função objetivo e as restrições são lineares - programação linear;

vi) quando a função objetivo é quadrática c as restrições são lineares - programação quadrática.

1.3 Objetivo do trabalho

Esse nosso trabalho insere-se em um contexto maior denominado projeto TALUS proposto cm 1986 pelo professor Fernando Menezes Campello dc Souza, do Departamento dc Eletrônica c Sistemas da UFPE, cujo objetivo é conceber, projetar e construir um computador com capacidade para resolver uma grande classe dc problemas matemáticos de interesse, que seja rápido (paralelo), inerentemente tolerante a falhas, dc grande facilidade no desenvolvimento e implementação dc software e de baixo custo. A idéia básica do projeto é a utilização de técnicas c algoritmos probabilísticos c a sua implementação num *hardware* cuja estrutura comporte geradores dc números aleatórios c microprocessadores operando em

(x . \ x , V (l I) ~~ —

fi>,\x₂*) - 0

[Himmelblau | pag 195

Função 18

Minimizar $f(x_1x_2)$ = **luojjrç - x f í + (l - * i)²**

(X|\x₂*) = 0 I)

fi>,\x₂*) = Ö

[Himmelblau | pag 195

Função 19

Minimizar **/(*, ,x_2) =[1.5-*,(l-** x_2)]² ^225 **x,(l-^|)J²** +[*2.625-x^l-x^jf*

(x,\x₂>= (3,1/2)

fiV,x₂*) - 0

|Himmelblau) pag 195

Função 20

Minimizar **/(jri,jr₂) = 4xf + 4jrf -4*,x₂ 12*₂**

(xr, x₂*) = (i 2)

n>,\ x/) = -i2

11 limmeiblau] pag 195

8.3 A solução do problema de programação linear

E justo esperar que um algoritmo que resolve o problema de programação global resolva um simples problema de programação linear. Simples no sentido que um algoritmo de otimização global consegue distinguir entre os vários extremos locais de uma determinada função encontrando os extremos globais em funções de dificuldades patológicas, logo nada mais justo que esperemos uma solução para um problema onde temos um conjunto viável convexo e todas as funções envolvidas lineares.

Vamos então resolver o seguinte problema de programação linear:

Maximize $f(X|,X2) = 2xi + 5x_2$

Sujeito á $X| \; i \; 4x_2 < 24$

$3xj \bullet x_2 < 2l$

$x, \; t \; x_2 < 9$

$Xj > 0 \quad i = 1,2.$

Evidentemente o objetivo é pura e simplesmente ilustrar esse aspecto, isto é, resolver problemas de programação linear com o mesmo algoritmo utilizado em otimização global, por isso o problema escolhido é simples porém extremamente didático. Para que possamos aplicar o algoritmo de otimização proposto nas equações 7.3.1 a 7.3.14 precisamos promover uma transformação do problema acima. Como no caso dos multiplicadores de Lagrange a idéia é incorporar as restrições à função objetivo e só então otimizar a função transformada, faremos isso utilizando o seguinte esquema:

Eq. 8.3.1 $a(xi,*_2) = /(*1.*2) —$ \= 1.2.

onde gi(xi,x_2) são as restrições e a é um número (não discutiremos aqui suas propriedades). Então no caso do problema acima temos

A solução do problema acima está em ($xi\backslash x_2^*$) = **(4** 5) isso nos dá f($xi\backslash x_2^*$) - 33. O termo que está sendo subtraído da função objetivo funciona como uma função penalidade, isto é, quando o valor dentro do parênteses é positivo (fora do conjunto viável) o valor do termo específico fica muito grande rapidamente; ao contrário se o termo entre parênteses é negativo o valor do termo específico fica muito pequeno e não influi no valor de f{xi^*,x_2^*). Além disso quando estamos maximizando G($x/,x_2^*$) é côncava.

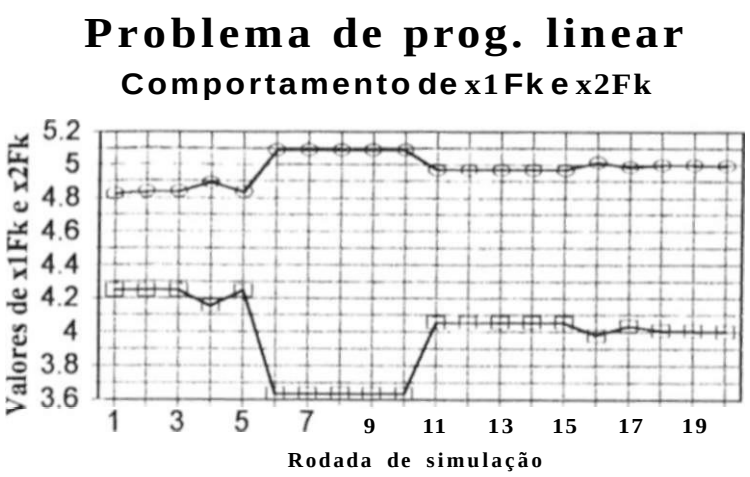


Gráfico 8.3.1 - Valores iterados para **xi** e **x_2.**

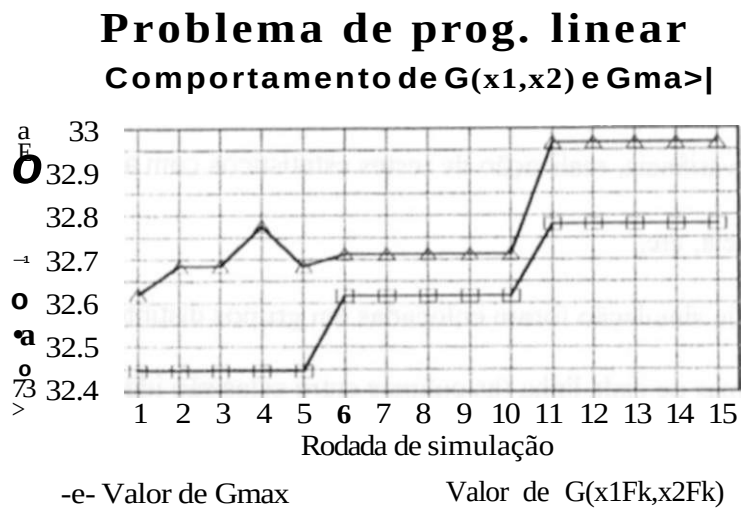


Gráfico 8.3.2 - Valores iterados para $G(x_1,x_2)$ e de G_{max}*.

Os gráficos **8.3** I e **8.3 2** acima mostram o resullado obtido com a aplicação do algoritmo a $G(x_1,x_2)$.

8.4 Resultados obtidos

Os resultados obtidos com a aplicação do algoritmo as funções de teste e com o problema de programação linear estão listados conveniente c detalhadamente nos anexos de B a X e no anexo Z apresentamos o desempenho do algoritmo relativamente a função de Rosenbrock com espaços de busca de tamanhos diferentes.

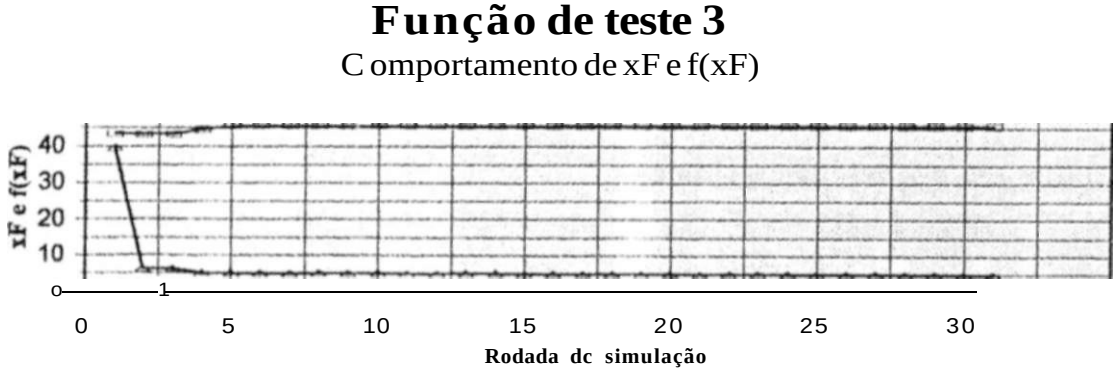
8.5 Comentários sobre os resultados obtidos

Iremos agora comentar e utilizar os resultados obtidos com a aplicação do algoritmo em nosso conjunto de funções de teste para ilustrar e mostrar a importância de tais resultados. O que devemos enfatizar inicialmente é que:

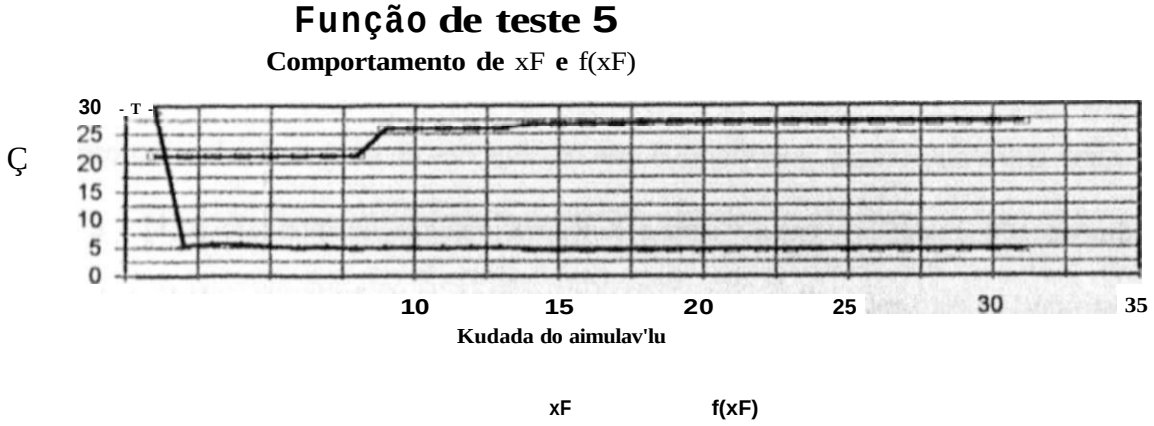
i) Na geração da nuvem inicial de pontos aleatórios nenhum cuidado adicional foi tomado relativamente ao condicionamento dos números no que tange, por exemplo, a utilização de variáveis antitéticas para diminuir a variância, realização de testes estatísticos com a amostra de forma a garantir uniformidade, independência, etc;

ii) ('orno as rodadas de simulação foram colocadas em grupos distintos de linhas, listamos as primeiras e as últimas colunas de cada linha (as colunas entre estas são idênticas), tanto com a grade da planilha - de modo a ilustrar o trabalho como um todo - quanto com o conteúdo das fórmulas ali contidas. Listamos também a capa da planilha onde temos os valores dos parâmetros ôi, Ô2, 63, *y\, j2* **e** Y3, **o espaço de busca e o erro utilizado para prevenir a ocorrência de** *overflow*.

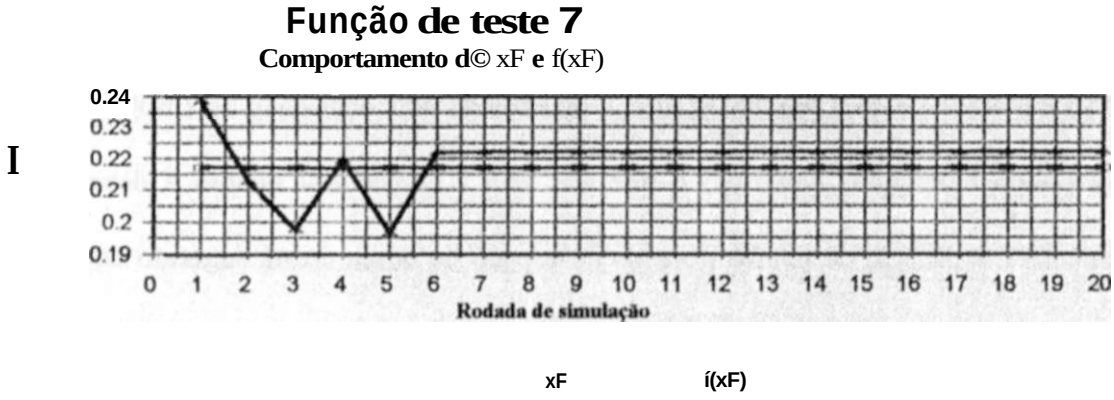
Iniciaremos observando que nas funções unidimensionais (funções de I a 8) a regiSo do ponto de máximo maximorum é alcançada em poucas iterações (frequentemente consegue-se chegar a uma vizinhança muito próxima deste ponto na primeira iteração) como ilustrado pelos gráficos 8.5.**1 (a), (b) e (c) referentes as funções de teste** 3, 5 **e** 7.



(a)



(b)



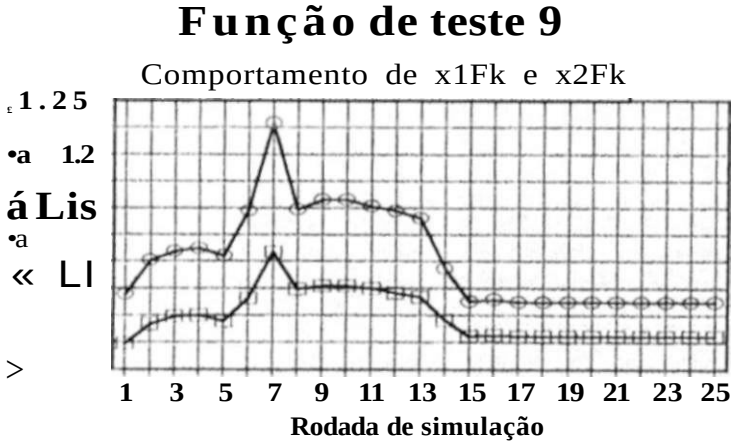
(c)

Figura 8 5.1 - Rapidez com que o algoritmo mnverge para a região ótima. (a) função 3 (b) função 5 (c) função 7.

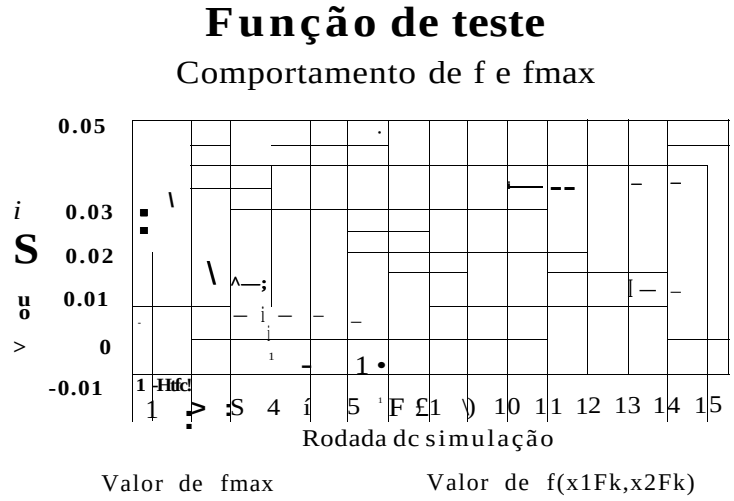
Observamos que estes resultados são obtidos apesar das funções apresentarem diversos máximos locais, observe o comportamento de xp nos gráficos acima (e para mais completa ilustração nos anexos no final deste trabalho). Hssa observação é verdadeira também no caso das funções com duas variáveis, por exemplo, a função 9 (Rosenbrock) atinge a região do ótimo na iteração 13 (XIF = 1.075614 e X2F = 1.210256) e a função 12 (Goldslein-Price) atinge a região do ótimo na iteração 8 (XJF = 0.005966 e Xa» = -1.00732) ou com uma precisão menor na iteração 1 (x_{n}.- = 0.120516 e *Xjp* = -0.94824), no exemplo de

função de quatro variáveis (função 14) a região de ótimo é atingida na iteração 10 (x_{1} = 0.554892, x_{2} = -0.36798, x_{3} = 0.425877 e x_{4} = -0.53239).

Uma outra observação importante é que nas funções de uma variável nosso espaço de busca tem o tamanho de 100 unidades - exceto para a função 8 definida no intervalo [0,1] - e o vetor aleatório é composto de 100 elementos e isso representa uma densidade de 1 ponto/unidade de busca, nas funções de duas variáveis, por exemplo, função 9 (Rosenbrock) o espaço de busca é [0 10J x [0 I0J (quadrado de área 100) e o vetor aleatório é composto de 200 elementos, vale dizer, uma densidade de 2 pontos/unidade de busca (supondo que estas são quadrados unitários) e no caso da função de quatro variáveis o espaço de busca é dado por [-10, I0jx[-10, I0]x[-I0, 10]x[-10, 10] (um hipercubo de volume 20x20x20x20 = 160.000) e o vetor aleatório de 200 pontos o que nos dá uma densidade de 1/800 pontos por unidade de busca (supondo hipercubos unitários) o que nos leva a conjecturar quanto a dependência da precisão relativamente a densidade de pontos gerados no espaço de busca e está sugerido como estudo futuro no capitulo 9 (procuramos ilustrar este fato fazendo variar o espaço de busca e mantendo constante o tamanho do vetor aleatório (simulando um aumento da densidade de pontos) na função de Rosenbrock como pode ser verificado na figura 8.5.2 a 8.5.4 [(a) e (b)| abaixo apresentado no anexo '/.).

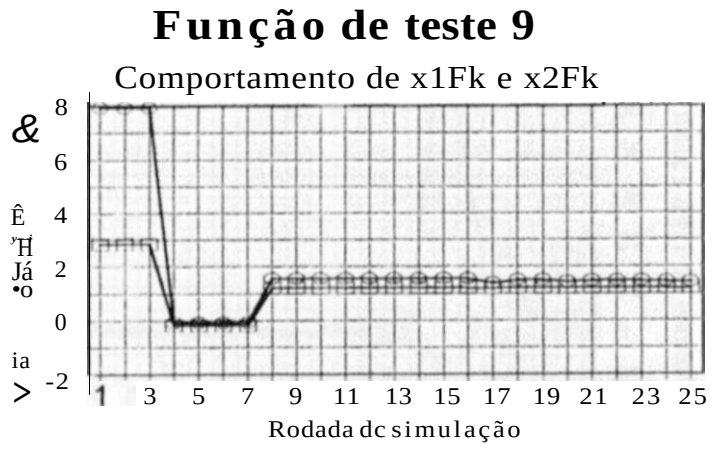


(a)

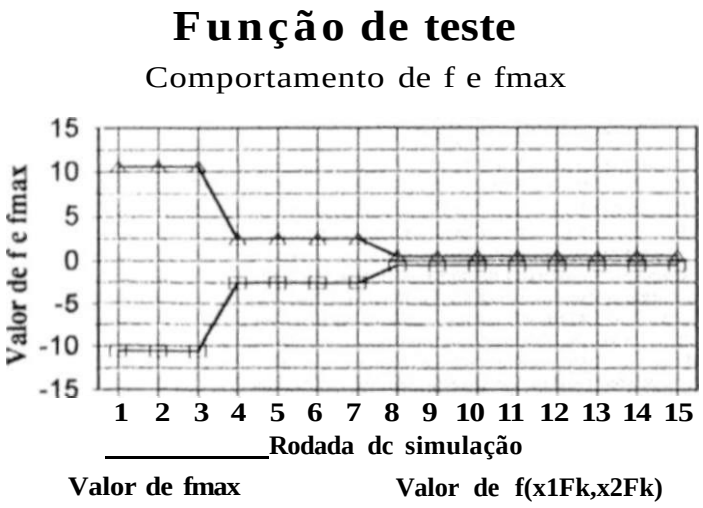


(b)

Figura 8.5.2 - Função dc Rosenbrock com espaço de busca **[0.5 1.5]** x **[0.5 1.5].** (a) comportamento dc x_M. e x_2j. (b) comportamento de $f_{max}(xi^k)$ e ***i\xn:*** , x_2**r).**

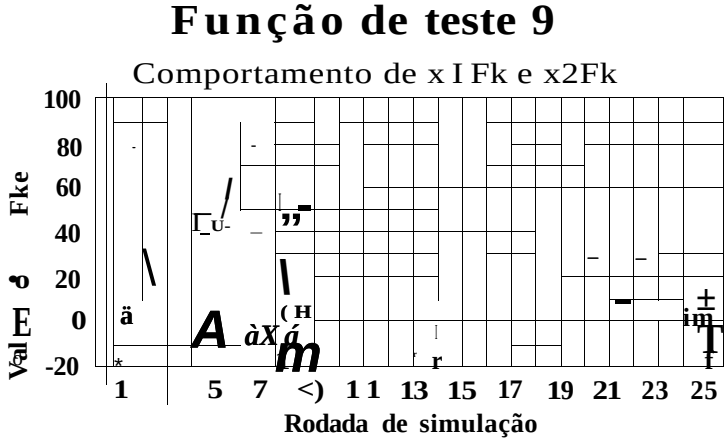


(a)

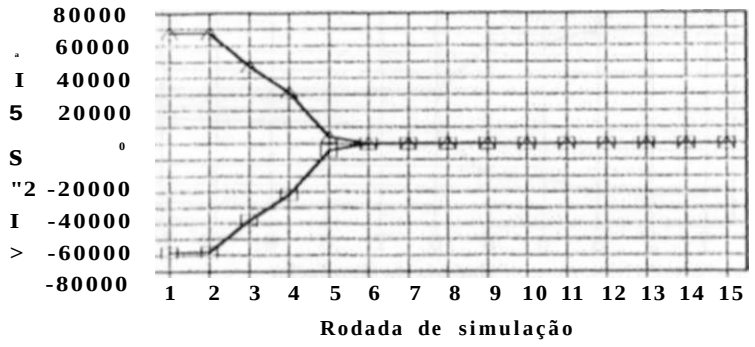


(b)

Figura 8.5.3 - Função de Rosenbrock com espaço de busca [-20 20J x [-20 20]. (a) comportamento de Xip e x_2|. (b) comportamento de f,„„x(xj^k) e *í[x_{iv},* x_2r).



(a)



(b)

figura 8.5.4 - Função de Rosenbrock com espaço de busca |-100 1001 x |-I0() I00J. (a) comportamento de *X\v* e \?v (b) comportamento de f_{max}(xj) e Í(XIF,X2F).

Através dos gráficos de f_{max}v(\j^k) podemos observar também a atuação do curare pois a seqüência { $fn_{\cup}$x(xi^k)} fica presa em determinado valor até que um valor melhor apareça. Isso pode ser observado, por exemplo, na figura 8.5.2 (b) entre <s iterações I e 4, 5 e 12 onde o valor de f_{max}x(xj^k) não muda, isso não significa que o algoritmo esteja estático, os pontos do vetor aleatório estão mudando a cada iteração mas nenhum deles consegui gerar um \ ;ilor para f(x) maior que o f_{m},>(x;^{k}) anterior, quando isso ocorrer a distribuição muda bruscamente seu centro e os pontos do vetor aleatório passam a se deslocar nessa nova direção.

Um outro aspecto que devemos evidenciar é quanto a geração da seqüência {$f_{max}{}^k$} que todos os gráficos das funções de teste mostram ser monotônica não decrescente, aqui apresentamos estes gráficos para as funções de teste 9, 12 e 15, nos gráficos (a), (b) e (c) abaixo (os valores de f_{milx}(xj^k) aparecem negativos porque estamos minimizando estas funções maximizando -fi»).

paralelo.

Esperamos que o computador TALUS (o primeiro randomizador que se tem registro na história da humanidade) seja capaz, de resolver problemas de: integração numérica, equações algébricas, equações diferenciais, otimização (programação linear, não linear, global, dinâmica, inteira, controle ótimo), simulação, teoria da decisão, redes neurais, processamento de sinais, reconhecimento dc padrões, controle de processos, etc.

Como será visto ao longo desse trabalho a abordagem probabilística possibilita uma paralelização natural de uma grande classe de problemas de matemática aplicada. Na verdade, cogiia-se que a vasta maioria dos problemas de matemática aplicada pode ser paralelizada dessa forma.

Por outro lado, os algoritmos probabilísticos permitem a abordagem conseqüente dc problemas de difícil solução prática, como é o caso, nesse trabalho, dos problemas de otimização global, bem como dc problemas dc programação não linear em geral. Isso mostra a importância do tratamento probabilístico de problemas de otimização, mesmo usando-sc as atuais arquiteturas.

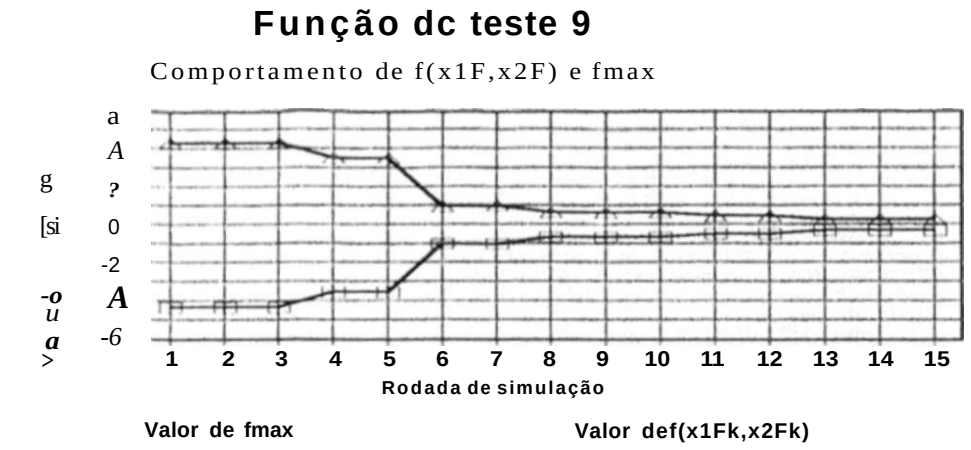
Km pesquisa operacional a resolução científica de problemas envolve:

i) A construção dc modelos matemáticos (descrição do problema), econômicos, estatísticos, financeiros, etc.

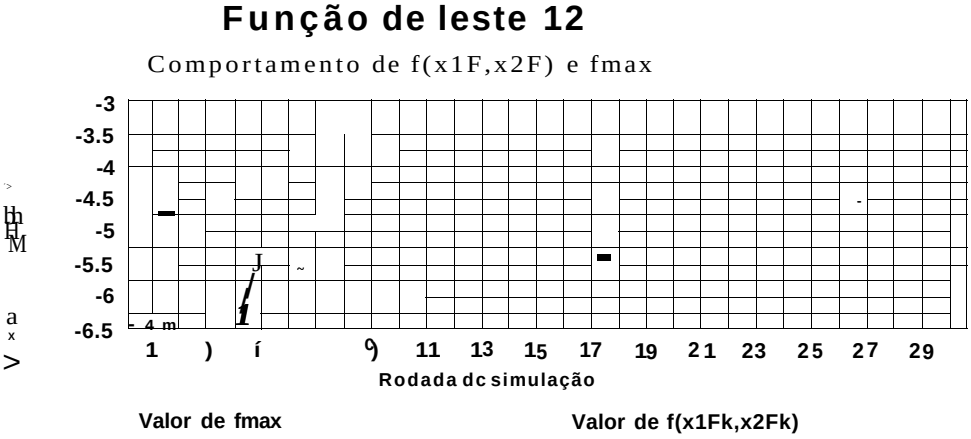
ii) Análise das relações que determinam as conseqüências futuras prováveis dc ações alternativas

O presente trabalho tem como objetivo principal fornecer aos pesquisadores cm geral, c, em particular, os da área de pesquisa operacional, um algoritmo eficiente, rápido, dc fácil implementação computacional c, principalmente, permitir sua utilização em máquinas que utilizem o paralelismo levando-o, assim, a convergir muito rapidamente à solução (x", **fíV))** no problema de otimização global.

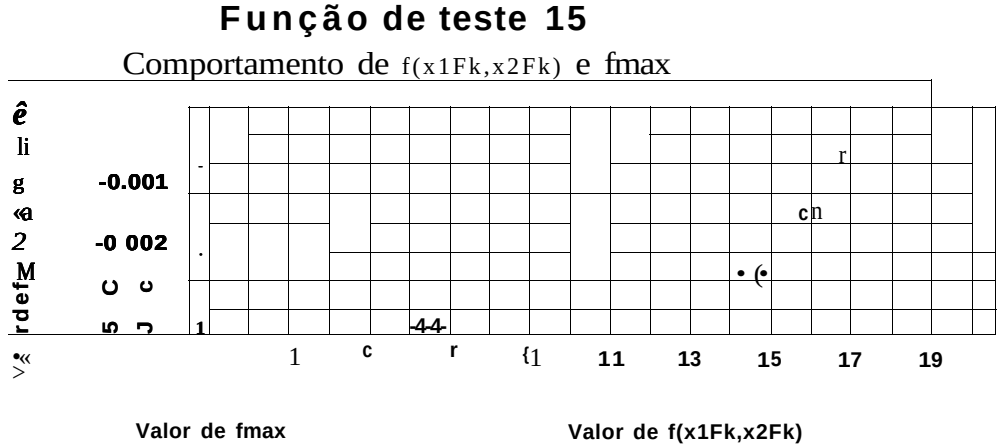
As técnicas utilizadas cm solução dc problemas dc programação não linear são, normalmente, assentadas em diversas suposições acerca do comportamento das funções do problema, tais como dimensão finita do espaço, convexidade, compacticidade do conjunto de soluções viáveis, etc. A



(a)



(b)

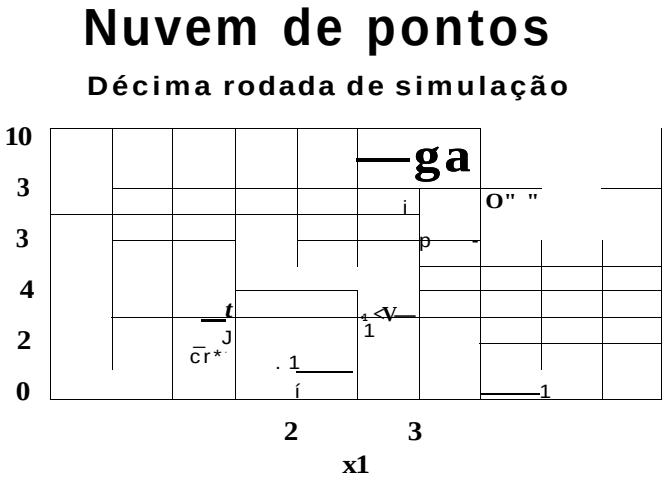


(C)

Figura 8.5.5 - Comportamento de $\{f_{max}(x_j^k)\}$ (a) Função 9 (b) função 12 (c) função 15.

Um outro aspecto interessante dessa discussão ao comportamento da nuvem de pontos, em relação a isso apresentamos abaixo os gráficos das funções de teste 9, 12 e 15 relativos a primeira, décima c vigésima iodada de simulação onde verificamos que o algoritmo realmente conduz os pontos da nuvem para posições tais que a média converge a cada iteração para o ponto de ótimo.

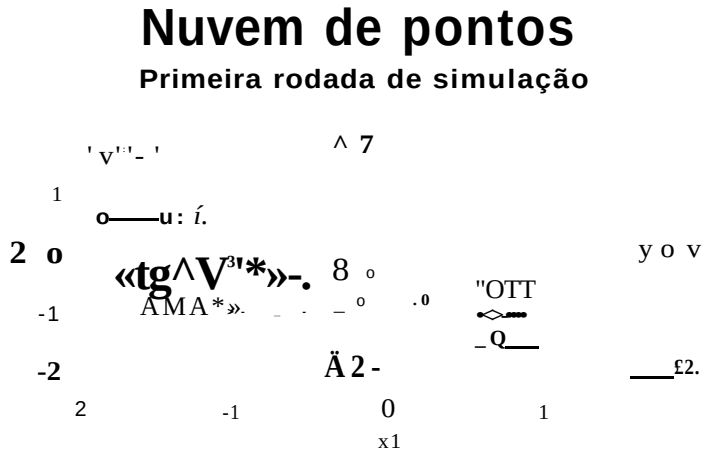
(a)



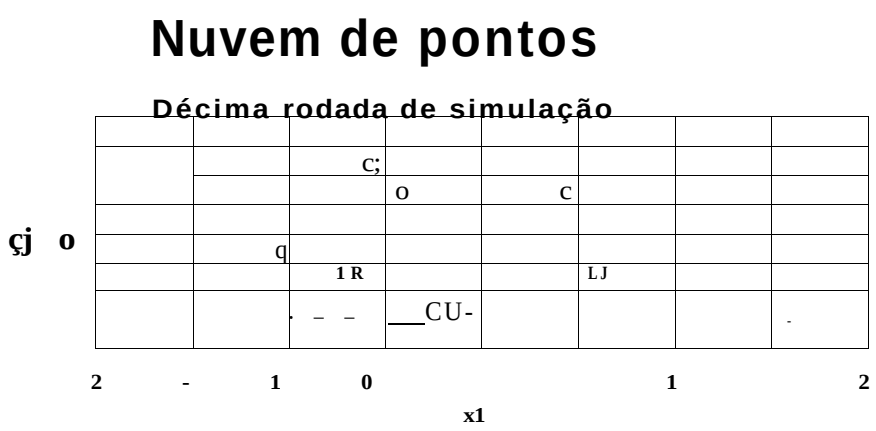
(b)

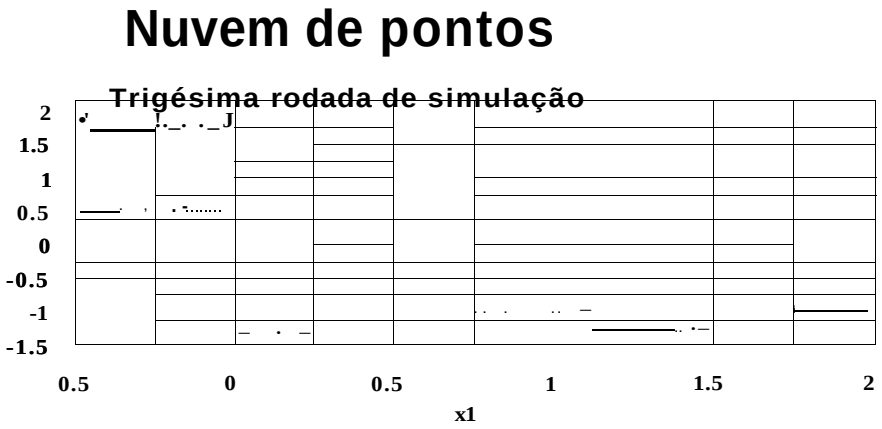
(C)

Figura 8.5.6 - Gráficos das nuvens dc pontos das primeira, décima e vigésima iteração da função de Rosenbrock



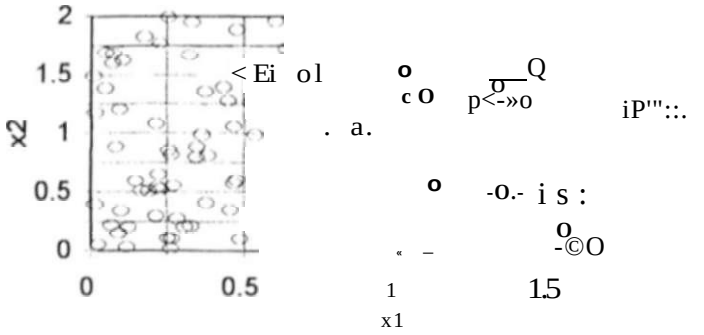
(a)



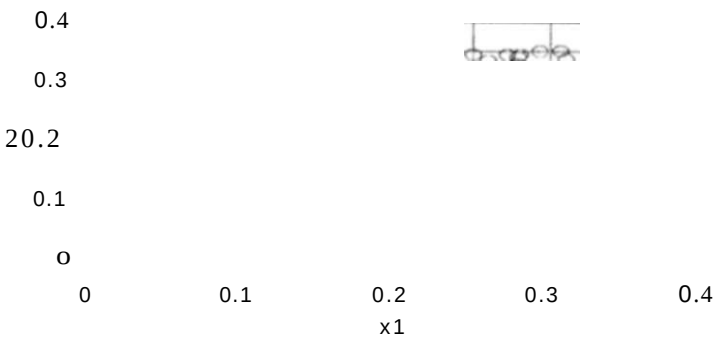


(c)

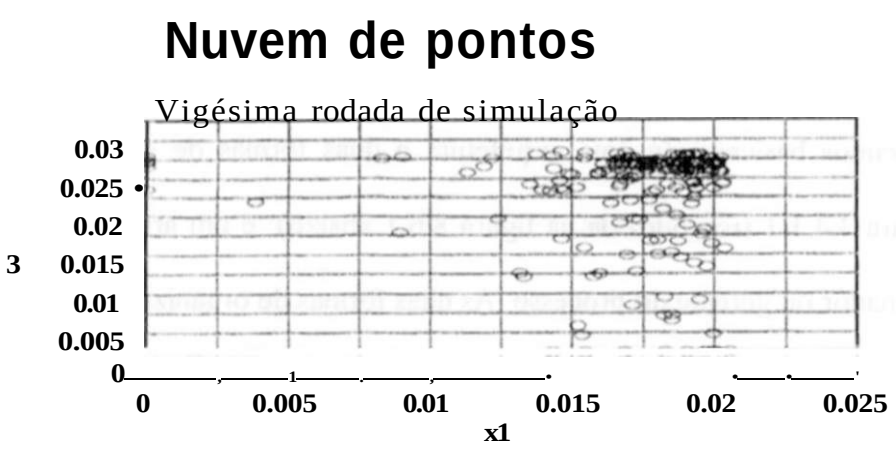
Figura 8.5.7 - Gráficos das nuvens de pontos das primeira, décima e vigésima da função de Goldstein-Price.


Nuvem de pontos
Primeira rodada de simulação
x2
x1
2
1.5
1
0.5
0
0
0.5
1
1.5

(a)


Nuvem de pontos
Décima rodada de simulação
x1
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4

(b)



(c)

Figura 8.5.8 - **Gráficos** das nuvens de pontos das primeira, décima e vigésima iteração da função de teste 15.

Observamos também que a grande maioria dos pontos do vetor aleatório anda na direção do valor verdadeiro x* como mostrado nos anexos de B a Z e que existe uma forte diminuição da variância amostrai, isto é, $a^k > 0$ quando k > oo (onde a^k é a variância do vetor aleatório na iteração k).

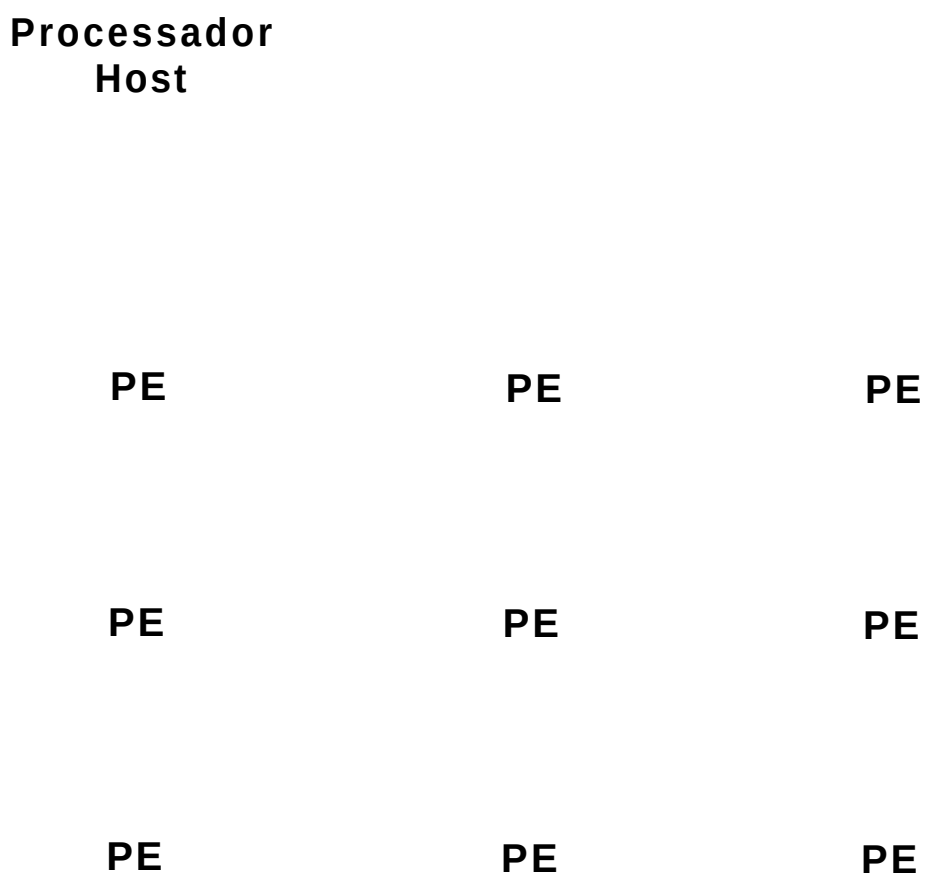
8.6 Proposta de uma arquitetura paralela para implementação cm hardware

Uma das aplicações possíveis para esse algoritmo seria, por exemplo, em controle de processos, feríamos uma função objetivo dinâmica, isto é, com seus valores de parâmetros sofrendo alterações ao longo do tempo e precisando ser otimizada on-line. Com esse propósito poderíamos ter esse algoritmo implementado em forma de circuito integrado VLSI.

Duas formas de implementação do algoritmo são: nível de instrução e nível de aplicação. No primeiro as instruções são realizadas em paralelo, por exemplo, C = (ci , **C2)** = (ai + bi , *a.2* **+ D2)** poderia ser implementado para realização simultânea com dois processadores um executando ai *¥ b*\ e o outro *&2* **+ D2.** No segundo caso poderíamos ter duas ou mais aplicações rodando concorrentemente executando passos

do algoritmo que não dependem dos outros e uma aplicação gerente controlando o processo como um todo e executando as tarefas totalizadas, por exemplo, eleição de fm»ₓ(xiᵏ), cálculo de aᵏ, etc.

Visualizamos basicamente uma arquitetura e duas formas de organização A arquitetura está mostrada na figura 3.4.1.1 (reproduzida na figura 8.6.1 abaixo); é um arranjo de processadores com um elemento coordenador ou gerente do processo. As duas formas de organização são:



1) Cada processador seria responsável pelo processo de realização das iterações naquele ponto por ele gerado e o processador *host* teria a função de gerente do processo (escolha de G™, cálculo da

assimetria em cada iteração, etc.) essa forma exige um processador para cada ponto a ser gerado e iterado c isso prejudica bastante a flexibilidade do algoritmo quanto ao tamanho da nuvem de pontos;

2) Uma outra forma de organizar o arranjo, essa mais flexível do ponto de vista da quantidade de processadores, é subdividir o intervalo de busca e cada elemento processador ficar responsável pela busca cm um subintervalo e o papel do gerente continuaria a ser de coordenação das iterações, a diferença c que nesse caso os pontos candidatos estariam vindo de regiões diferentes do espaço de busca, essa informação numa segunda etapa poderia ser usada, por exemplo, para usar essa nova região como a nova região de busca do algoritmo.

8.7 Conclusões

A estrutura do algoritmo aqui apresentada é paralelo por construção, significa dizer que não é uma adaptação de uma estrutura seqüencial para funcionar em paralelo como muitos pesquisadores têm proposto Um *hardware* para esse algoritmo pode ser construído a partir de microprocessadores baratos e dedicados formando um arranjo para o qual já existem alguns desenvolvimentos teóricos e práticos no sentido de desenvolvimento de sistemas operacionais, compiladores e *software* básicos em geral. O algoritmo é inerentemente tolerante a falhas por causa da facilidade com que os pontos fora da curva podem ser detectados estatisticamente e eliminados do processo, sendo os erros menores absorvidos pela nuvem de pontos pois trabalhamos sempre com valores médios.

Capítulo 9

Conclusões e sugestões

"Depois de muito refletir, calcular e suar, conclui paradoxalmente que o mundo é composto de uma desordem organizada."

ljyrenz

Capítulo 9

Conclusões e sugestões para pesquisas futuras

9.1 Introdução

Procuramos ao longo desse trabalho evidenciar a necessidade de um algoritmo para otimização global, mais que isso, procuramos ressaltar a demanda computacional desse *t'\po* de algoritmo e por consequência propusemos uma concepção paralela de forma a compensar o grande volume de cálculos envolvidos com um incremento na velocidade com que esses cálculos são realizados; para que o algoritmo pudesse ser inerentemente paralelo cm sua construção a abordagem probabilística foi essencial.

Buscamos apresentar muitas das dificuldades existentes para implementações paralelas bem como as deficiências dos algoritmos determinísticos em algumas de suas variantes e destacamos algumas técnicas utilizadas cm otimização global probabilística.

9.2 Resultados alcançados

Observamos o seu desempenho em duas *dezenas* de funções de teste (algumas de alto grau de dificuldade e outras mais simples) e os resultados obtidos se ainda não são os ideais são, no mínimo, promissores.

Verificamos que a cada iteração os pontos da nuvem *\nic\a*, gerada aleatoriamente, são incrementados/decrementados de forma independente um do *ouiro e* que o seu valor médio converge para o valor ótimo (no mínimo para a região onde o ótimo global se encontra dependendo da *precisão* exigida pela aplicação e do resultado alcançado) como fica evidenciado pelos gráficos das funções de teste mostrados nos anexos B a X.

convergência dos métodos c algoritmos são demonstradas para uma determinada classe dc funções. Nesse sentido têm importância fundamental a velocidade dc convergência, os critérios dc parada c a forma como os erros sc propagam ao longo das iterações, já que na grande maioria das vezes são métodos numéricos que levarão a uma solução em uma certa quantidade de iterações. Por exemplo, algoritmos baseados em derivadas, cm formas matriciais, etc. Além, obviamente, do número dc iterações necessárias para sc chegar a uma solução aceitável.

Procuraremos não nos deter aqui nas técnicas envolvidas no levantamento c construção dessas funções (questão essa que envolve muita arte do projetista e que é bastante explorada ^or-(Wagner]), pois nosso pressuposto básico é que estas são dadas. Nosso objetivo fundamental é: dada a expressão analítica da função objetivo, encontrar o vetor (x*,f(x*)) que soluciona o problema.

Como já mencionado anteriormente, usaremos, exaustivamente, técnicas probabilísticas, cm particular os três primeiros momentos dc uma variável aleatória para sua construção.

Iremos também mostrar como esse algoritmo pode ser utilizado para estimar integrais **c** encontrar raízes dc uma função, l-speramos deste modo estar contribuindo com uma ferramenta realmente útil para os profissionais dc todas as áreas que sc deparem com problemas dc otimização global.

Alguns resultados eram impressindíveis que fossem obtidos de forma a ressaltar que as sequências geradas por esse procedimento numérico tinham comportamentos adequados aos objetivos do algoritmo, são eles:

1) $\|x_{k+1} - x_k\| > 0$ **quando k —>** *co*

2) $a_k > 0$ **quando k ~>** *ao*

3) $f(x_k) \to f(x^*)$ **quando k ->** *ao*

4) f(xj.ᵏ) -> *t\\)* quando k -> oo, sendo x* o ponto onde *t\x)* tem seu máximo maximorum.

Esses resultados estão apresentados nos gráficos constantes dos anexos B a X. Nossa avaliação é que esse objetivo foi alcançado, é preciso lembrar que determinados resultados não são triviais, por exemplo, [Himmelblau] implementou o método de Powell para resolver a função de Rosenbrock e partindo de (-0.990 0.990)' alcançou o ponto ótimo em 1562 iterações, implementou o método de Nelder-Mead e alcançou o mesmo ponto em 32 iterações. Se utilizarmos os algoritmos determinísticos implementados no *Quatro pro* **e quizermos solucionar a função de Rosenbrock, dependendo do ponto inicial o algoritmo perde completamente o ponto de ótimo e se apega a um mínimo local qualquer.**

Uma característica importante desse algoritmo é que sua complexidade não cresce junto com o aumento do número de variáveis na mesma proporção da maioria dos métodos determinísticos existentes como pode ser visto nas funções de duas e de quatro variáveis que implementamos.

Finalmente deve ser resaltada a facilidade de implementação computacional proporcionada por ele e sua relativa adaptabilidade relativa ao espaço de busca (isso foi simulado aumentando e diminuindo o espaço de busca na função de Rosenbrock) como mostrado no anexo Z.

Um outro resultado importante é a solução de problemas de programação linear poder serem resolvidos através da aplicação do algoritmo aqui apresentado, conforme mostrado no capítulo 8.

9.3 Sugestões para pesquisas futuras

Algumas sugestões de pesquisas futuras relativas a esse algoritmo são:

1) Outras aplicações

   a) programação inteira

   b) raízes de funções

   c) estimação dc integrais

2) Estudar os valores ótimos para os parâmetros do algoritmo **(71,72,** ***Yh*** **6|, 62**,5*, a)

3) Demostração analítica dc convergência

Anexo A

Anexo A

Alguns resultados c definições básicas

Para o entendimento dos aspectos teóricos da programação não linear são necessários alguns conceitos sobre os quais são construídos boa parte dos algoritmos existentes, passaremos então a fornecer tais conceitos e **enunciar** alguns teoremas, os teoremas aqui serão apresentados sem prova, entretanto serão feitas referências a bibliografia aonde a prova pode ser encontrada.

Definição 1: f: R'' > R definida sobre um subconjunto X dc R " é dito convexa se V x l , x2 e X, e V 0 G [0,1]

1(0 x I i (I - 0) x2) < 0 fi> 1) i (I - 0)n>2), e é estritamente convexa se

fíGxl I (I -0)x2)<0fi>1) t-(l -9)11x2).

Definição 2: f: R" > R definida sobre um subconjunto X convexo de R" é dita quase-convexa se V x l , x 2 G X, e V0 c [0,l|

l{0xl 1 (1 - 0)x2) < Max {f(x I),f(x2)}, e é estritamente quase-convexa se
fíOxl * (I -G)x2)<Max (n>l),^x2)}, com fi>l) * fi>2).

Definição 3; Um conjunto de pontos é dito convexo se para todos os pares de pontos **xi** e **x?** no conjunto, o segmento de reta ligando-os está inteiramente contido no conjunto. Isto implica em que, para cada ponto x, a relação abaixo é válida

x = 0x, » (l-0)x₂ 0 G [0,11

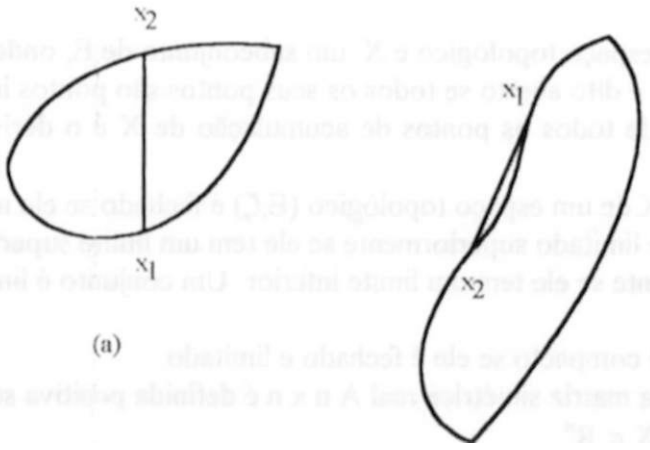


Figura 2.1.1 - Exemplos de conjuntos convexo (a) e não convexo (b).

Definição 4: Seja (E,Q um espaço topológico e X um subconjunto de E, onde E é o suporte do espaço topológico O ponto x G X é um ponto interior de X se existe vizinhança N de x contida em X.

Definição 5: Um espaço vetorial V sobre um corpo K c um conjunto de objetos que podem ser adicionados c multiplicados por elementos de K, de tal forma que a soma de dois elementos de V é

também um elemento de V, o produto de um elemento de V por um elemento de K é um elemento de V e as seguintes propriedades são satisfeitas:

a) Dados u, v e w e V
(uiv)ivv u'(ví\v)

b) Existe um elemento de V denotado por 0 tal que
0t-u = u » 0 - u

c) Dado um elemento u E V, existe um elemento -u E V tal que
u+(-u) - 0

d) Para todos os elementos u e v e V, temos
u+v = v+u

e) Se C é um número, então C(u^ v) = Cu + Cv

0 Se a, b são dois números, então (a t-b)v = av +bv

g) Se a, b são dois números, então (ab)v = a(bv)

h) Para todo elemento u de V, temos 1 .u

Definição 6: Seja V um espaço vetorial, e seja W um subconjunto de V. Definimos W como subespaço se W satisfaz as seguintes condições:

i) Se v, w são elementos de W, sua soma v+w também é elemento de W

ii) Se v e W e C é um número, Cv e W

iii) O elemento 0 de V é também de W

Definição 7: Seja V um espaço vetorial sobre o corpo K, e seja vi,... , v_n elementos de V. Dizemos que vi,...,v_n são linearmente dependentes sobre K se existem elementos ai,...,a_n em K não todos iguais a zero tais que

a,vrK.. 'a^Vn = 0

se tais números não existirem dizemos que Vi,...,v_n são linearmente independentes.

Definição 8: Um ponto x e X é dito ser um ponto de fronteira se qualquer que seja a sua vizinhança existem pontos que pertencem a X e pontos que pertencem a *X.*

Definição 9: Seja (E,Q um espaço topológico e X um subconjunto de E, onde E é o suporte do espaço topológico. O ponto x e X é um ponto de acumulação de X se toda vizinhança contém pontos de X diferentes de x.

Definição 10 Seja (E,Ç) um espaço topológico e X um subconjunto de E, onde E é o suporte do espaço topológico. O conjunto X é dito aberto se todos os seus pontos são pontos interiores.

Definição 11.0 conjunto de todos os pontos de acumulação de X é o derivado de X e é notado como X*.

Definição 12: Um conjunto X de um espaço topológico (E,Q é fechado se ele não é aberto.

Definição 13: Um conjunto é limitado superiormente se ele tem um limite superior e analogamente um conjunto é limitado inferiormente se ele tem um limite inferior. Um conjunto é limitado se ele é limitado superior e inferiormente.

Definição 14: Um conjunto é compacto se ele é fechado e limitado.

Definição 15: Diz-se que uma matriz simétrica real A n x n é definida positiva se X' AX > 0 para qualquer vetor (coluna) não nulo X F R^n.

Teorema I: Seja A uma matriz simétrica real. Então existe uma matriz unitária real U tal que
U*AU = U"'AU é uma matriz diagonal.

Definição 16: seja X um espaço métrico. Todos os pontos e conjuntos mencionados abaixo são elementos e subconjuntos de X.

a) Uma vizinhança de um ponto p é um conjunto N_r(p) consistindo de todos os pontos q tais que d(p,q) < r. O número r é chamado o raio dc N_r(p).

b) Um ponto p é um ponto limite do conjunto E se toda vizinhança de p contém um ponto q * p tal que q e E.

c) Sc p **G** E e p não c um ponto limite de E, então p é chamado um ponto isolado de E.

d) E c fechado sc todo ponto limite de E c um ponto de E.

c) Um ponto p é um ponto interior de E se existe uma vizinhança N de p tal que NczE.

0 E é aberto sc todo ponto de E é um ponto interior de E.

g) O complemento de E (E^c) é o conjunto de todos os pontos p **G** X tais que p í E.

h) E é compacto se todo subconjunto infinito de E tem um ponto limite em E.

i) Bé pci feito se E é fechado e sc todo ponto de E é um ponto limite de E.

j) E é limitado sc existir um número real M e um ponto q **G** X tal que d(p,q) < M para todo p **G** E.

k) L& é denso sc todo ponto de X é um ponto limite de E, ou um ponto de E (ou ambos).

Definição 17: A coleção de conjuntos {G„} é dita ser uma cobertura do conjunto K se somente se *K([J(i„* , onde II é o conjunto de índices. Essa cobertura é dita ser aberta se somente se todo G„ é um « d / conjunto aberto.

Definição 18: Um conjunto K c dito ser compacto se somente se toda cobertura aberta de K contém uma subcobertura aberta finita de K. Isso significa precisamente que se {G„} é uma coleção de conjuntos abertos c sc

k c, **\Ja**$_a$

onde 11 é um conjunto de índice, então existe uma subcoleção finita G„i, G<x2....G,* de {G„} tais que

Definição 19: Seja (E,Ç) um espaço topológico e A um subconjunto dc E. Um ponto x **G** E se diz aderente a A se e somente se numa vizinhança qualquer de x existem elementos de A.

Definição 20: O conjunto dc todos os pontos aderentes a A se denomina aderência ou fecho dc A (notado *Ã).*

Método de métrica variável

E uma classe de métodos (também denominados quase-Newton ou gradiente de passo grande (large-step)) que aproxima a matriz llessiana ou sua inversa usando somente a informação de primeira ordem (derivada primeira) Muitos desses métodos empregam direções conjugadas alguns outros não. Métodos de métrica variável calculam um novo vetor x usando a equação abaixo:

x*$^{+\cdot}$ = x* + A ^ * = x* - ^ x*) v / (x*)

onde n(x^k) é chamada dc matriz dirccional c representa uma aproximação de H"'(x), mais informações pode ser encontrada em |l limmelblau) e [Avriel].

Direção conjugada

Duas direções Sj e S; são ditas conjugadas se

SJQSj **- O s c i / j**

S - (/V, £ 0 sc i - j

onde Q = V ^ x^k) é uma matriz quadrada positiva definida.

Definição 21 - A seqüência {a,,} converge para o ponto A se somente se dado qualquer e > 0, existe um inteiro positivo N tal que **\a„ - *A\ { s*** para todo inteiro n > N. Uma seqüência é dita ser convergente se somente se existe um ponto A tal que {a„} converge para A. O ponto A é denominado o limite da seqüência {a,,}, c escrevemos Lim a„ = A.

Lema 1 - Seja {a_n} uma seqüência monotônica não decrescente limitada, e seja A o supremum desta seqüência. Então dado qualquer número positivo **6,** existe um inteiro positivo N tal que **\a„ - *A\ { 8*** para todo inteiro n > N.

Teorema 2 - Toda seqüência limitada {a„} contém uma subseqüência convergente.

Definição 22 - A seqüência {a„} c dita ser uma seqüência de Cauchy se para todo e > 0 existe um inteiro N tal que n > N, m > N implica em

I a_n - aj < **e.**

Definição 23 - Uma seqüência $\{a_n\}$ dc números reais c dita ser

a) monotonicamente crescente sc a„ < a**„ii** (n = 1, 2, 3, . .);

b) monotonicamente decrescente se $a_{11} > a_n$**.i**(n 1,2,3,,..).

Teorema 3 - Suponha que $\{a_n\}$ é monotônica. Fntão $\{a_n\}$ converge se somente se ela é limitada.

Anexo B

Cálculo	9.946444	57.46467	72.81716	61.1341	76.02848	93.5285	63.31607

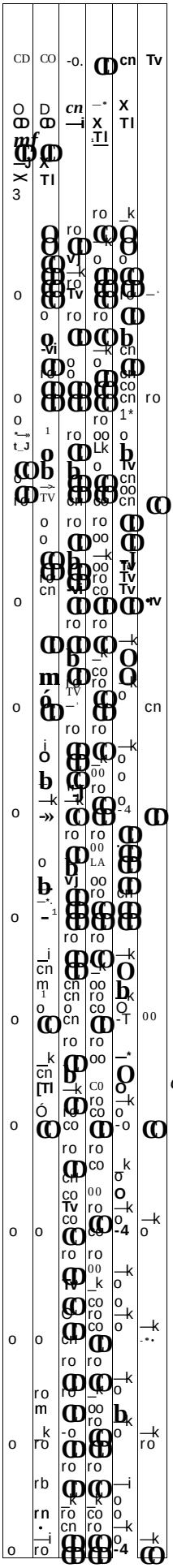
Capítulo 2

gramação não linear

'Pensar que a natureza ê formada por fenômenos previsíveis é, ate certo ponto, sentir o criador. A previsibilidade dos fatos, sua lógica e razão nos conduz facilmente à matemática, enquanto a imprevisibilidade, o caos, nos atola no pântano da escuridão".

Descartes

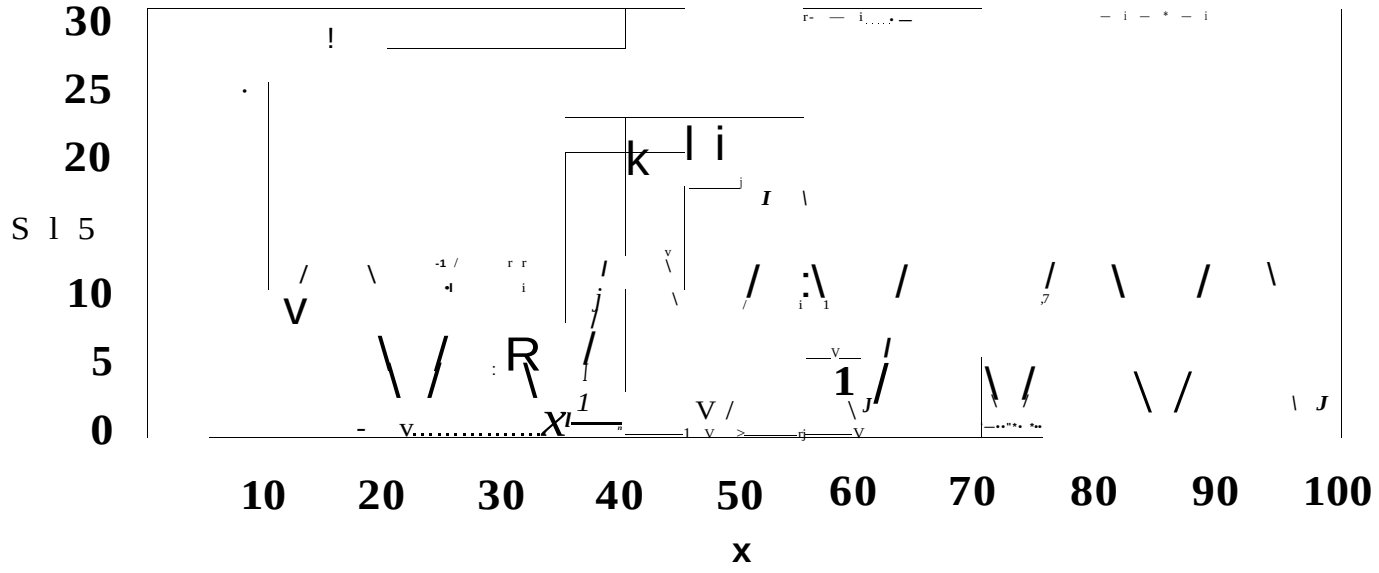
A:A25: **Iteração**
A:B25: 1
A:A26: **x**
A:B26: **+$B$13+($C$13-$B$13)*@RAND**
A:C26: **+$B$13+($C$13-$B$13)*@RAND**
A:A27: **f(x)**
AB27: **+$B$6*@EXP(-(B26-$C$6)^(2*$D$6))+$E$6*@EXP(-(B26-$F$6)^(2*$G$6))+$H$6*B26+$I$6+$B$9*(B26-$C$9)^$D$9*@EXP(-$E$9*(B26-$F$9))+$G$9*@SIN($H$9*B26-l9)**
A:C27: **+$B$6*@EXP(-(C26-$C$6)^(2*$D$6))+$E$6*@EXP(-(C26-$F26-$C$9)^$D$9*@EXP(-$E$9*(C26-$F$9))+$G$9*@SIN($H$9*C26-$l$9)**
A:A28: **a**
AB28: **(B22-$DA27)^3**
A:C28: **(C22-$DA27)^3**
AA29: **F**
A:B29: **@EXP(-$E$15*@EXP($G$15*$B21)*(B27-$CZ27)^(2*$F$15))**
A:C29: **@EXP(-$E$15*@EXP($G$15*$B21)*(C27-$CZ27)^(2*$F$15))**
A:A30: **F/SumF**
A:B30: **+B29/($D$13+$CX29)**
A:C30: **+C29/($D$13+$CX29)**
A:A31: **Curare**
A:B31: **($CZ27-B27)^(1/($D$15+($CZ27-B27)))**
A:C31: **($CZ27-C27)^(1/($D$15+($CZ27-C27)))**
A:A32: **xF-a-xi**
A:B32: **+$CZ29-B22**
A:C32: **+$CZ29-C22**
A:A33: **f(xF-a)**
A:B33: **+$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*($CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)**
A:C33: **+$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$B$9*($CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)**
A:A34: **Sinal**
AB34: **(B33-B26)/($D$13+(B33-B26))**
A:C34: **(C33-C26)/($D$13+(C33-C26))**
AA35: **Passo**
A:B35: **+$B$15*@EXP(-($C$15/$B25))**
A:C35: **+$B$15*@EXP(-($C$15/$B25))**
AA36: **Cálculo**
A:B36: **+B26+(B35*B3rB34*B32)**
A:C36: **+C26+(C35*C31*C34*C32)**
A:CX26: **@SUMPRODUCT(B26. CW26.B27. CW27)/$CX27**
A:CZ26: **fmax**
A:DA26: **xF**
A:CX27: **@SUM(B27..CW27)**
A:CZ27: **@MAX(B27..CW27)**
A:DA27: **@SUMPRODUCT(B26..CW26,B30..CW30)**
A:CX28: **(@SUMPRODUCT(B26..CW26^29..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)**
A:CZ28: **xF-a**
A:DA28: **f(xF)**
A:CX29: **@SUM(B29..CW29)**
A:CZ29: **+DA23-CX28**
A:DA29: **+$B$6*@EXP(-(DA27-$C$6)^(2*$D$6))+$E$6*@EXP(-(DA27^F$6)^(2*$G$6))+$H$6*DA27+$I$6+$B$9*(DA27-$C$9)^$D$9*@EXP(-$E$9*(DA27-$F$9))+$G$9*@SIN($H$9*DA27-$I$9)**
A:CZ30: **STD x**
A:DA30: **Média**
A:CZ31: **@STD(B26..CW26)**
A:DA31: **@AVG(B26..CW26)**



>	AL	AM	> z
4V	ro CO	ro CO	co o
—A cn	o .01071	10.0107	o 02237
cn	ro 00 18239	28.18239	ro CO CO ro CO CO
	ro cn 37618	25.0426	ro 4v 69377
00	1 .2E-05	0.011673	. o.01167
—k. «o	o	o	o

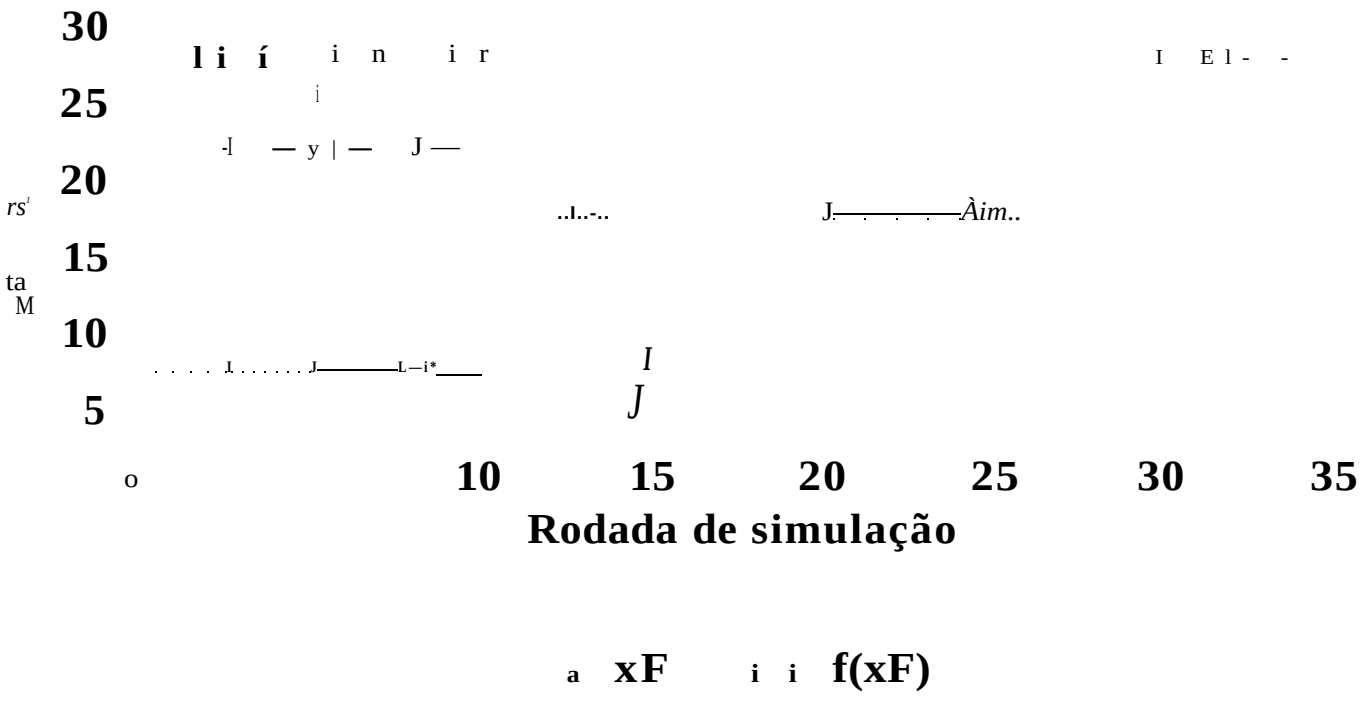
Função de teste unidimensional

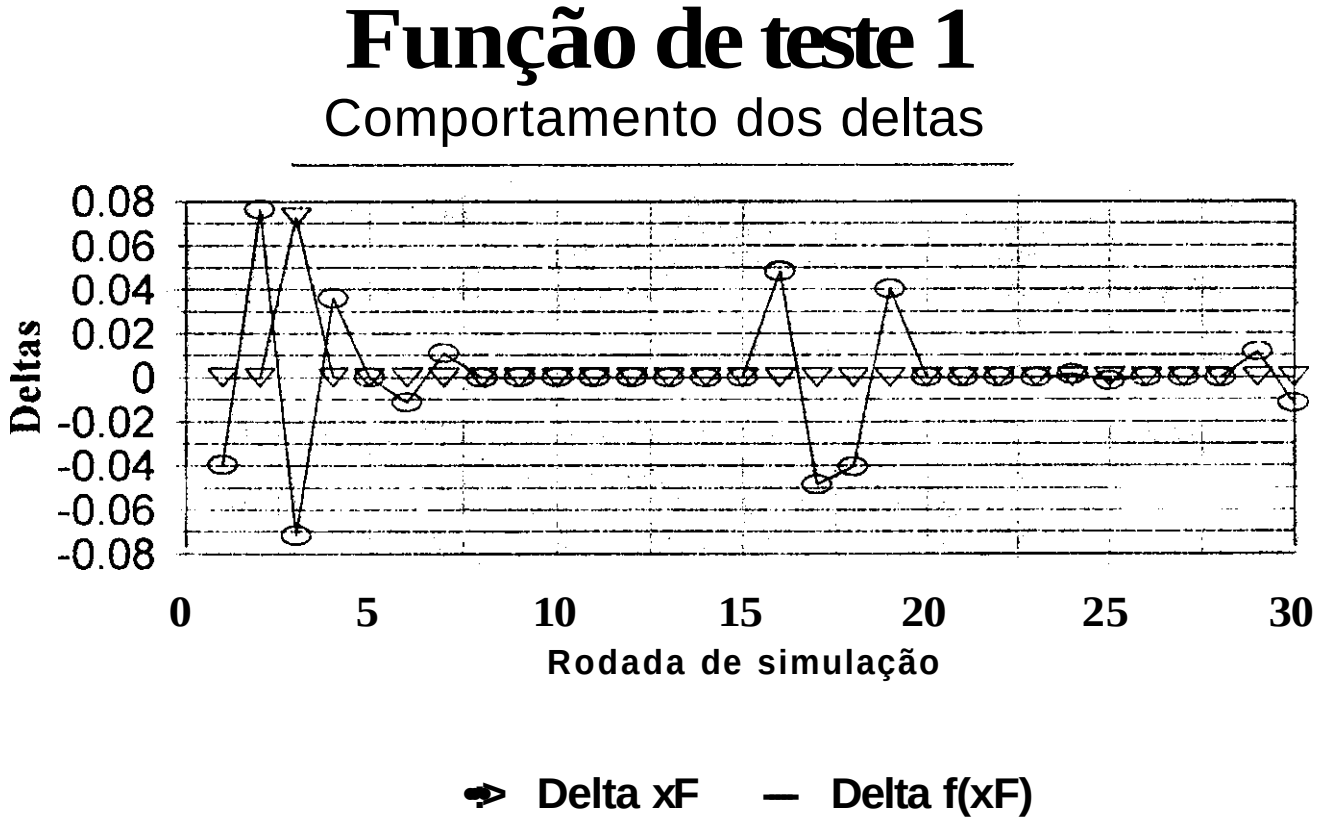
Função 1



Função de teste 1

Comportamento de xF e f(xF)




Função de teste 1
Comportamento dos deltas
Deltas
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
0
5
10
15
20
25
30
Rodada de simulação
Delta xF
Delta f(xF)

Função universal de 1teste para o caso unidimensinal

Função de te:

Parâmetros da função de tes

	Exponenci	Exponenci	Reta
	Polinómio	Exponenci	Seno

Parâmetros do algoril:mo

	xmin	xmax	Erro

	Gama 1	Gama 2	Gama 3	Delta 1	Delta 2	Delta 3

	f(xFk)	xmax	f(xmax)	Delta x	Delta f
	4.977545	23.86845	23.89418	0.022455	0.025734

Cálculo 12.48584 23.0079 11.91254 97.89741 87.27601 48.62448 57.63436

CAPÍTULO 2

Programação não linear

2.1 Introdução

De acordo com [Avricl] resolver um problema de programação não linear, se nada for dito em contrário, consiste em encontrar um vetor x* que otimize uma solução cm particular e não todas as soluções que possam existir. Reconhecer um vetor x* ótimo e estudar suas propriedades ó o tema central do estudo da teoria da otimização.

A justificativa para o uso de qualquer método de solução dos problemas **1**.2.**1.1** e **1**.2.**1**.2 apoia-se na eficácia de resolver tais problemas em termos de implementação computacional.

Para avaliar o desempenho de um ou vários algoritmos são desenvolvidos estudos comparativos (fora do escopo desse trabalho). A questão a ser respondida é: qual o melhor método a ser utilizado? [Himmelblau] afirma que essa só pode ser respondida de forma muito complexa, pois a resposta depende do tipo de problema a ser resolvido, o grau de preparação a ser realizado pelo usuário e a disponibilidade de informação sobre a região de vetores x factíveis.

[Himmelblau] aponta alguns critérios a serem considerados:

1) Tempo requerido em uma série de testes (tempo de execução ou número de iterações);

2) Tamanho (dimensão, número de restrições de desigualdade, número de restrições de igualdade) do problema;

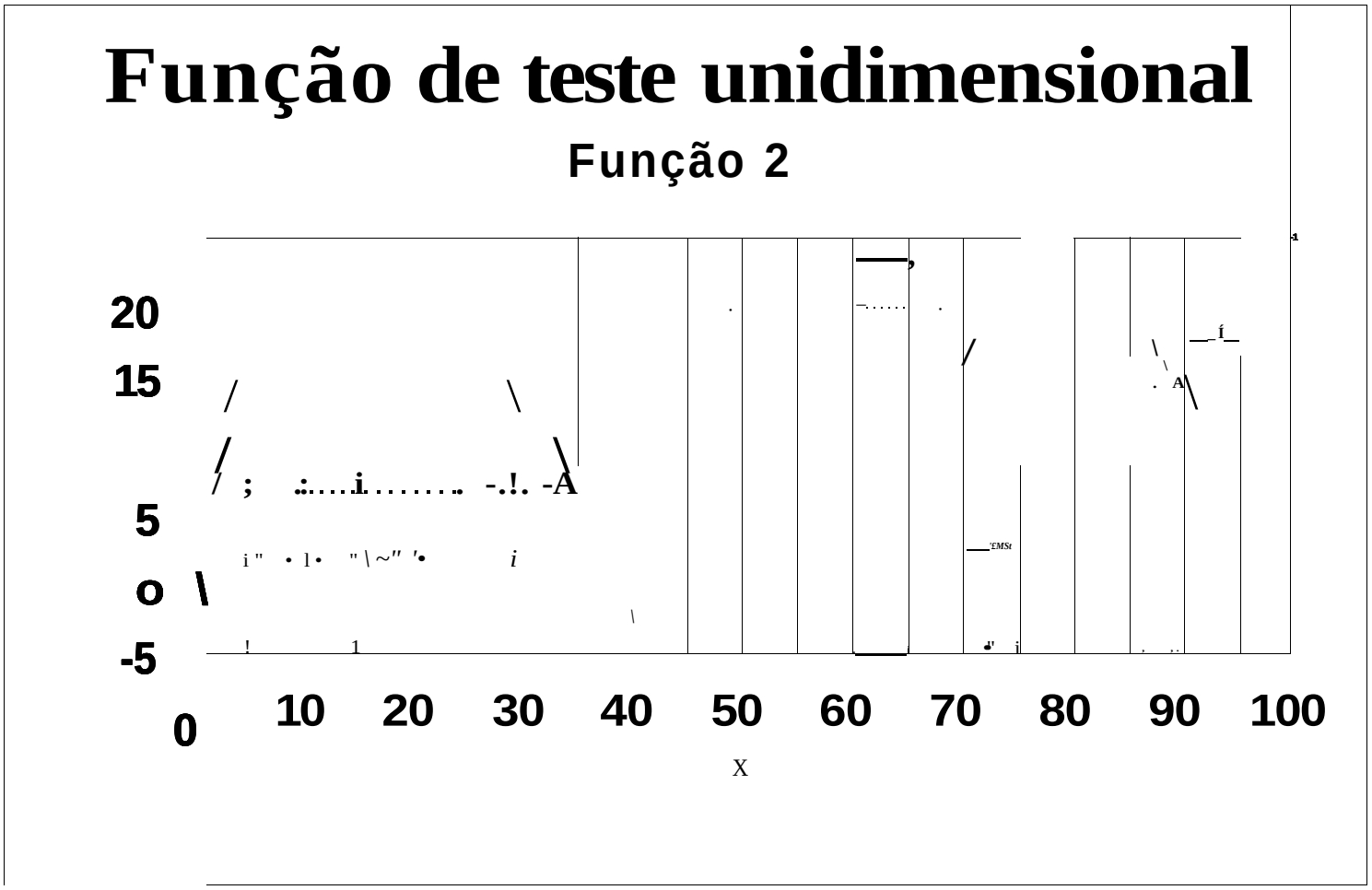
3) Precisão da solução relativa ao vetor ótimo x* e/ou relativa a fíx*), h(x*), g(x*) e Vf(x*);

4) Simplicidade de uso (tempo requerido para introduzir os dados c funções no computador);

5) Simplicidade do programa de computador para executar o algoritmo;

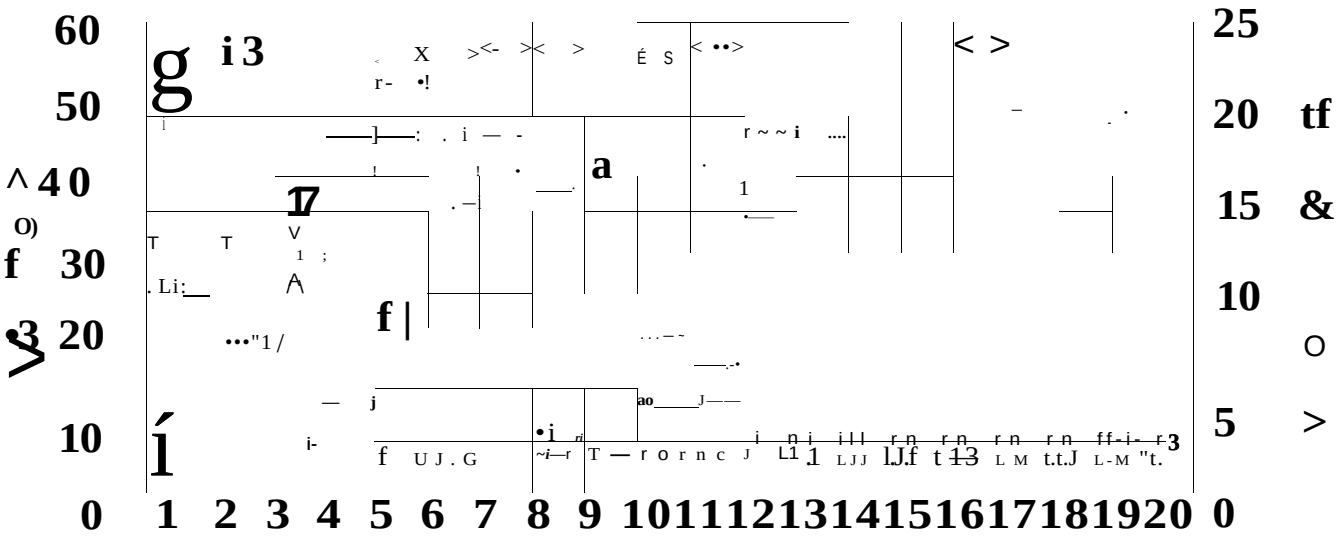
6) Resolva problemas do mundo real.

A:A25: Iteração
A:B25: 1
A:A26: x
A:B26: +$B$13+($C$13-$B$13)*@RAND
A:C26: +$B$13+($C$13-$B$13)*@RAND
A:A27: f(x)
A:B27: +$B$6*@EXP(-(B26-$C$6)^(2*$D$6))+$E$6*@EXP(-(B26-$F$6)^(2*$G$6))+$H$6*B26+$I$6+$B$9*(B26-$C$9)^$D$9*@EXP(-$E$9*(B26-$F$9))+$G$9*@SIN($H$9*B26-$I$9)
A.C27: +$B$6*@EXP(-(C26-$C$6)^(2*$D$6))+$E$6*@EXP(-(C26-$F$6)^(2*$G$6))+$H$6*C26+$I$6+$B$9*(C26-$C$9)^$D$9*@EXP(-$E$9*(C26-$F$9))+$G$9*@SIN($H$9*C26-$I$9)
A:A28: a
A:B28: (B22-$DA27)^3
A:C28: (C22-$DA27)^3
A:A29: F
A:B29: @EXP(-$E$15*@EXP($G$15*$B21)*(B27-$CZ27)^(2*$F$15))
A:C29: @EXP(-$E$15*@EXP($G$15*$B21)*(C27-$CZ27)^(2*$F$15))
A:A30: F/SumF
A:B30: +B29/($D$13+$CX29)
A:C30: +C29/($D$13+$CX29)
A.A31: Curare
A:B31 : ($CZ27-B27)^(1/($D$15+($CZ27-B27)))
A:C31 : ($CZ27-C27)^(1/($D$15+($CZ27-C27)))
A:A32: xF-a-xi
AB32: +$CZ29-B22
A:C32: +$CZ29-C22
A:A33: f(xF-a)
A:B33: +$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*|$CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)
A:C33: +$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6'@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*<$CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)
A:A34: Sinal
A:B34: (B33-B26)/($D$13+(B33-B26))
A:C34: (C33-C26)/($D$13+(C33-C26))
A:A35: Passo
A:B35: +$B$15*@EXP(-($C$15/$B25))
A:C35: +$B$15*@EXP(-($C$15/$B25))
A:A36: Cálculo
A:B36: +B26+(B35*B31*B34*B32)
A:C36: +C26+(C35*C3rC34*C32)
A:CX26: @SUMPRODUCT(B26..CW26,B27..CW27)/$CX27
A:CZ26: fmax
A:DA26: xF
A:CX27: @SUM(B27..CW27)
A:CZ27: @MAX(B27 .CW27)
A:DA27: @SUMPRODUCT(B26..CW26,B30..CW30)
A.CX28: (@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)
A:CZ28: xF-a
A:DA28: f(xF)
A:CX29: @SUM(B29..CW29)
A:CZ29: +DA23-CX28
A:DA29: +$B$6*@EXP(-(DA27-$C$6)^(2*$D$6))+$E$6*@EXP(-(DA27-$F$6)^(2*$G$6))+$H$6*DA27+$I$6+$B$9*(D^27-$C$9)^$D$9*@EXP(-$E$9*(DA27-$F$9))+$G$9*@SIN($H$9*DA27-l9)
A:CZ30: STD x
A:DA30: Média
A:CZ31 : @STD(B26 .CW26)
A:DA31: @AVG(B26..CW26)


Função de teste unidimensional
Função 2
20
15
5
0
-5
0
10
20
30
40
50
60
70
80
90
100
X

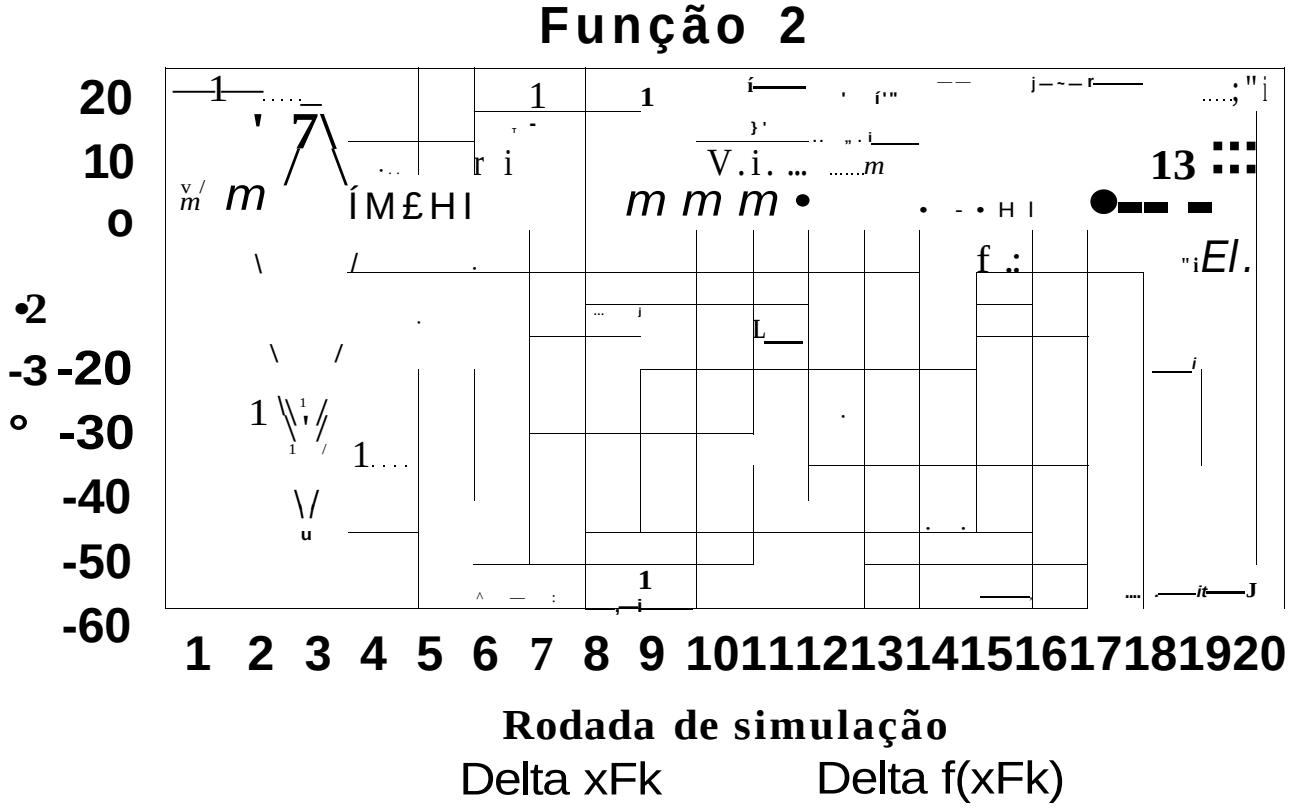
Função de teste unidimensional

Função 2



ID xFk x f(xFk)

Função de teste unidimensional



Anexo D

Função universal de teste para o caso unidimensional

Função de teste

Parâmetros da função de teste

Exponencial 1 — Exponencial 2 — Polinómio — Seno

Parâmetros do algoritmo

Xmax — Erro

Gama 1 — Gama 2 — Gama 3 — Delta 1 — Delta 2 — Delta 3

Resultados

1		xmax		Delta X	Delta f
	5.015121	45.82264	45.83665	0.014006	

Cálculo	61.21479	29.862	50.87057	31.0401	11.83793	76.45873

São referencias importantes aqui [Himmclblau], [Avrielj, | Ribeiro| e [Mahey],

2.2 Condições de olimalidade

Muitas das pesquisas tem sido desenvolvidas na área de otimização não linear com o intuito específico de encontrar resultados que levem as condições de necessidade e de suficiência para a solução desse problema. Apresentaremos as condições de otimalidade (necessidade e suficiência) cm forma de teoremas matemáticos para os quais não apresentaremos prova, sendo, no entanto, informado ao leitor aonde consegui-las.

Seja uma função f com domínio D c R " . Dizemos que f tem um mínimo local cm um ponto x* t D, se existir um õ e R " , 5 > 0 tal que

Eq. 2.2**.1.1** f(x)>f(x*)

para todo x e D, satisfazendo | x - x* j < õ.

Evidentemente nem toda função real tem um mínimo.

Seja x G D c R " ser um ponto onde a função real f é diferenciável. Sabemos que se uma função real fé diferenciável em um ponto interior x e D, então sua derivada parcial de primeira ordem existe em x. Se além disso, se as derivadas parciais são contínuas em x, então fé dita ser continuamente diferenciável em x. Similarmente, se f é duas vezes diferenciável e m x e D , então a segunda derivada parcial existe, se elas são contínuas em x, então f é dita ser duas vezes continuamente diferenciável em x.

Definimos o gradiente de f cm x como o vetor V H[x), dado por

$$\nabla f(x) \quad \frac{\partial f(x)}{\partial x_1} \quad \frac{\partial f(x)}{\partial x_2}$$

Eq. 2.2**.1**.2

```
A:A25.    Iteração
A:B25:    1
AA26:     x
A:B26:     +$B$13+($C$13-$B$13)*@RAND
A:C26:     +$B$13+($C$13-$B$13)*@RAND
A:A27:     f(x)
A:B27:       +$B$6*@EXP(-(B26-$C$6)^(2*$D$6))+$E$6*@EXP(-(B26-$F$6)^(2*$G$6))+$H$6*B26+$I$6+$B$9*(B26-$C
           $9)^$D$9*@EXP(-$E$9*(B26-$F$9))+$G$9*@SIN($H$9*B26-$I$9)
AC27:        +$B$6*@EXP(-(C26-$C$6)^(2*$D$6))+$E$6*@EXP(-(C26-$F$6)^(2*$G$6))+$H$6*C26+$I$6+$B$9*(C26-$C
           $9)^$D$9*@EXP(-$E$9*(C26-$F$9))+$G$9*@SIN($H$9*C26-$I$9)
A:A28:     a
A:B28:     (B22-$DA27)^3
A:C28:     (C22-$DA27)^3
A:A29:     F
A:B29:    @EXP(-$E$15*@EXP($G$15*$B21)*(B27-$CZ27)^(2*$F$15))
A:C29:     @EXP(-$E$15*@EXP($G$15*$B21 )*(C27-$CZ27)^(2*$F$15))
A:A30:    F/SumF
A:B30:     +B29/($D$13+$CX29)
A.C30:     +C29/($D$13+$CX29)
A:A31:    Curare
A:B31:     ($CZ27-B27)^(1/($D$15+($CZ27-B27)))
A:C31:    ($CZ27-C27)^(1/($D$15+($CZ27-C27)))
A:A32:    xF-a-xi
AB32:      +$CZ29-B22
A:C32:     +$CZ29-C22
A:A33:    f(xF-a)
A:B33:       +$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*(
           $CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)
A:C33:       +$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*(
           $CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)
A:A34:    Sinal
A:B34:     (B33-B26)/($D$13+(B33-B26))
A:C34:    (C33-C26)/($D$13+(C33-C26))
A:A35:     Passo
A:B35:    +$B$15*@EXP(-($C$15/$B25))
A:C35:     +$B$15*@EXP(-($C$15/$B25))
A:A36:    Cálculo
A:B36:     +B26+(B35*B31*B34*B32)
A:C36:     +C26+(C35*C31*C34*C32)
A:CX26:  @SUMPRODUCT(B26..CW26,B27..CW27)/$CX27
A:CZ26:    fmax
A:DA26:    xF
A:CX27:   @SUM(B27..CW27)
A:CZ27:   @MAX(B27..CW27)
A:DA27:   @SUMPRODUCT(B26..CW26,B30 .CW30)
A:CX28:      (@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,B29..C
           W29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)
A:CZ28:    xF-a
A:DA28:   f(xF)
A:CX29:   @SUM(B29..CW29)
A:CZ29:    +DA23-CX28
A:DA29:      +$B$6*@EXP(-(DA27-$C$6)^(2*$D$6))+$E$6*@EXP(-(DA27-$F$6)^(2*$G$6))+$H$6*DA27+$I$6+$B$9*(DA
           27-$C$9)^$D$9*@EXP(-$E$9*(DA27-$F$9))+$G$9*@SIN($H$9*DA27-$I$9)
ACZ30:    STD x
A:DA30:   Média
A:CZ31:   @STD(B26 .CW26)
A:DA31:   @AVG(B26..CW26)
```

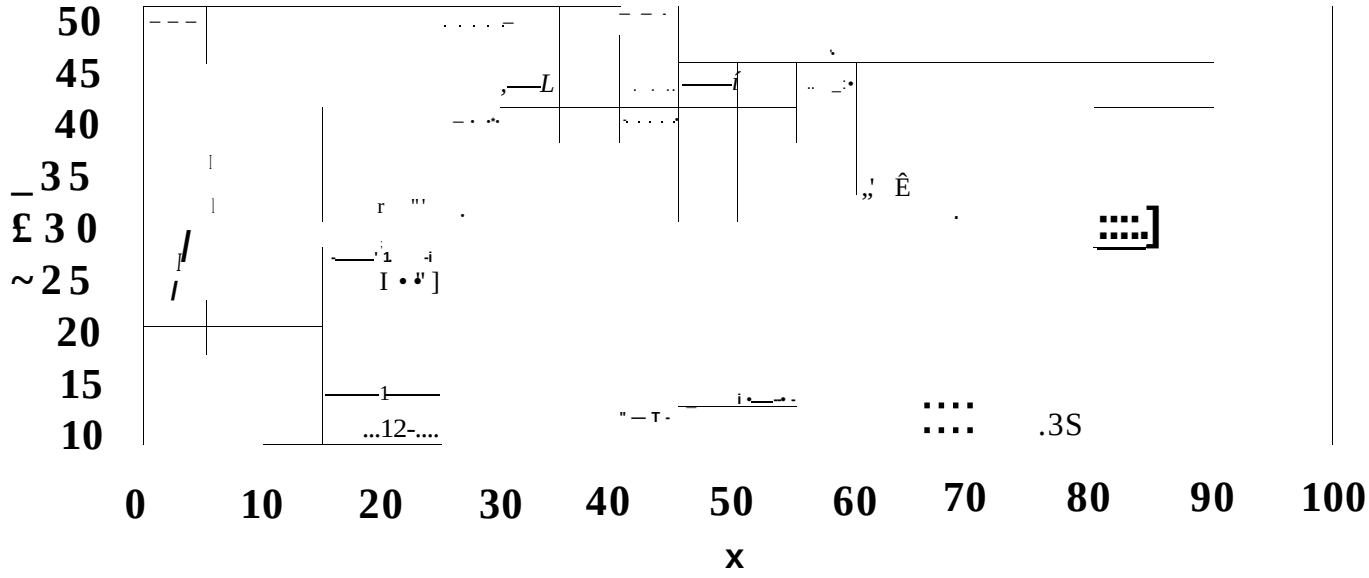
^	c_	r	S	z	O	"0	D	73	Oi	H	/_ v–
14		1	2	3	4	cn	o)	7	8	9	10	¡ ¡ ▪
15	xF	39.72852	6.307853	6.224453	5.230816	5.084858	4.982526	4.967895	5.075724	5.014573	5.075675	Ø 8 K Ø Ø
16	f(xF)	43.45185	43.45185	43.45185	44.66848	45.67479	45.67479	45.67479	45.67479	45.67479	45.67479	cn Ø Ø
17	STD	28.15586	28.1444	28.14373	28.14216	28.1395	28.13326	28.12181	28.10905	28.09518	28.06479	ro 8 ▪ ro ï cn
18	Delta xF	-33.4207	-0.0834	-0.99364	-0.14596	-0.10233	-0.01463	0.107828	-0.06115	0.061103	-0.09098	o o Ø o Ø 8 A
19	Delta f(xF	0	0	1.216632	1.006311	o	o	o	o	o	o	o

^	V		X	V	N	AA	AB	AC	AD	^m	AF	AG
	[illegible]	[illegible]	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	5.075677	5.07555	5.075675	5.035934	5.044935	[illegible]	5.015812	[illegible]	5.015121	5.C3C236	5.C15944	5.015943
[illegible]	45.67479	45.67479	45.67479	[illegible]	45.82162	45.82162	45.82162	45.82162	45.82162	45.82162	45.82162	45.82162
	27.97503	27.9278	27.8795	27.82053	27.74913	27.66642	27.57897	27.4845	27.38011	27.27268	27.14518	27.0071
[illegible]	-2.7E-05	2.6E-05	-0.03974	0.009001	-0.02903	-9.6E-05	0.000132	-0.00082	0.015115	-0.01429	-6.3E-07	0.016928
[illegible]	o	o	o	0.146826	O	O	O	O	O	O	O	O

^	AH	^	AJ	AK	^ T	AM	AN
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	4.970045	5.001596	4.987263	4.987263	[illegible] 99872	4.990949
[illegible]	45.82162	45.82162	45.84132	45.84132	45.84132	[illegible]	45.84132
	[illegible]	26.6908	26.51829	26.33187	[illegible]	[illegible] 91463	25.691
[illegible]	-0.06283	0.031551	-0.01433	o	0.011457	[illegible] 00777	[illegible]
[illegible]	o	0.019698	o	o	o	o	o

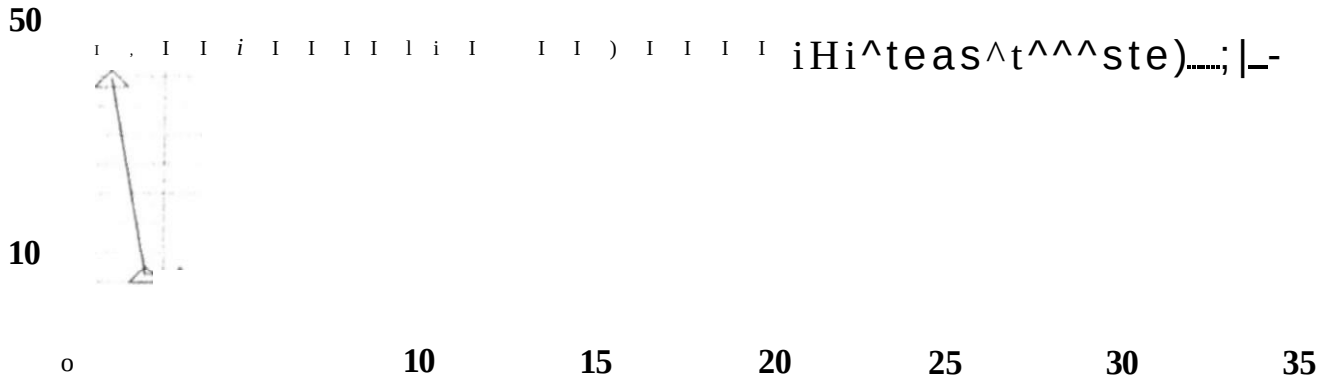
Função de teste unidimensional

Função 3



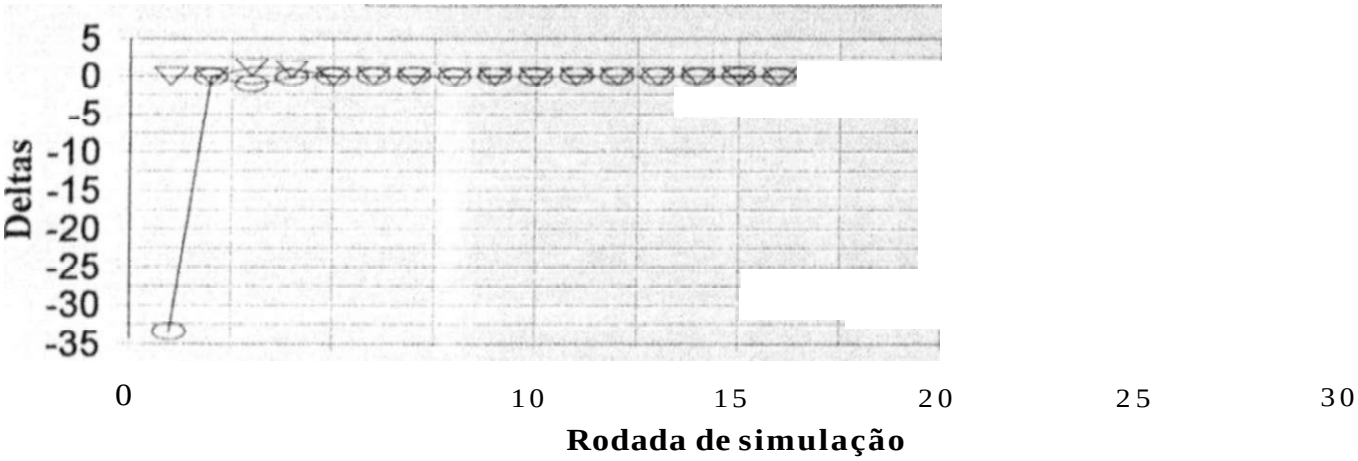
Função de teste 3

Comportamento de **xF** e **f(xF)**

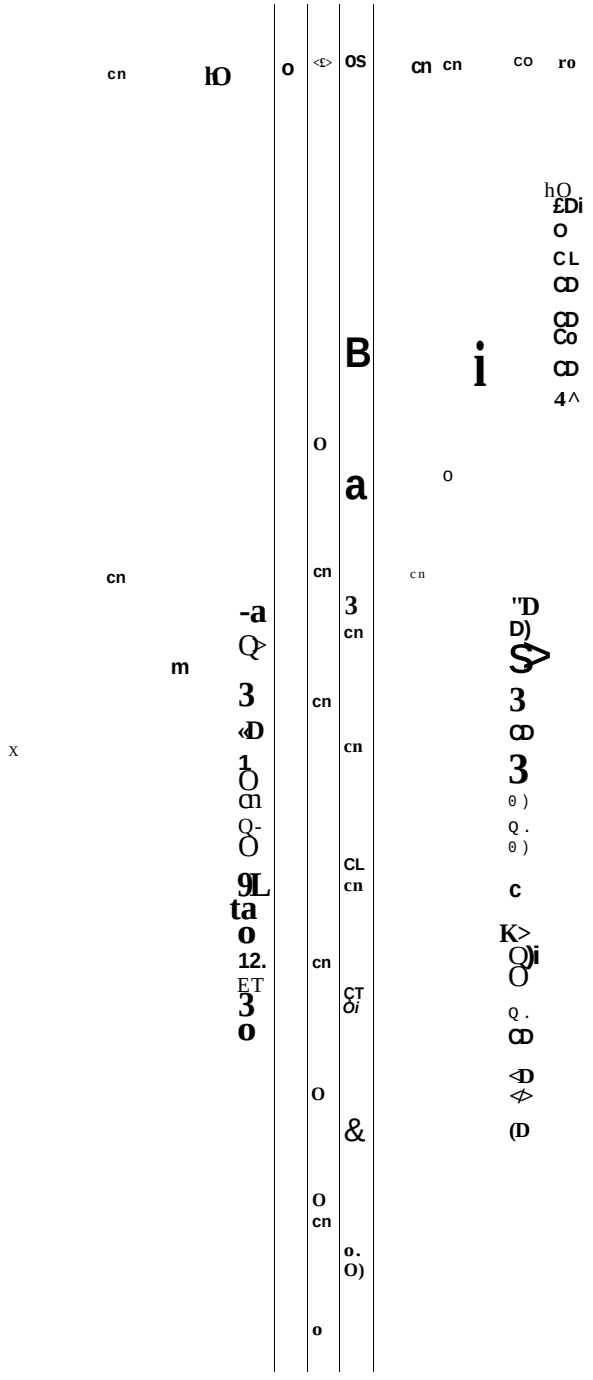


Função de teste 3

Comportamento dos deltas



Anexo E



Similarmente, se f é duas vezes diferenciável em x, definimos a matriz Hcssiana de fem x como a matriz simétrica n x n V²f\x), dada por

$$\mathbf{v^2/(*)} = \frac{c?f(x')}{\hat{a}x\hat{a}xy} \quad \frac{dx.... dx.}{dx} \quad \mathbf{< ? / í > ^{\circ})}$$

ou simplesmente

Eq. 2.2.1.3 $$v7W \quad \hat{a}x_{,} \quad \hat{a}Xj \qquad i,j = l,\ldots,n.$$

2.2.1 (aso desvinculado

Condição necessária

Teorema 2.2.1.1: Seja x* um ponto interior de D no qual ftem um mínimo local. Se fé diferenciável em x* , então

Eq. 2.2.**1**.3 $Vf(x^*) \quad 0$

Condições suficientes

Teorema 2.2.1.2: Seja x* um ponto interior de D no qual fé duas vezes continuamente diferenciável.

Se

Eq. 2.2.1.4 $Vfi>^*) \quad 0$

e

Eq. 2.2.1.5 $Z \; V f (x^*) Z > 0$

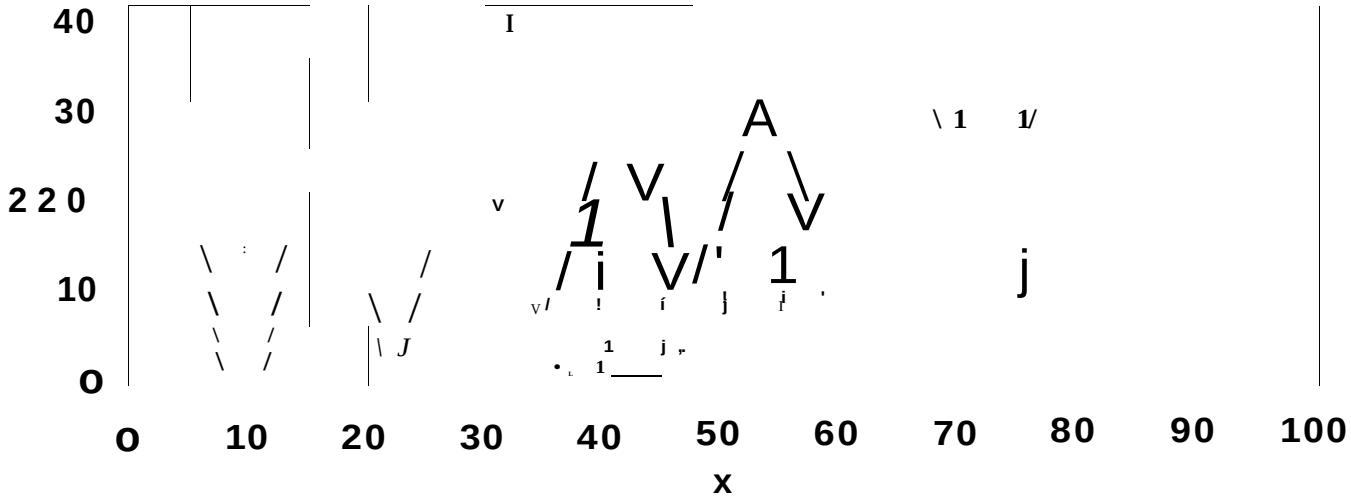
V Z * 0, então f tem mínimo local cm x*. Além disso o mínimo é estrito.

AA25: **Iteração**
A:B25: **1**
A.A26: **x**
AB26: **+$B$13+($C$13-$B$13)*@RAND**
A:C26: **+$B$13+($C$13-$B$13)*@RAND**
A:A27: **f(x)**
A:B27: **+$B$6*@EXP(-(B26-$C$6)^(2*$D$6))+$E$6*@EXP(-(B26-$F$6)^(2*$G$6))+$H$6*B26+$I$6+$B$9*(B26-$C $9)^$D$9*@EXP(-$E$9*(B26-$F$9))+$G$9*@SIN($H$9*B26-$I$9)**
A:C27: **+$B$6*@EXP(-(C26-$C$6)^(2*$D$6))+$E$6*@EXP(-(C26-$F$6)^(2*$G$6))+$H$6*C26+$I$6+$B$9*(C26-$C $9)^$D$9*@EXP(-$E$9*(C26-$F$9))+$G$9*@SIN($H$9*C26-$I$9)**
A:A28: **a**
A:B28: **(B22-$DA27)^3**
A:C28: **(C22-$DA27)^3**
A:A29: **F**
A:B29: **@EXP(-$E$15*@EXP($G$15*$B21)*(B27-$CZ27)^(2*$F$15))**
A:C29: **@EXP(-$E$15*@EXP($G$15*$B21)*(C27-$CZ27)^(2*$F$15))**
A:A30: **F/SumF**
A:B30: **+B29/($D$13+$CX29)**
A:C30: **+C29/($D$13+$CX29)**
A:A31: **Curare**
A:B31: **($CZ27-B27)^(1/($D$15+($CZ27-B27)))**
A:C31: **($CZ27-C27)^(1/($D$15+($CZ27-C27)))**
A:A32: **xF-a-xi**
AB32: **+$CZ29-B22**
A:C32: **+$CZ29-C22**
A:A33: **f(xF-a)**
A:B33: **+$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*| $CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)**
A:C33: **+$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*< $CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)**
A:A34: **Sinal**
A:B34: **(B33-B26)/($D$13+(B33-B26))**
A:C34: **(C33-C26)/($D$13+(C33-C26))**
A:A35: **Passo**
A:B35: **+$B$15*@EXP(-($C$15/$B25))**
A:C35: **+$E3$15*@EXP(-($C$15/$B25))**
A:A36: **Cálculo**
A:B36: **+B26+(B35*B31 *B34*B32)**
A:C36: **+C26+(C35*C31*C34*C32)**
A:CX26: **@SUMPRODUCT(B26..CW26.B27..CW27)/$CX27**
A:CZ26: **fmax**
A:DA26: **xF**
A:CX27: **@SUM(B27..CW27)**
A:CZ27: **@MAX(B27..CW27)**
A:DA27: **@SUMPRODUCT(B26 CW26.B30..CW30)**
A:CX28: **(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,B29..C W29)/$CX29))*@ABS(@SUMPRODUCT(B26 ..CW26,B29..CW29)/$CX29)^(1/3)**
A:CZ28: **xF-a**
A:DA28: **f(xF)**
A:CX29: **@SUM(B29..CW29)**
A:CZ29: **+DA23-CX28**
A:DA29: **+$B$6*@EXP(-(DA27-$C$6)^(2*$D$6))+$E$6*@EXP(-(DA27-$F$6)^(2*$G$6))+$H$6*DA27+$I$6+$B$9*(DA 27-$C$9)^$D$9*@EXP(-$E$9*(DA27-$F$9))+$G$9*@SIN($H$9*DA27-l9)**
A:CZ30: **STD x**
A:DA30: **Média**
A:CZ31: **@STD(B26..CW26)**
A:DA31: **@AVG(B26..CW26)**

[illegible]	[illegible]	85.78851	[illegible] 62008	88.4812	89.11368	[illegible]	,97764	91.51008	91.15214	91.37334	44081	44081
[illegible]	[illegible]	35.70287	[illegible]70287	35.70287	36.11075	[illegible]43336	.97646	38.22485	38.26117	38.26552	26552	26552
[illegible]	[illegible]	26.83101	[illegible]	26.84021	26.84133	[illegible]82793	[illegible]	26.75022	26.68683	26.61843	[illegible]49676	[illegible]
[illegible]	Delta [illegible]	3.831572	13888	0.632481	3.102165	[illegible]	[illegible]46756	-0.35795	0.221208	0.067468	[illegible].8E-07	[illegible]
[illegible]	Delta [illegible]	0	0	0.407885	1.322602	D.5431	[illegible]48397	0.036322	0.004342	0	0	0

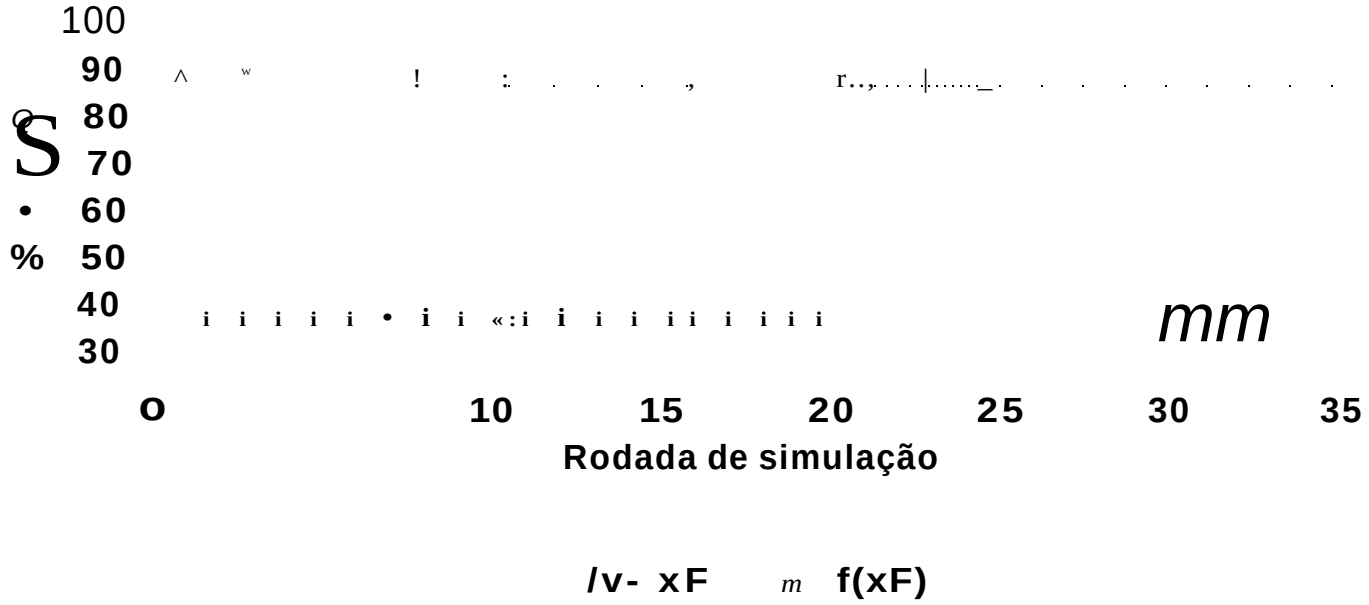
Função de teste unidimensional

Função 4



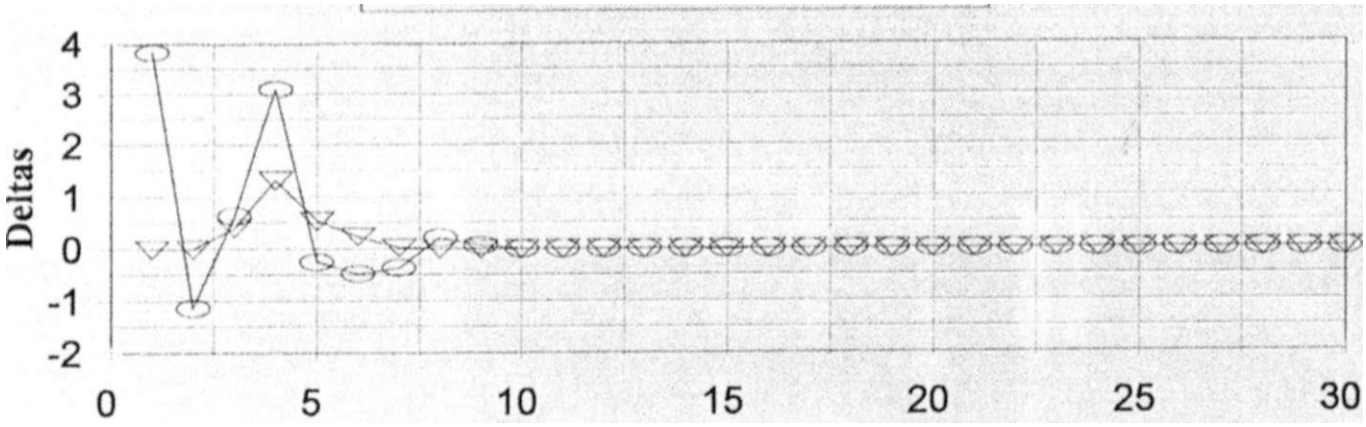
Função de teste 4

Comportamento de xF e f(xF)

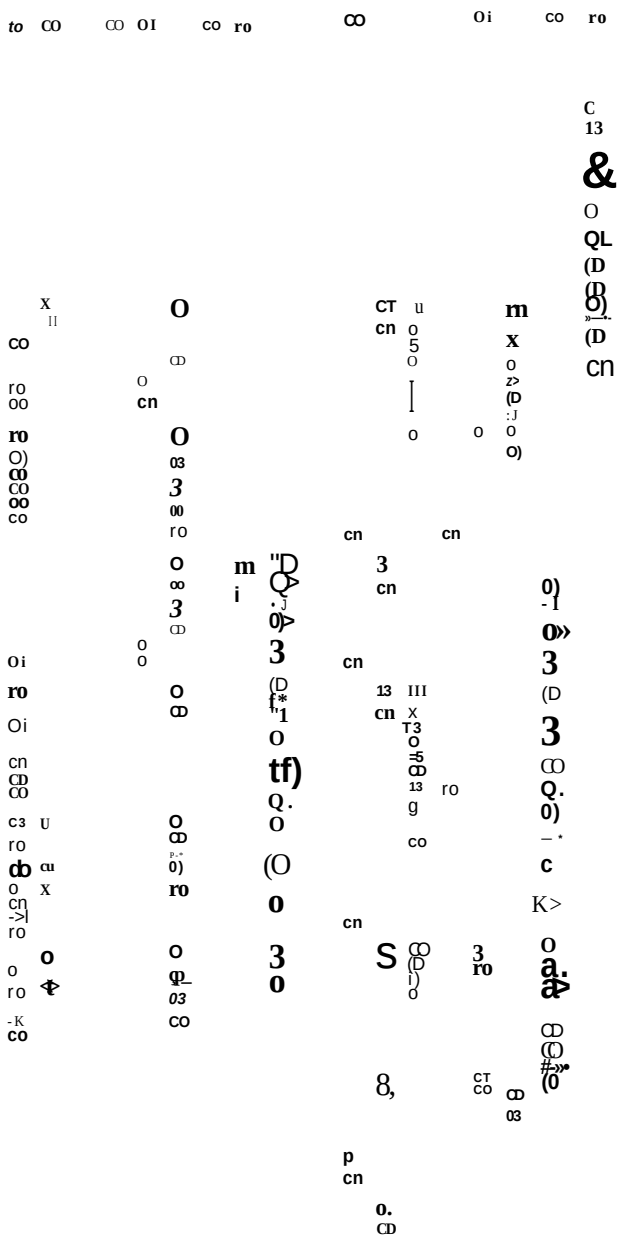


Função de teste 4

Comportamento dos deltas



Delta xF -*S*- Delta f(xF)



Cálculo 57.63039 77.2475 .199354 98.45766 46.78124 26.54918 64.48644

Nos dois teoremas estamos utilizando o comportamento da função em x* . Se entretanto queremos investigar o comportamento da função em alguma vizinhança do mínimo, o seguinte teorema prove condições adicionais para o mínimo local.

Teorema 2.2.1.3: Seja um ponto interior de D e assuma que f é duas vezes continuamente diícrenciável cm D. Para que exista um mínimo local dc f cm x* c necessário que

Vfiy) - 0

c

Z'VfiV)Z >0

V Z ^ O . Condições suficientes para um mínimo local são que Eq. 2.2.1.4 valha c que V x em alguma vizinhança (x*) c para todo Z c- R^n, tenhamos

Eq. 2.2.1.6 / V f (x > . >0

Prova: Ver [Avricl|.

Um último resultado apresenta condições de suficiência para mínimo local estrito baseado cm uma vizinhança dc x\

Teorema 2.2.1.4. Seja x* um ponto interior de D c assuma que f é duas vezes continuamente difcrcnciávcl. Se

VfiV) 0

c

$Z^T V^2 f(x) Z$ >0

V x / x* cm uma vizinhança dc x* c para lodo Z *t*- 0, então f tem um mínimo estrito cm x\

A figura 2.2.1.1 abaixo ilustra como um mínimo pode existir mas não satisfazer as condições necessárias c suficientes.

Contudo quando as condições de suficiência são satisfeitas o mínimo é garantido

AA25: **Iteração**
AB25: **1**
A:A26: **x**
A:B26: **+$B$13+($C$13-$B$13)*@RAND**
A:C26: **+$B$13+($C$13-$B$13)*@RAND**
A:A27: **f(x)**
A:B27: **+$B$6*@EXP(-(B26-$C$6)^(2*$D$6))+$E$6*@EXP(-(B26-$F$6)^(2*$G$6))+$H$6*B26+$I$6+$B$9*(B26-$C$9)^$D$9*@EXP(-$E$9*(B26-$F$9))+$G$9*@SIN($H$9*B26-$I$9)**
A:C27: **+$B$6*@EXP(-(C26-$C$6)^(2*$D$6))+$E$6*@EXP(-(C26-$F$6)^(2*$G$6))+$H$6*C26+$I$6+$B$9*(C26-$C$9)^$D$9*@EXP(-$E$9*(C26-$F$9))+$G$9*@SIN($H$9*C26-$I$9)**
A:A28: **a**
A:B28: **(B22-$DA27)^3**
A.C28: **(C22-$DA27)^3**
A:A29: **F**
A:B29: **@EXP(-$E$15*@EXP($G$15*$B21)*(B27-$CZ27)^(2*$F$15))**
A:C29: **@EXP(-$E$15*@EXP($G$15*$B21)*(C27-$CZ27)^(2*$F$15))**
A:A30: **F/SumF**
A:B30: **+B29/($D$13+$CX29)**
A:C30: **+C29/($D$13+$CX29)**
AA31: **Curare**
A:B31: **($CZ27-B27)^(1/($D$15+($CZ27 B27)))**
A:C31: **($CZ27-C27)^(1/($D$15+($CZ27-C27)))**
AA32: **xF-a-xi**
A:B32: **+$CZ29-B22**
A:C32: **+$CZ29-C22**
A:A33: **f(xF-a)**
A:B33: **+$B$6*@EXP(-($CZ29-$C$6)^Í?*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*($CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)**
A:C33: **+$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9*($CZ?9-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)**
A:A34: **Sinal**
A:B34: **(B33-B26)/($D$13+(B33-B26))**
A:C34: **(C33-C26)/($D$13+(C33-C26))**
A:A35: **Passo**
A:B35: **+$B$15*@EXP(-($C$15/$B25))**
A:C35: **+$R$15*@EXP(-($C$15/$B25))**
AA36: **Cálculo**
A:B36: **+B26+(B35*B3rB34*B32)**
AC36: **+C26+(C35*C31*C34*C32)**
A:CX26: **@SUMPRODUCT(B26.CW26,B27..CW27)/$CX27**
A:CZ26: **fmax**
A:DA26: **xF**
A:CX27: **@SUM(B27..CW27)**
A:CZ27: **@MAX(B27..CW27)**
A:DA27: **@SUMPRODUCT(B26..CW26,B30..CW30)**
A.CX28: **(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)**
A:CZ28: **xF-a**
A:DA28: **f(xF)**
A:CX29: **@SUM(B29..CW29)**
A:CZ29: **+DA23-CX28**
A:DA29: **+$B$6*@EXP(-(DA27-$C$6)^(2*$D$6))+$E$6*@EXP(-(DA27-$F$6)^(2*$G$6))+$H$6*DA27+$I$6+$B$9*(DA27-$C$9)^$D$9*@EXP(-$E$9*(DA27-$F$9))+$G$9*@SIN($H$9*DA27-l9)**
A:CZ30: **ST D x**
A:DA30: **Média**
A:CZ31: **@STD(B26..CW26)**
A:DA31: **@AVG(B26..CW26)**

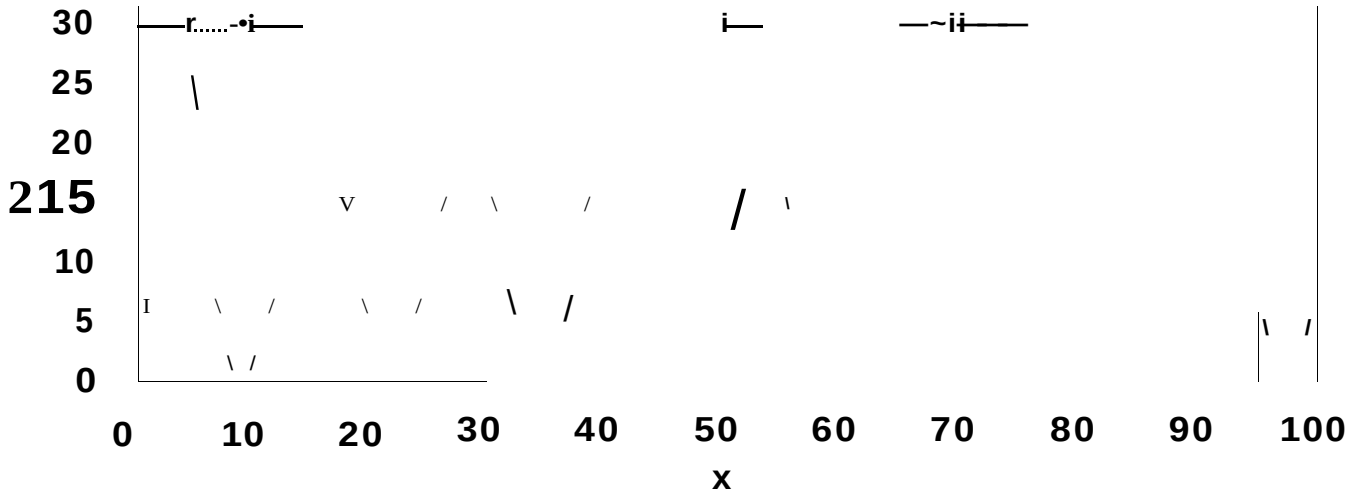
A	J	K	L
14		1	2
15	xF	29.18927	5.587694
16	f(xF)	21.34983	21.34983
17	STD	27.06071	27.04798
18	Delta xF	-23.6016	0.125984
19	Delta f(xF	0	0

A	V	w	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
14	12	13	14	15	16	17	18	19	20	21	22	23
15	5.131303	5.131315	4.75161	4.682284	4.631681	4.765532	4.700805	4.739428	4.739428	4.739428	4.739428	4.739428
16	26.08539	26.08539	26.77906	26.77906	26.77906	26.80753	26.99894	26.99894	26.99894	26.99894	26.99894	26.99894
17	26.84027	26.80249	26.76372	26.71778	26.66459	26.6109	26.55548	26.50047	26.44581	26.38693	26.313	26.22787
18	1.2E-05	-0.3797	-0.06933	-0.0506	0.133851	-0.06473	0.038623	0	C	0	0	0.116882
19	0	0.693668	0	0	0.028473	0.191405	C	0	0	0	0	0

^	AH	^	AJ	AK	AL	^	AN
cn	4.85631	4.767886	4.739428	4.739428	4.739465	4.783369	
	26.99894	26.99894	26.99894	26.99894	26.99394	26.99894	
- 4	26.1173	25.99208	25.74439	25.62297	25.50294	25.37996	
CO	-0.08842	-0.02846		3.7E-05	0.043904	-0.04394	

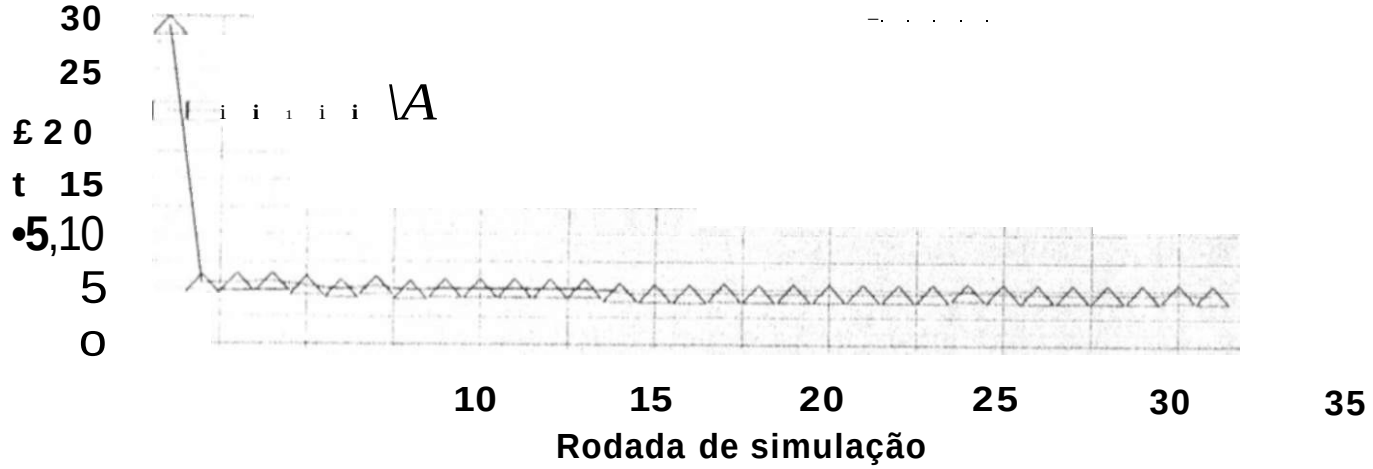
Função de teste unidimensional

Função 5



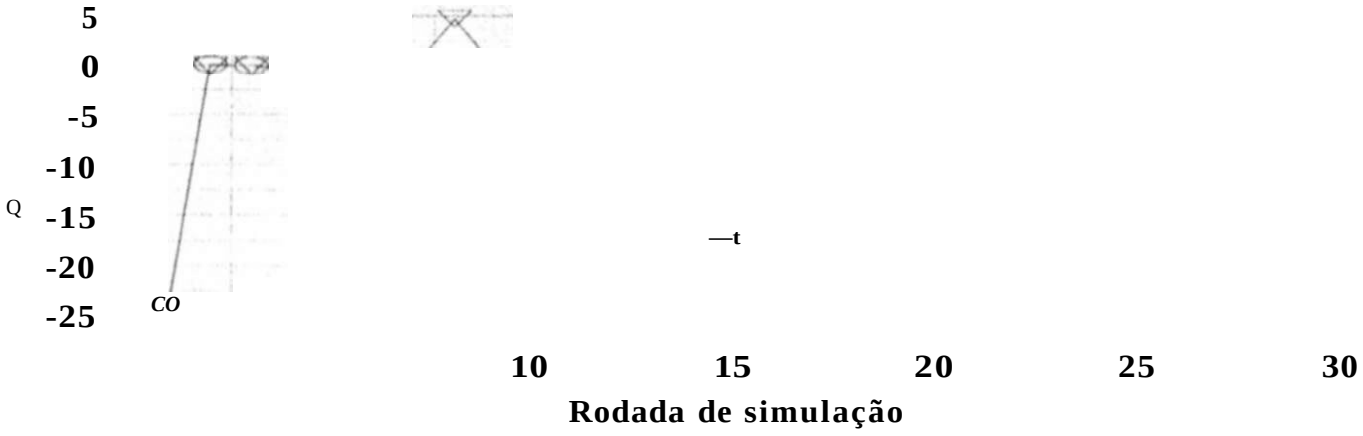
Função de teste 5

Comportamento de xF e f(xF)



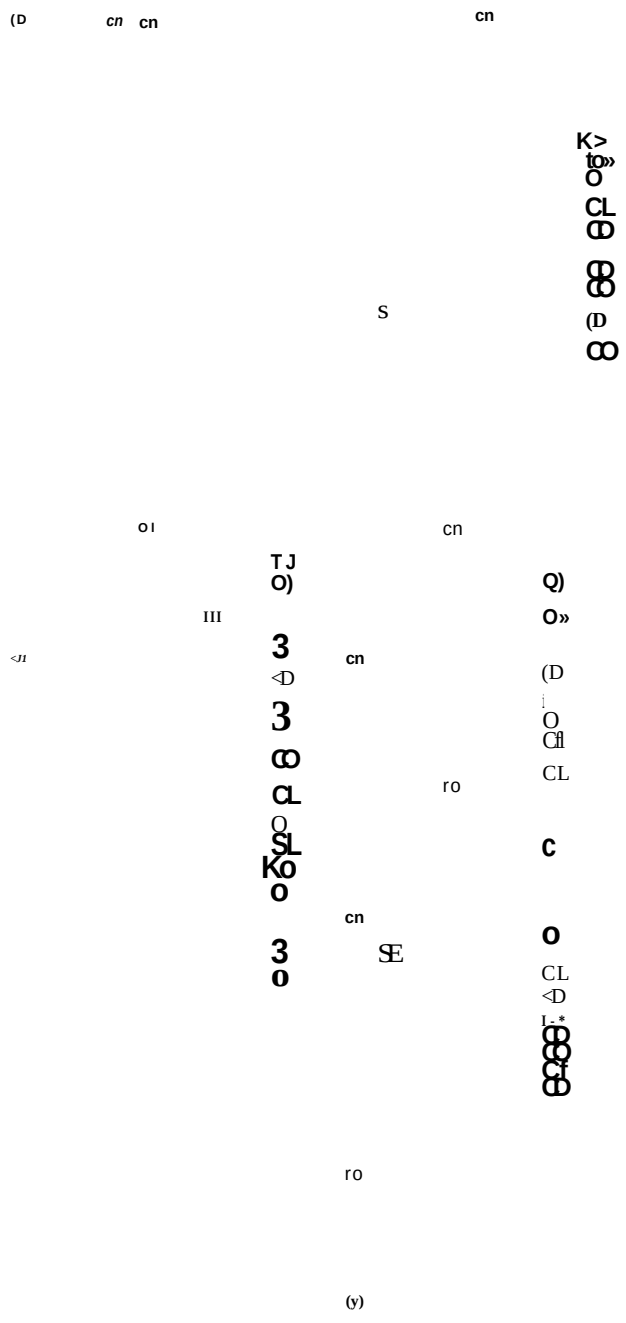
Função de teste 5

Comportamento dos deltas



Delta xF v Delta f(xF)

Anexo G



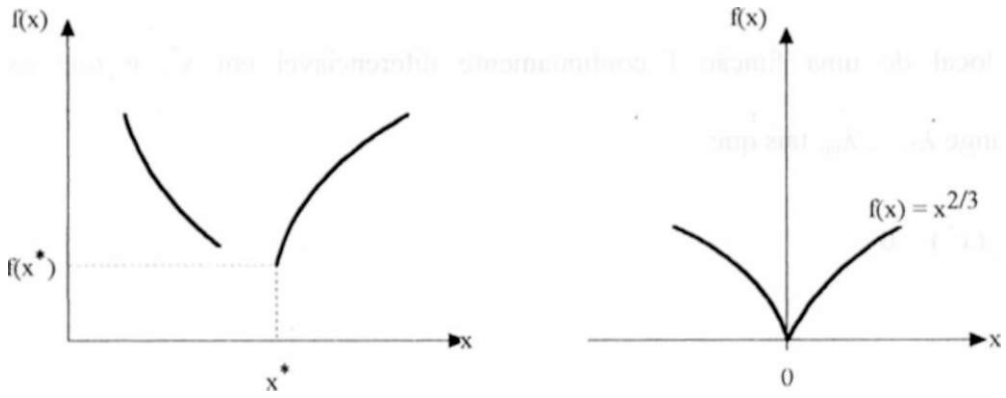


Figura 2.2.1.1 - Exemplo de funções que tem mínimo mas não atendem as condições dc otimalidade.

2.2.2 Caso vinculado

Veremos agora as condições necessárias c as condições de suficiência quando o problema tem restrições dos dois tipos: igualdade e desigualdade. **0** problema então é

Problema 2.2.2.1 Minimizar fi» x c R"

Sujeito à hj(x) 0 j = l, . . . , m

$_{g}$ j (x) > 0 j - m + l, . . . , p

Consideremos primeiro o caso em que gj(x) = 0, V j = m+1,..., p, ou seja, onde só existem restrições de igualdade e suponhamos que as hipóteses (i) c (ii) abaixo valem.

Hipóteses

i) hj e C ,j = 1,..., m.

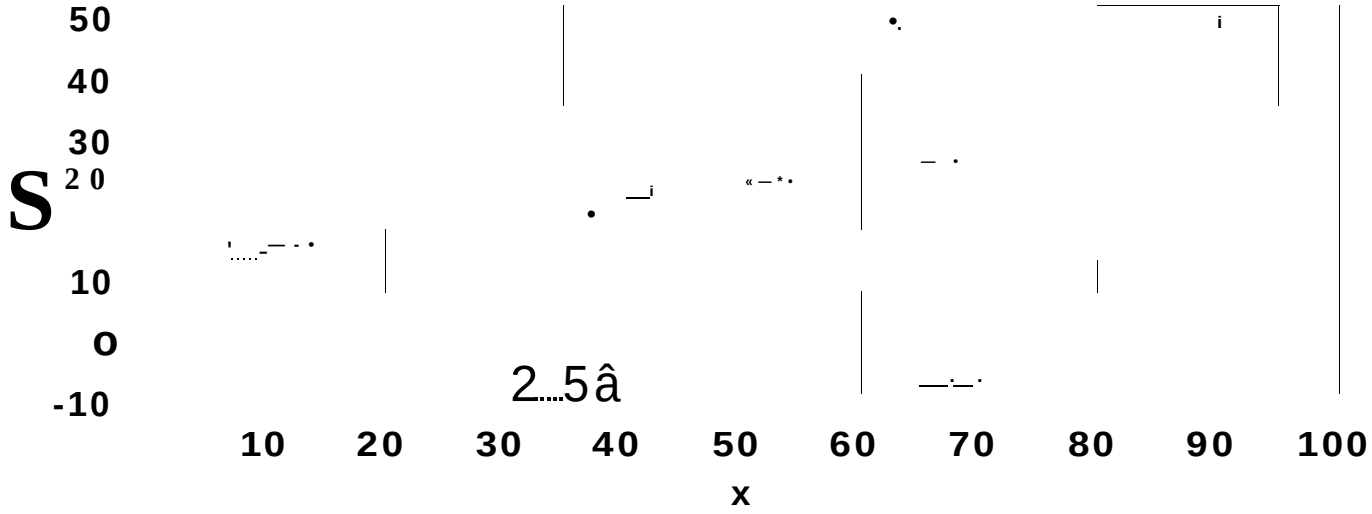
ii) Os gradientes {Vhj(x*), j'-= 1,..., m} são linearmente independentes.

```
AA25:    Iteração
A:B25:   1
A:A26:   x
A:B26:    +$B$13+($C$13-$B$13)*@RAND
A:C26:    +$B$13+($C$13-$B$13)*@RAND
A:A27:   t(x)
A:B27:   •  $B$6·@EXP(-(B26-$C$6)^(2*$D$6))+$E$6*@EXP(-(B26-$F$6)^(2*$G$6))+$H$6*B26+$I$6+$B$9*(B26-$C
         $9)^$D$9*@EXP(-$E$9*(B26-$F$9))+$G$9*@SIN($H$9*B26-$I$9)
A:C27:     +$B$6*@EXP(-(C26-$C$6)^(2*$D$6))+$E$6*@EXP(-(C26-$F$6)^(2*$G$6))+$H$6*C26+$I$6+$B$9*(C26-$(
         $9)^$D$9*@EXP(-$E$9*(C26-$F$9))+$G$9*@SIN($H$9*C26-$I$9)
A:A28:   a
A:B28:    (B22-$DA27)^3
A:C28:    (C22-$DA27)^3
A:A29:   F
A:B29:   @EXP(-$E$15*@EXP($G$15*$B21)*(B27-$CZ27)^(2*$F$15))
A:C29:   @EXP(-$E$15*@EXP($G$15*$B21 )*(C27-$CZ27)^(2*$F$15))
AA30:    F/SumF
AB30:     +B29/($D$13+$CX29)
A:C30:    +C29/($D$13+$CX29)
AA31:    Curare
A:B31:    ($CZ27-B27)^(1/($D$15+($CZ27-B27)))
A:C31:    ($CZ27-C27)^(1/($D$15+($CZ27-C27)))
AA32:    xF-a-xi
AB32:     +$CZ29-B22
A:C32:    +$CZ29-C22
AA33:    f(xF-a)
A:B33:     +$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9
         $CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)
A:C33:     +$B$6*@EXP(-($CZ29-$C$6)^(2*$D$6))+$E$6*@EXP(-($CZ29-$F$6)^(2*$G$6))+$H$6*$CZ29+$I$6+$B$9
         $CZ29-$C$9)^$D$9*@EXP(-$E$9*($CZ29-$F$9))+$G$9*@SIN($H$9*$CZ29-$I$9)
AA34:    Sinal
A:B34:    (B33-B26)/($D$13+(B33-B26))
A:C34:    (C33-C26)/($D$13+(C33-C26))
A:A35:    Passo
AB35:     +$B$15*@FXP(-($C$15/$B25))
A C3Í>:   +$B$15*@EXP(-($C$15/$B2S))
AA36:    Cálculo
AB36:     +B26+(B35*B31*B34*B32)
AC36:     +C26+(C35*C31*C34*C32)
A:CX26:  @SUMPRODUCT(B26..CW26,B27..CW27)/$CX27
A:CZ26:   fmax
ADA26:   xF
A:CX27:  @SUM(B27 .CW27)
A:CZ27:  @MAX(B27..CW27)
A:DA27:  @SUMPRODUCT(B26..CW?6,B30..CW30)
A:CX28: (@SUMPRODUCT(B26..CW26329..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,B29..C
         W29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)
A:CZ28:  xF-a
ADA28:   f(xF)
A:CX29:  @SUM(B29..CW29)
ACZ29:   +DA23-CX28
ADA29:     +$B$6*@EXP(-(DA27-$C$6)^(2*$D$6))+$E$6*@EXP(-(DA27-$F$6)^(2*$G$6))+$H$6*DA27+$I$6+$B$9*(L^
         27-$C$9)^$D$9*@EXP(-$E$9*(DA27-$F$9))+$G$9*@SIN($H$9*DA27-$I$9)
A.CZ30:  STD x
A:DA30:  Média
ACZ31:   @STD(B26. CW26)
A:DA31:  @AVG(B26..CW26)
```

4.994957	4.994957	4.998355	4.991504	4.994957	4.994957	5.014501	4.994957	4.960367	4.994957	4.994957	4.994957	4.994957
41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475	41.59475
29.78276	29.70002	29.615	29.51964	29.41975	29.30155	29.15391	29.00363	28.85284	25.59-29	28.52242	28.34709	
0	0.003398	-0.00685	0 0 0		0.019545	-0.01954	-0.03459	0.034589	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

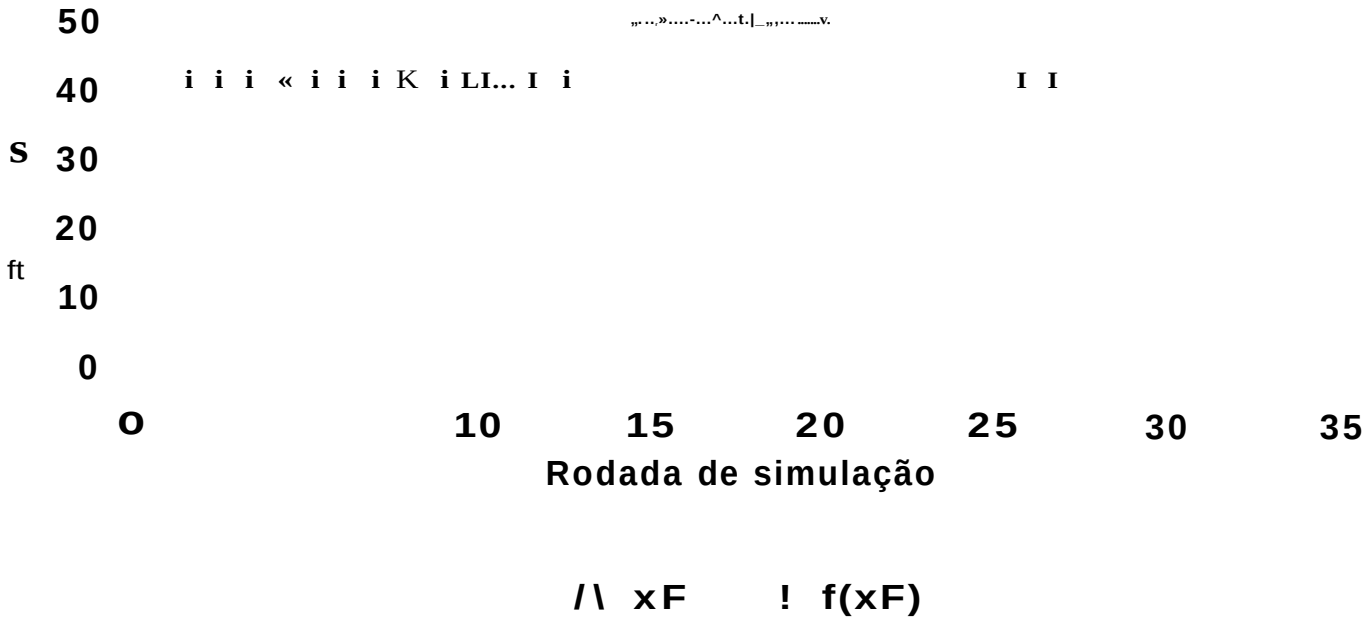
Função de teste unidimensional

Função 1



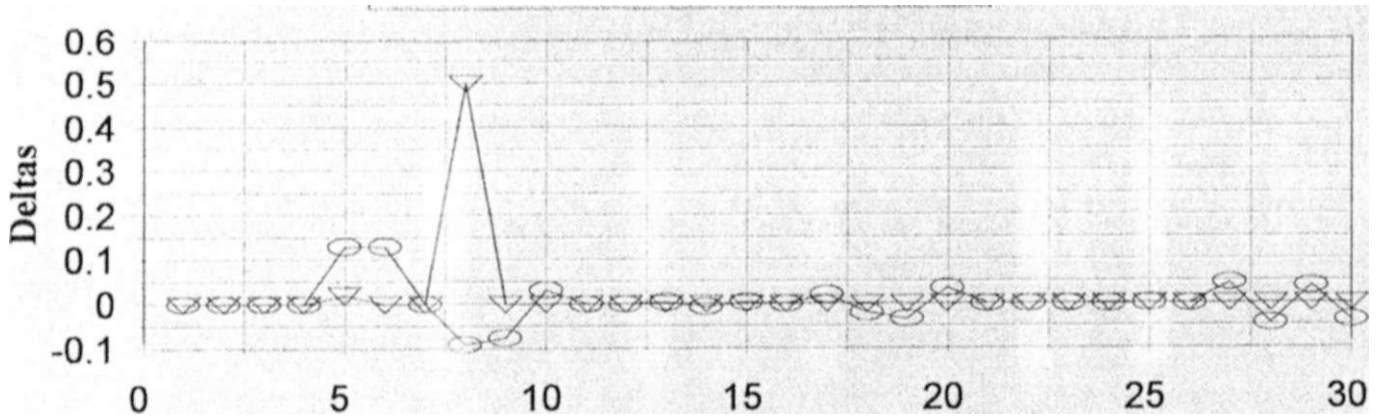
Função de teste 6

Comportamento de xF e f(xF)



Função de teste 6

Comportamento dos deltas



Delta xF v Delta f(xF)

Anexo H

Função universal de 1teste para o caso unidimensinal

Função de te: [illegible]

Parâmetros do algoril:mo

xmin	xmax	Erro			
0.05	[illegible]	1E-10			
Gama 1	Gama 2	Gama 3	Delta 1	Delta 2	Delta 3
[illegible]	[illegible]	[illegible]			
[illegible]	[illegible]		[illegible]	[illegible]	Delta f
0.222376	0.217232		0.223	0.217225	0.000624	-7.7E-06

Teorema 2.2.2.1 - Uma condição necessária para que o ponto x* (R", satisfazendo as hipóteses (i) e (ii), seja um mínimo local de uma função f continuamente diicrcnciávcl em x, c que existam multiplicadores de Lagrange X_{nu} tais que:

v/(x*) i £/t v//,(.0 o

\>0

Xjhj(x*) 0 j I. . . .p

Prova: ver |Mahcy|

l.ssa c uma condição necessária dc primeira ordem (também conhecida como condições de Kuhn-Tuckcr). Agora a condição necessária dc segunda ordem c enunciada como:

Teorema 2.2.2.2 - Suponha x* **(** I) c R", c que as hipóteses (i) c (ii) valem em x\ Suponha ainda que *i* hi, h_n, c C . Então se x* c um mínimo local dc f, existem reais À,, tais que:

a) $\nabla l_x(x,\lambda) = f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla h_i(x^*) = 0,\ \lambda_i \geq 0$ e $\lambda_i h_i(x^*)$ **o**

b) Ax'll(x\X)Ax > **0,** V Ax (M {y < R" : Vh(x*)y ()}, onde **11** é a matriz hessiana dc l,(x,A.) cm relação a x:

I/(x,Ã) V ' / **W i** ^ ^ V ' **X W**

onde M é um subespaço definido como M {y c R" : Vh(x)y 0}.

Prova: ver | Mahcy |

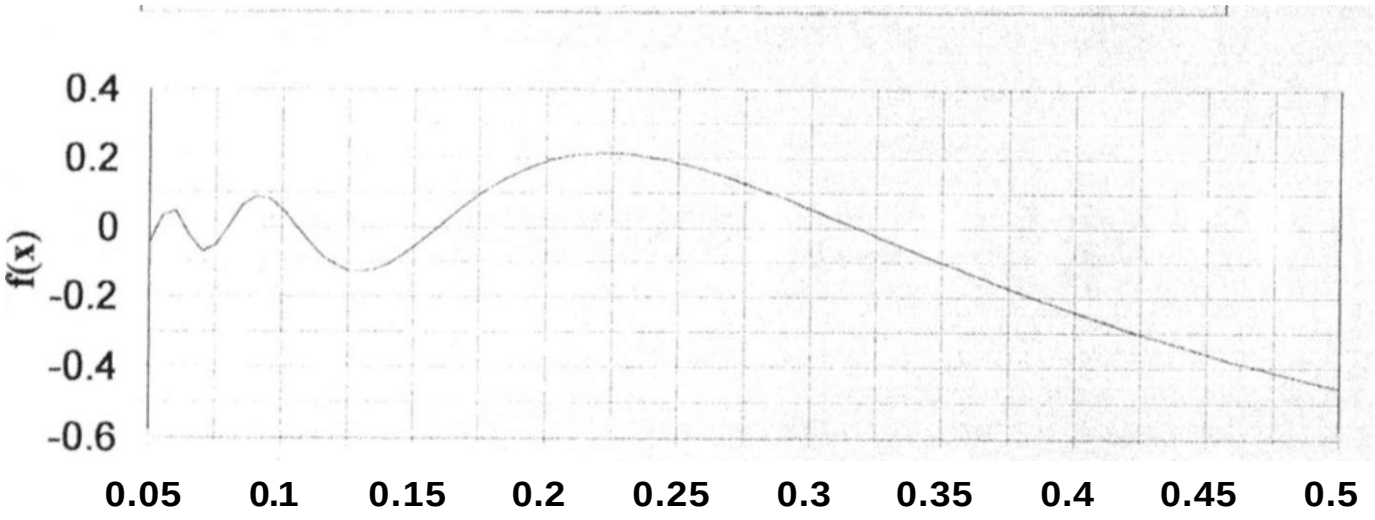
2.3 Métodos determinísticos

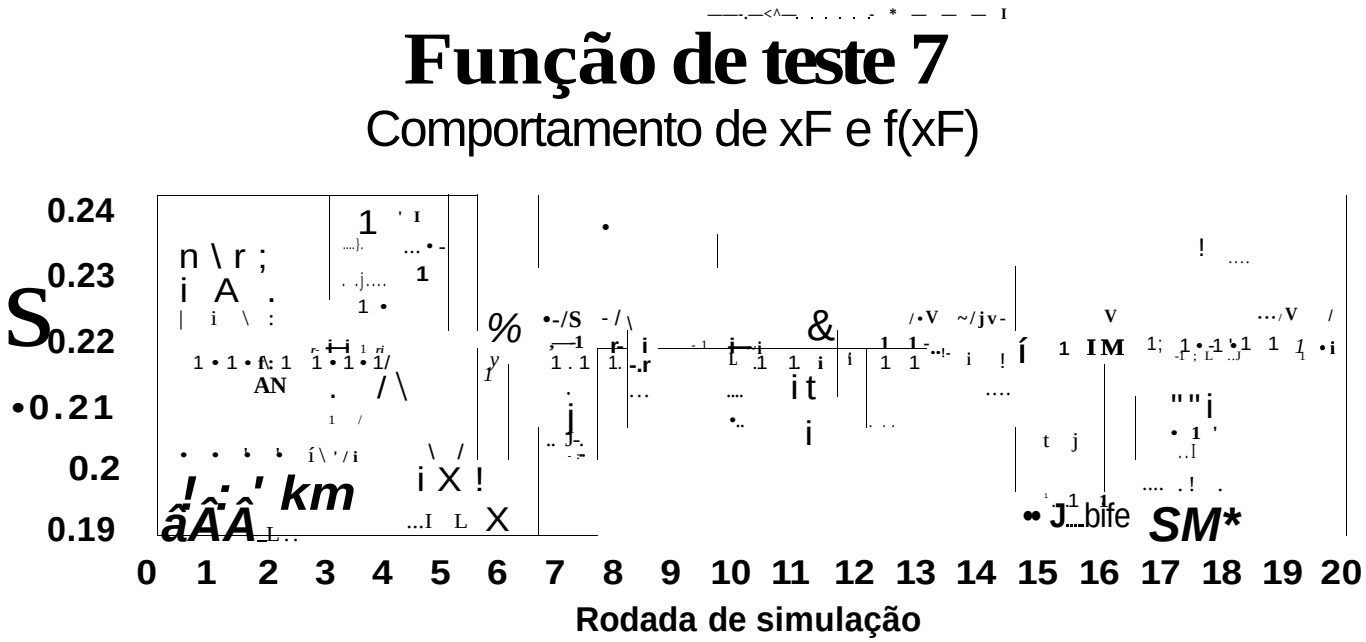
2.3.1 Otimização desvinculada

A:A25: Iteração
A:B25: 1
A:A26: x
A:B26: +$B$13+($C$13-$B$13)*@RAND
A:C26: +$B$13+($C$13-$B$13)*@RAND
A.A27: l(x)
A:B27: ~B26*@SIN(1/($D$13+B26))
A:C27: -C26*@SIN(1/($D$13+C26))
A:A28: a
A:B28: (B26-$DA27)^3
A:C28: (C22-$DA27)^3
A:A29: F
A:B29: @EXP($E$15*@EXP($G$15*$B25)*(B27-$CZ27)^(2*$F$15))
A:C29: @EXP(-$E$15*@EXP($G$15*$B25)*(C27-$CZ27)^(2*$F$15))
A:A30: F/SumF
A:B30: +B29/($D$13+$CX29)
A:C30: +C29/($D$13+$CX29)
A:A31. Curare
A:B31: ($CZ27-B27)^(1 /($D$15+($CZ27-B27)))
A:C31: ($CZ27-C27)^(1/($D$15+($CZ27-C27)))
A:A32: xF-a-xi
A:B32: +$CZ29-B26
A:C32: +$CZ29-C26
AA33: f(xF-a)
A.B33: -$CZ29*@SIN(1 /($D$13+$CZ29))
A:C33: -$CZ29*@SIN(1/($D$13+$CZ29))
A:A34: Sinal
A:B34: (B33-B27)/($D$13+(B33-B27))
A:C34: (C33-C27)/($D$13+(C33 C27))
A:A35: Passo
A:B35: +$B$15*@EXP(-($C$15/$B25))
A:C35: +$B$15*@EXP(-($C$15/$B25))
A:A36: Cálculo
A:B36: +B26+(B35*B3rB34*B32)
A:C36: +C26+(C35*C3rC34*C32)
A:CX?6: @SUMPRODUCT(B26.CW26,B27..CW27)/$CX27
A:CZ26: írnax
A:DA26: xF
A:CX27: @SUM(B27..CW27)
A:CZ27: @MAX(B27..CW27)
A:DA27: @SUMPRODUCT(B26..CW26,B30..CW30)
A:CX28: (@SUMPRODUCT(B26. CW26.B29. CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26. CW26.B29. CW29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)
A:CZ28: xF-a
A:DA28: f(xF)
A:CX29: @SUM(B29. CW29)
A:CZ29: +DA23-CX28
A:DA29: -DA27*@SIN(1/($D$13+DA27))
A:CZ30: STD x
A.DA30: Média
A:CZ31: @STD(B26..CW26)
A:DA31: @AVG(B26 .CW26)

Função de teste unidimensional

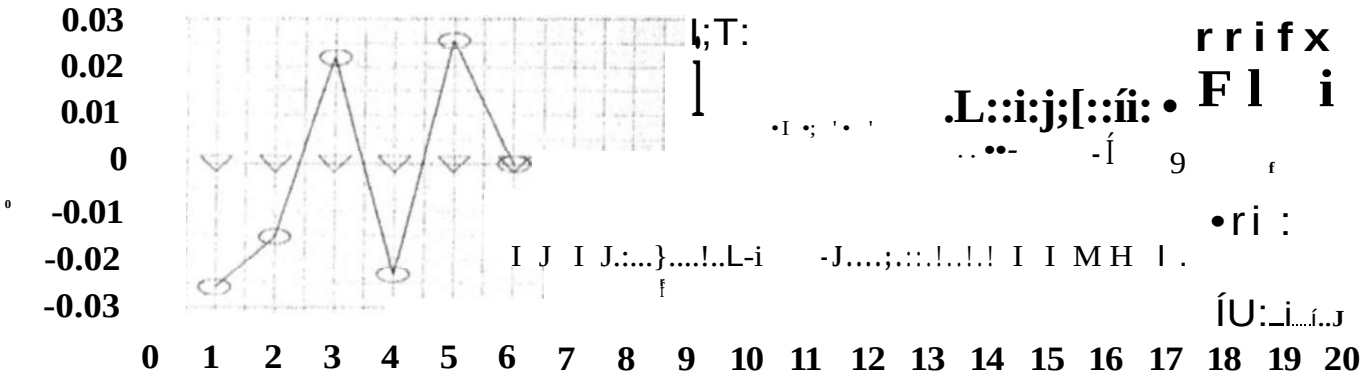
Função 7



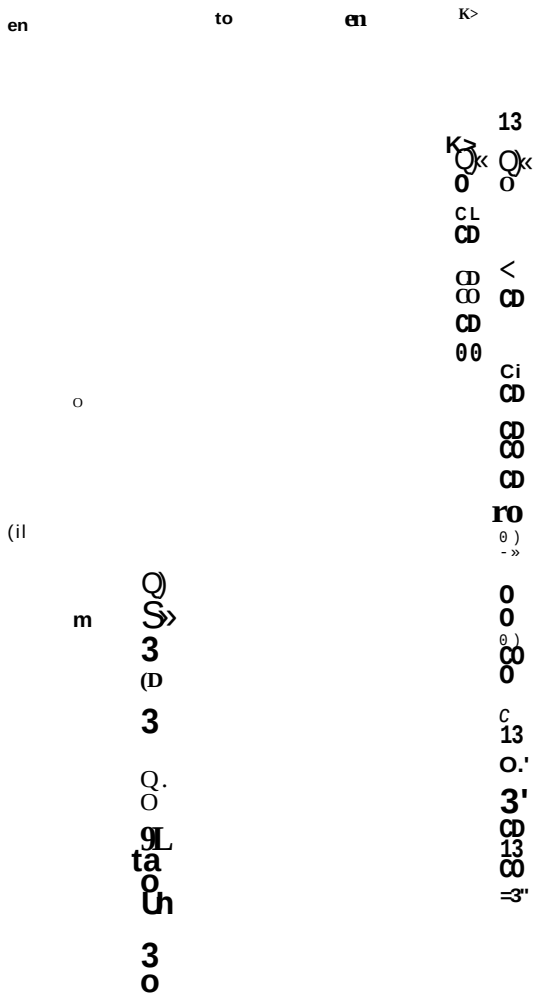

Função de teste 7
Comportamento de xF e f(xF)
0.24
0.23
0.22
0.21
0.2
0.19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Rodada de simulação
/\ xF
f(xF)

Função de teste 7

Comportamento dos deltas



Anexo I



42	F	0.864708	0.845746	0.875449
43	F/SumF	0.009538	0.009329	0.009657
44	Curare	0.980576	0.981273	0.98014
45	xF-a-xi	-1.6718	-0.90924	-0.93099
46	f(xF-a)	-1.43294	-1.43294	-1.43294
47	Sinal	1	1	1
4«	Passo	0.082085	0.082085	0.082085
49	Cálculo	0.739949	0.038719	0.058803

```
A:A25:   Iteração
A:B25:   1
A:A26:   x
A:B26:     +$B$13+($C$13-$B$13)*@RAND
A:C26:     +$B$13+($C$13-$B$13)*@RAND
A:A27:   f(x)
A:B27:    +B26*(1-B26)
A:C27:    +C26*(1-C26)
A:A28:   a
A:B28:    (B26-$DA27)^3
A:C28:    (C26-$DA27)^3
A:A29:   F
A:B29:    @EXP(-$E$15*@EXP($G$15*$B25)*(B27-$CZ27)^(2*$F$15))
A:C29:    @EXP(-$E$15*@EXP($G$15*$B25)*(C27-$CZ27)^(2*$F$15))
A:A30:   F/SumF
A:B30:    +B29/($D$13+$CX29)
A:C30:    +C29/($D$13+$CX29)
A:A31:   Curare
A:B31:     ($CZ27-B27)^(1/($D$15+($CZ27-B27)))
A:C31:     ($CZ27-C27)^(1/($D$15+($CZ27-C27)))
A:A32:   xF-a-xi
A:B32:    +$CZ29-B26
A:C32:    +$CZ29-C26
A:A33:   f(xF-a)
A:B33:    +$CZ29*(1-$CZ29)
A:C33:    +$CZ29*(1-$CZ29)
A:A34:   Sinal
A:B34:     (B33-B27)/($D$13+(B33-B27))
A:C34:    (C33-C27)/($D$13+(C33-C27))
A:A35:   Passo
A:B35:    +$B$15*@EXP(-($C$15/$B25))
A:C35:    +$B$15*@EXP(-($C$15/$B25))
A:A36:   Cálculo
A:B36:     +B26+(B35*B31*B34*B32)
A:C36:    +C26+(C35*C31*C34*C32)
A:CX26:  @SUMPRODUCT(B26..CW26,B27..CW27)/$CX27
A:CZ26:   fmax
A:DA26:  xF
A:CX27:   @SUM(B27..CW27)
A:CZ27:   @MAX(B27..CW27)
A:DA27:  @SUMPRODUCT(B26..CW26,B30..CW30)
A:CX28:   (@SUMPRODUCT(B26..CW26329..CW29)/$CX29)/($D$13+@ABS(@SUMPRODUCT(B26..CW26,^
          W29)/$CX29))*@ABS(@SUMPRODUCT(B26..CW26,B29..CW29)/$CX29)^(1/3)
A:CZ28:  xF-a
A:DA28:  f(xF)
A:CX29:   @SUM(B29..CW29)
A:CZ29:   +DA23-CX28
A:DA29:    +DA27*(1-DA27)
A:CX30:   @SUM(B30..CW30)
A:CZ30:  STD x
A:DA30:  Média
A:CZ31:   @STD(B26..CW26)
A.DA31:   @AVG(B26..CW26)
```

2.3.1.1 Métodos sem derivadas

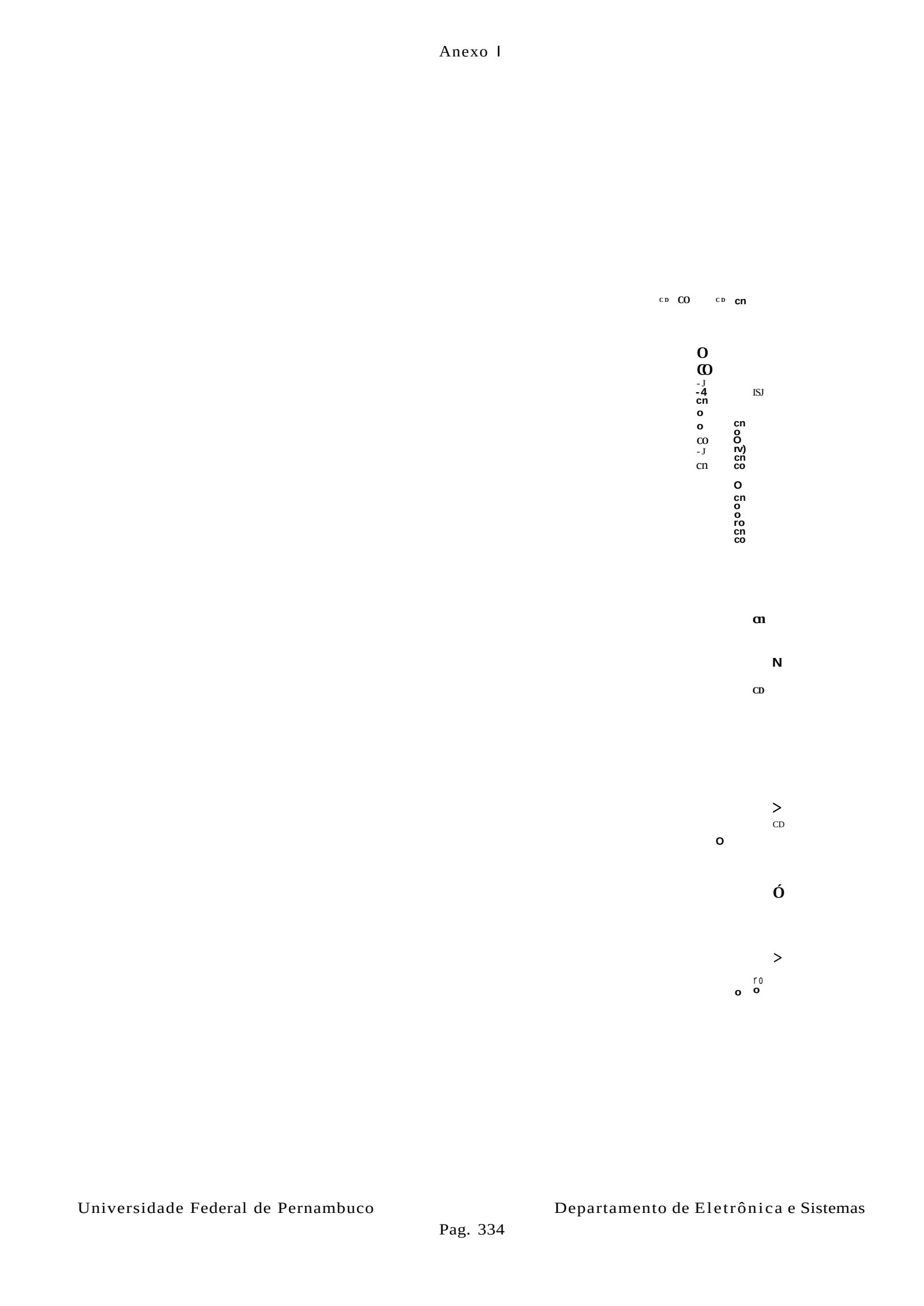
Existe uma gama muito grande de métodos numéricos destinados **à** resolução do problema de otimização desvinculada, trataremos aqui dos métodos que não necessitam de derivadas. Km muitos casos não sc conhece **explicitamente** a expressão analítica da função objetivo (esses valores podem ser calculados somente **ponto** a ponto) ou conhecemos sua expressão analítica porém o cálculo do gradiente é extremamente penoso face **a** complexidade da função ou ainda quando sc quer obter facilidade no preparo das informações iniciais advindas da não utilização dc gradientes, nestas circunstancias os métodos que não utilizam derivadas são preferíveis aqueles que as usam. **Embora,** em geral, a eficiência desses métodos seja inferior a dos que usam gradientes existem algoritmos que não usam derivadas cujo desempenho pode ser considerado excelente c mesmo superior ao dc vários daqueles que delas fazem uso, ver | Ribeiro|.

Trataremos agora dos métodos que utilizam comparações dc valores da função cm vários pontos dc um intervalo. A idéia implícita nesse tipo dc método (comumente chamados métodos dc otimização por procura) é a dc reduzir, a cada iteração, o tamanho do intervalo inicial até um valor considerado satisfatório, ('orno regra geral na solução desse tipo dc problema métodos que utilizam gradiente e derivada segunda convergem mais rapidamente que os métodos dc procura, ver [Himmelblau], Segundo ainda 11 limmelblau| pelas dificuldades de implementação dos métodos com derivadas c pelo lato dos métodos sem derivadas não requerer regularidade c continuidade da função objetivo esles são preferíveis aqueles, do ponto dc vista do usuário.

Apresentaremos, sucintamente, as características mais importantes dc alguns desses métodos,

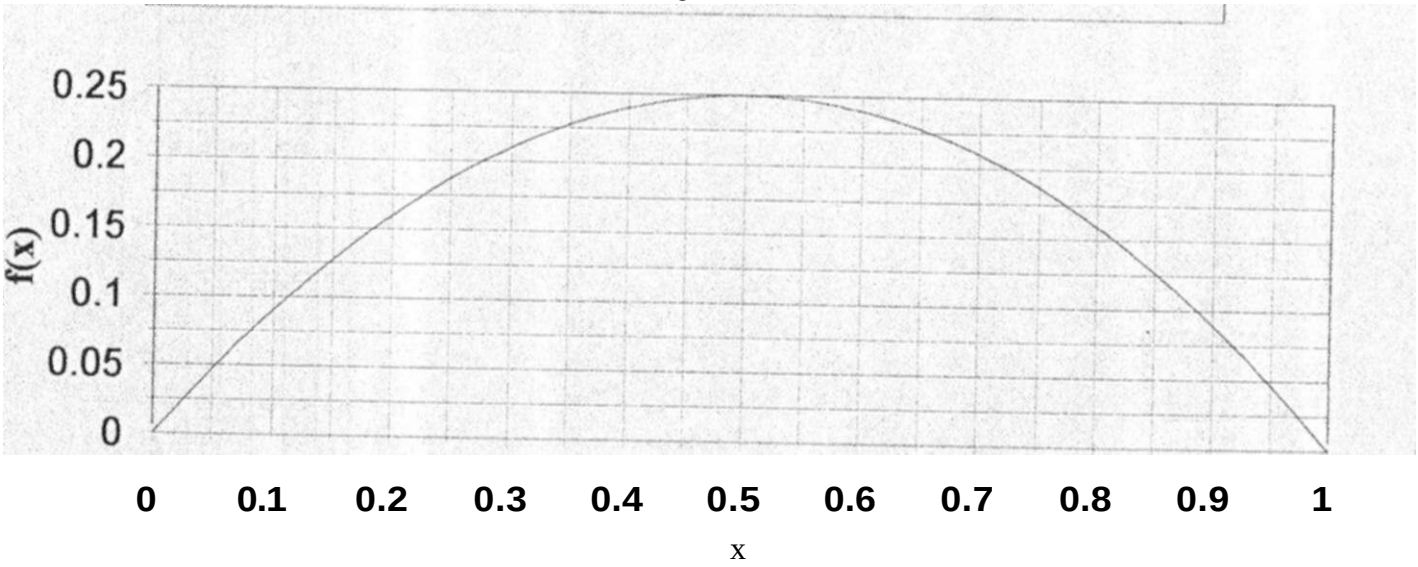
i) Procura direta

Dc grande simplicidade conceituai, o método dc procura direta é implementado alterando o valor dc uma variável ao longo do tempo enquanto as outras são mantidas constantes. Até um mínimo ser



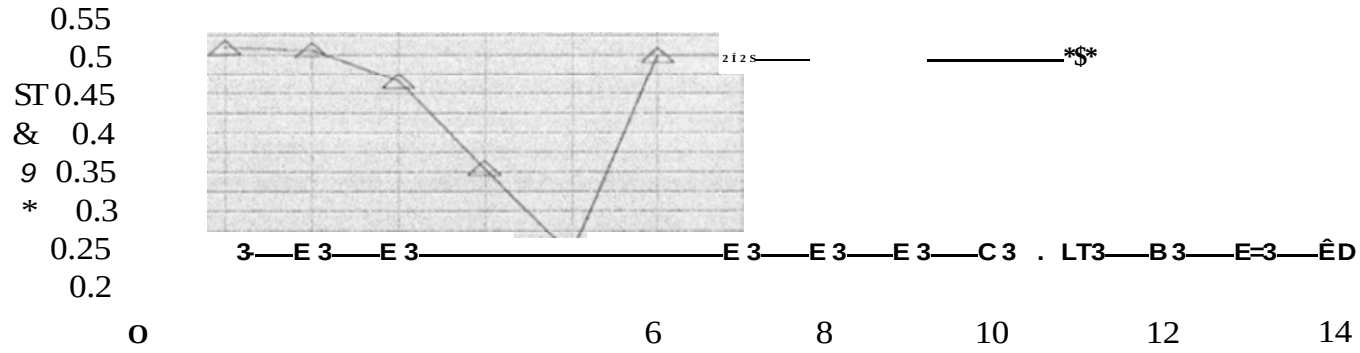
Função de teste unidimensional

Função 8



Função de teste 8

Comportamento de xF e f(xF)



xF - s f(xF)

Função de teste 8

Comportamento dos deltas

Delta xF Delta f(xF)

Anexo J

A	A	B	C	O	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	7.970714	4.58314	9.429739	0.303089	8.510445	4.985771	2.936625	9.967702	Resultado da simulação		
24	x2	7.419844	7.382215	0.494226	6.008752	4.291287	4.851049	2.908435		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1,x2)	-314909	-18571.3	-781982	-3501.44	-3.5E+07				0.803888	0.438914	4 336697
26	(x1-x1fk)^3	368.1125	53.97812	641.8091	-0.1256	3.6E-46				a1k	a2k	fmax
27	(x2-x2fk)^3	340.2044	334.7326	0.000169	172.7937	3.9E-47				3.6E-46	3.9E-47	-4.3367
28	Fk(x1,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					0.803888	0.438914	
30	Curare	1.00004	1.000527	1.000017	1.002271							
31	x1Fk-a1k-x1ik	-7.16683	-3.77925	-8.62585	0.500798							
32	x2Fk-a2k-x2ik	-6.98093	-6.9433	-0.05531	-5.56984							
33	f(x1Fk-a1k,x2ik)	-4588.22	-4537.38	-2.34917	-2875.7							
34	f(x1ik,x2Fk-a2k)	-398126	-42310	-782961	-12.5301							
35	Sinal 1	1	1	1	1							
36	Sinal 2	-1	-1	-1	1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	7.946568	4.570401	9.400678	0.30478							
39	Cálculo 2	7.443364	7.405619	0.494412	5.989945							
40												
41	Iteração	2										
42	x1	7.946568	4.570401	9.400678	0.30478	8.484913	4.973025	2.925602	9.936829	Resultado da simulação		
43	x2	7.443364	7.405619	0.494412	5.989945	4.301244	4.860457	2.913951		x1Fk	x2Fk	f(x1Fk,x2F
44	f(x1 ,x2)	-310348	-18191.7	-772331	-3478.01	-3.5E+07				0.803888	0.438914	4.336697
45	(x1-x1fk)^3	364.4044	53.43412	635.3441	-0.12433	3.6E-46				a1k	a2k	fmax
46	(x2-x2fk)^3	343.6546	338.1289	0.000171	171.0492	3.9E-47				3.6E-46	3.9E-47	-4.3367
47	Fk(x1 ,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					0.803888	0.438914	
49	Curare	1.000041	1.000536	1.000018	1.002284							
50	x1Fk-a1k-x1ik	-7.14268	-3.76651	-8.59679	0.499107							
51	x2Fk-a2k-x2ik	-7.00445	-6.9667	-0.0555	-5.55103							
52	f(x1Fk-a1k,x2ik)	-4620.13	-4568.96	-2.3435	-2855.56							
53	f(x1ik,x2Fk-a2k)	-393290	-41831.6	-773307	-12.4565							
54	Sinal 1	1	1	1	1							
55	Sinal 2	-1	-1	-1	1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	7.653402	4.415731	9.047838	0.325312							
58	Cálculo 2	7.730856	7.691703	0.49669	5.761596							

A:A22: **Iteração**
A:B22: **1**
A:A23: x1
A:B23: **+$B$9+($B$10-$B$9)*@RAND**
A:C23: **+$B$9+($B$10-$B$9)*@RAND**
A:A24: **x2**
A:B24: **+$B$11+($B$12-$B$11)*@RAND**
A:C24: **+$B$11+($B$12-$B$11)*@RAND**
A:A25: **f(x1 ,x2)**
A:B25: **-100*(B24-B23^2)^2-(1 -B23)^2**
A:C25: **-100*(C24-C23^2)^2-(1 -C23)^2**
A:A26: **'(x1-x1fk)^3**
A:B26: **(B23-$GY25)"3**
A:C26: **(C23-$GY25)**
A:A27: **'(x2-x2fk)^3**
A:B27: **(B24-$GZ25)^3**
A:C27: **(C24-$GZ25)^3**
A:A28: **Fk(x1 ,x2)**
A:B28: **@EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))**
A:C28. **@EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))**
A:A29: **Fk/SumFk**
A:B29: **+B28/($D$14+$GU28)**
A:C29: **+C28/($D$14+$GU28)**
A:A30: **Curare**
A:B30: **($HA27-B25)^(1/($F$10+($HA27-B25)))**
A:C30: **($HA27-C25)^(1 /($F$10+($HA27-C25)))**
A:A31: **x1Fk-a1k-x1ik**
A:B31: **+$GY29-B23**
A:C31: **+$GY29-C23**
A:A32: **x2Fk-a2k-x2ik**
A:B32: **+$GZ29-B24**
A:C32: **+SGZ29-C24**
A:A33: **f(x1Fk-a1k,x2ik)**
A:B33: **-100*(B24-$G Y29^2)^2-(1 -$GY29)^2**
A:C33: **-100*(C24-$GY29^2)^2-(1 -$GY29)^2**
A:A34: **f(x1ik,x2Fk-a2k)**
A:B34: **-100*($GZ29-B23^2)^2-(1-B23)^2**
A:C34: **-100*($GZ29-C23^2)^2-(1 -C23)^2**
A:A35: **Sinal 1**
A:B35: **(B33-B25)/($D$14+@ABS(B33-B25))**
A:C35: **(C33-C25)/($D$14+@ABS(C33-C25))**
A:A36: **Sinal 2**
A:B36: **(B34-B25)/($D$14+@ABS(B34-B25))**
A:C36: **(C34-C25)/($D$14+@ABS(C34-C25))**
A:A37: **Passo**
A:B37: **+$D$10*@EXP(-($E$10/$B22))**
A:C37: **+$D$10*@EXP(-($E$10/$B22))**
A:A38: **Cálculo 1**
A:B38: **+B23+B37*B30*B35*B31**
A:C38: **+C23+C37*C30*C35*C31**
A:A39: **Cálculo 2**
A:B39: **+B24+B37*B30*B36*B32**
A:C39: **+C24+C37*C30*C36*C32**
A.GU23: **@SUMPRODUCT(B23..GT23,B25..GT25)/$GU25**
A:GV23: **@AVG(B23..GT23)**
A:GW23: **@STD(B23..GT23)**
A:GX23: **@MAX(B23..GT23)**

A:GY23: **Resultado da simulação**
A:GU24: **@SUMPRODUCT(B24..GT24,B25..GT25)/$GU25**
A:GV24: **@AVG(B24..GT24)**
A:GW24: **@STD(B24..GT24)**
A.GY24: **x1Fk**
A:GZ24: **x2Fk**
A:HA24: **f(x1Fk,x2Fk)**
A:GU25: **@SUM(B25..GT25)**
A:GY25: **@SUMPRODUCT(B23..GT23,B29..GT29)**
A:GZ25: **@SUMPRODUCT(B24..GT24,B29..GT29)**
A:HA25: **+ 100*(GZ25-GY25^2)^2+(1-GY25)^2**
A:GU26: **(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)**
A:GY26: **a1k**
A:GZ26: **a2k**
A:HA26: **fmax**
A:GU27: **(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)**
A:GY27: **+GU26**
A:GZ27: **+GU27**
A:HA27: **@MAX(B25..GT25)**
A:GU28: **@SUM(B28..GT28)**
A:GY28: **x1Fk-a1k**
A:GZ28: **x2Fk-a2k**
A:GY29. **+GY25-GY27**
A:GZ29: **+GZ25-GZ27**

alcançado Por exemplo, em f(**xi**,x,) manier **X|** constante e variar x> ate que um mínimo seja obtido mantém-se agora \> fixo e varia-se agora **xj** até encontrarmos outro valor mínimo

Evidentemente esse método de busca direta pura como apresentado acima é de pouca eficiência . contudo as suas variantes procuram suprir suas maiores ineficiências, ver [Himmelblau] e [Avriel],

ii) Método de Pibonacci (seção áurea)

Esse método não necessita da informação sobre a derivada primeira de f(x), portanto não exige que a função objetivo seja diferenciável, entretanto supõe f(x) convexa, pois se tal não ocorrer nada se pode afirmar sobre o valor iterado x_k **i** a partir de **xi.**

0 processo é o seguinte, dados: uma precisão *í:* > **0,** um ponto inicial x() e uma direção de descida S,

ao longo da qual deve ser realizada a busca, procura determinar um *X'* > 0 tal que , onde *X**

0 c um valor de *X* tal que

f(x ı *X.S)* min {f(x ı *XS)}.

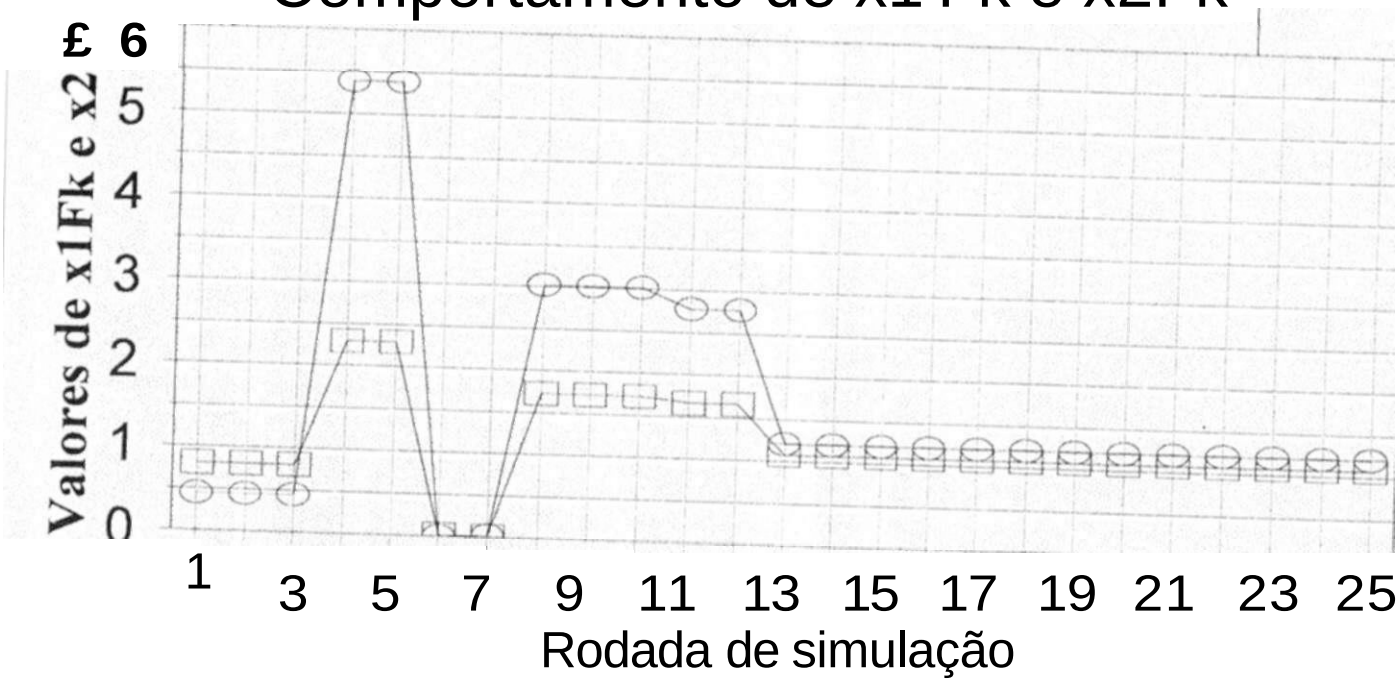
O método determina um intervalo inicial |a,b| tal que *X** (|a,b| e a partir de divisões sucessivas desse intervalo, de valor x (divisão áurea)

ii) Método de Davies - Swann - Campey - Powcll

Composto a partir da fusão de dois outros algoritmos, um devido a Davies - Swann - Campey e o outro devido a Powcll, determina dentro de uma precisão preestabelecida, um valor *X* aproximado para um ponto de mínimo unidirecionaJ *X** usando extrapolação e interpolação.

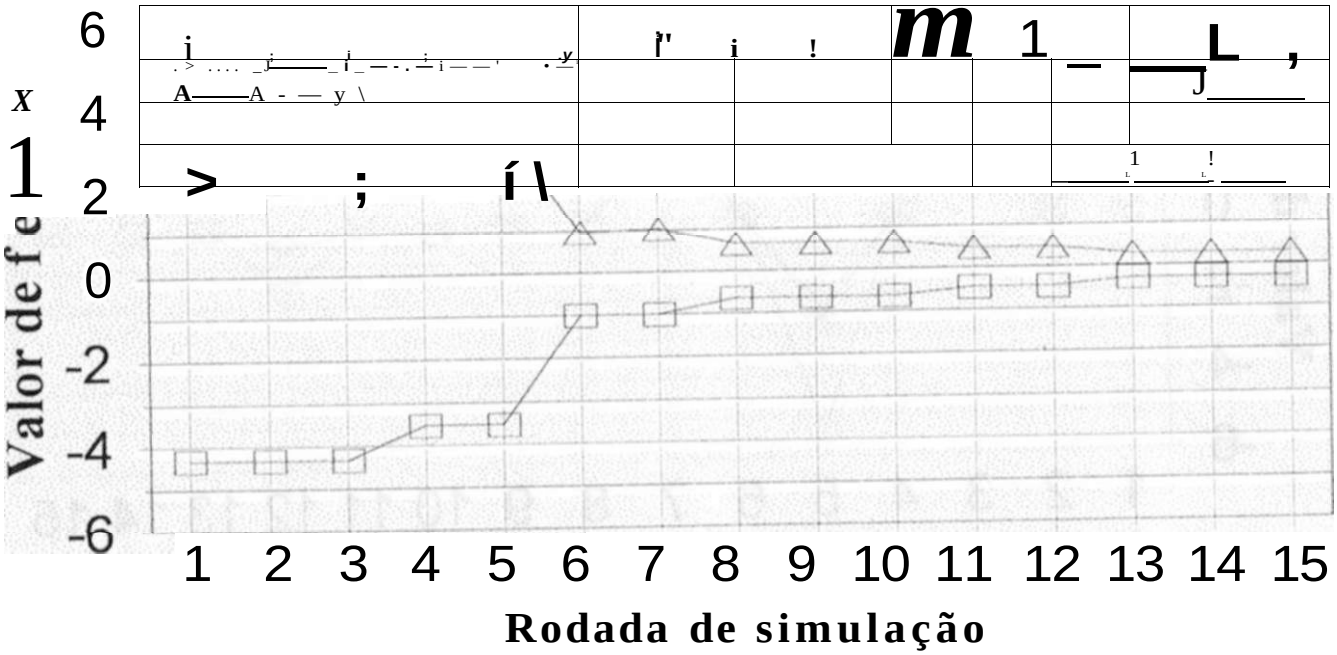
A	J	K	L	M	N	0	P	Q	R	S	T	U
3	Rodada -->	1	2	3	4	5	6	7	8	9	10	11
4	x1Fk	0.803888	0.803888	0.803888	2.2912	2.2912	0.019061	0.010002	1.726389	1.726389	1.726389	1.662661
5	x2Fk	0.438914	0.438914	0.438914	5.386331	5.386331	0	0.020139	3.013759	3.013759	3.013759	2.781668
6	fmax	-4.3367	-4.3367	-4.3367	-3.53686	-3.53686	-1	-1	-0.63881	-0.63881	-0.63881	-0.4688
7	f(x1Fk,x2Fk]	4.336697	4.336697	4.336697	3.536856	3.536856	0.962255	1.020251	0.638807	0.638807	0.638807	0.468799
8	Delta x1Fk	0	0	1.487312	-8.2E-10	-2.27214	-0.00906	1.716386	0	0	-0.06373	0
9	Delta x2Fk	0	0	4.947418	-4.2E-09	-5.38633	0.020139	2.99362	0	0	-0.23209	0
10	Delta f	0	0	-0.79984	-1.5E-08	-2.5746	0.057996	-0.38144	0	0	-0.17001	0

Função de teste 9



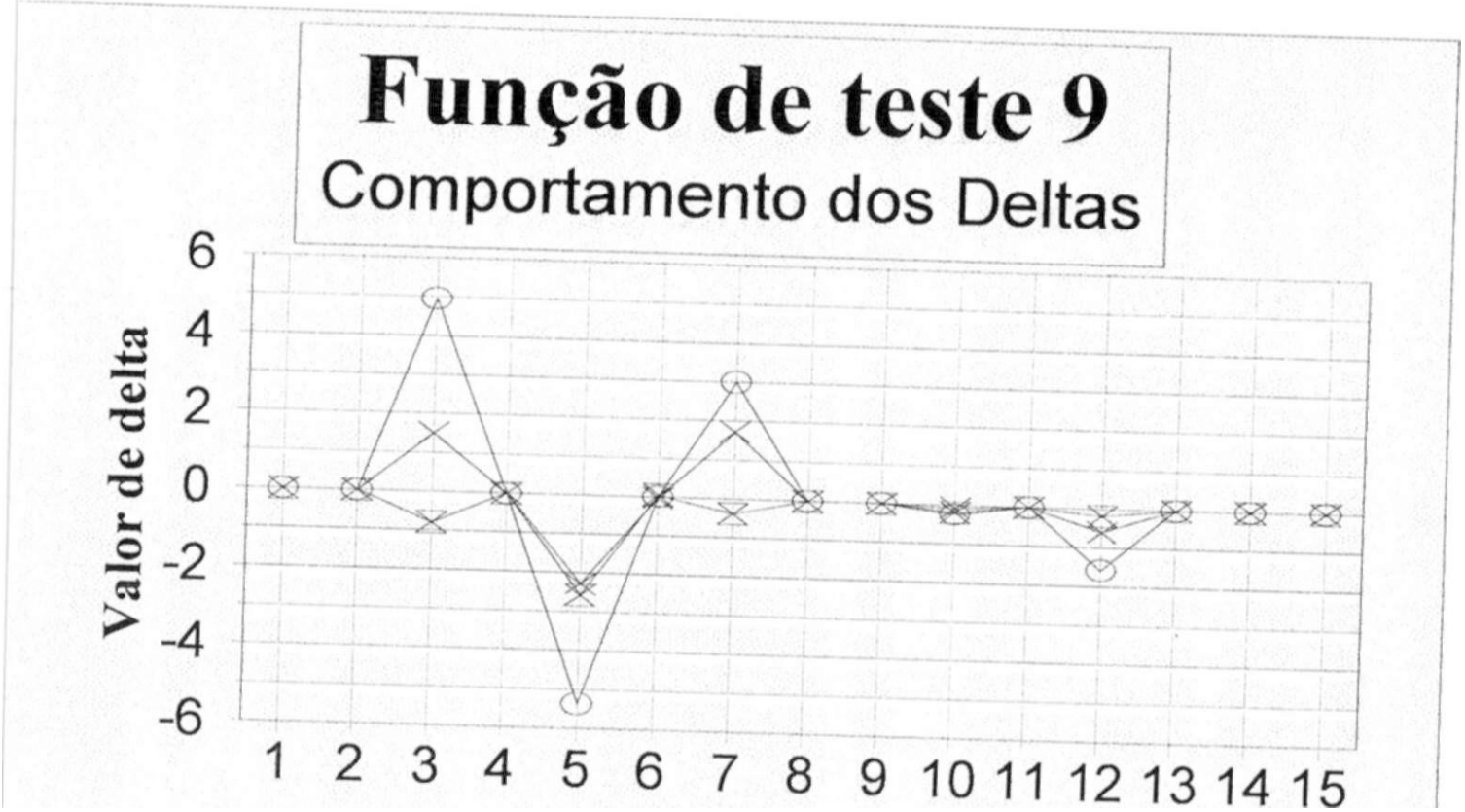
Função de teste 9

Comportamento de f e fmax



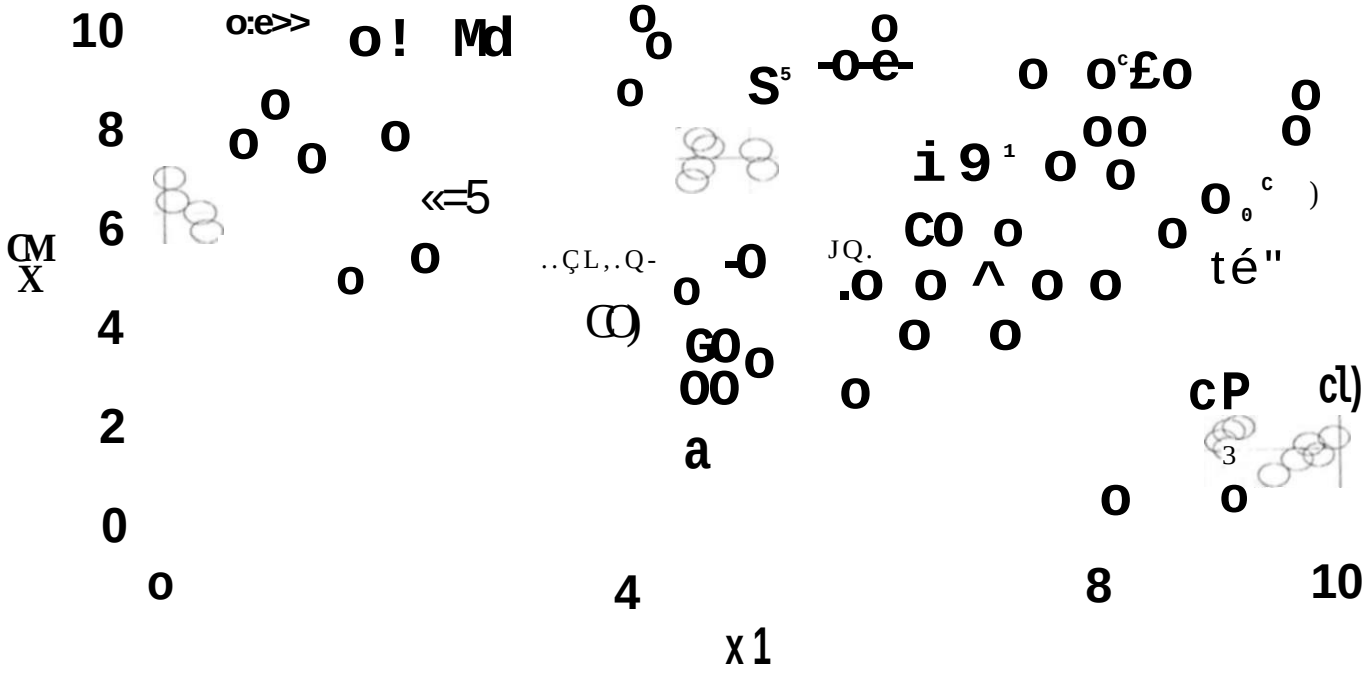
Valor de fmax

-br Valor de f(x1Fk,x2Fk)


Função de teste 9
Comportamento dos Deltas
Valor de delta
6
4
2
0
-2
-4
-6
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Rodada de simulação
x Delta x1Fk
0 Delta x2Fk
Delta f(x1Fk,x2Fk)

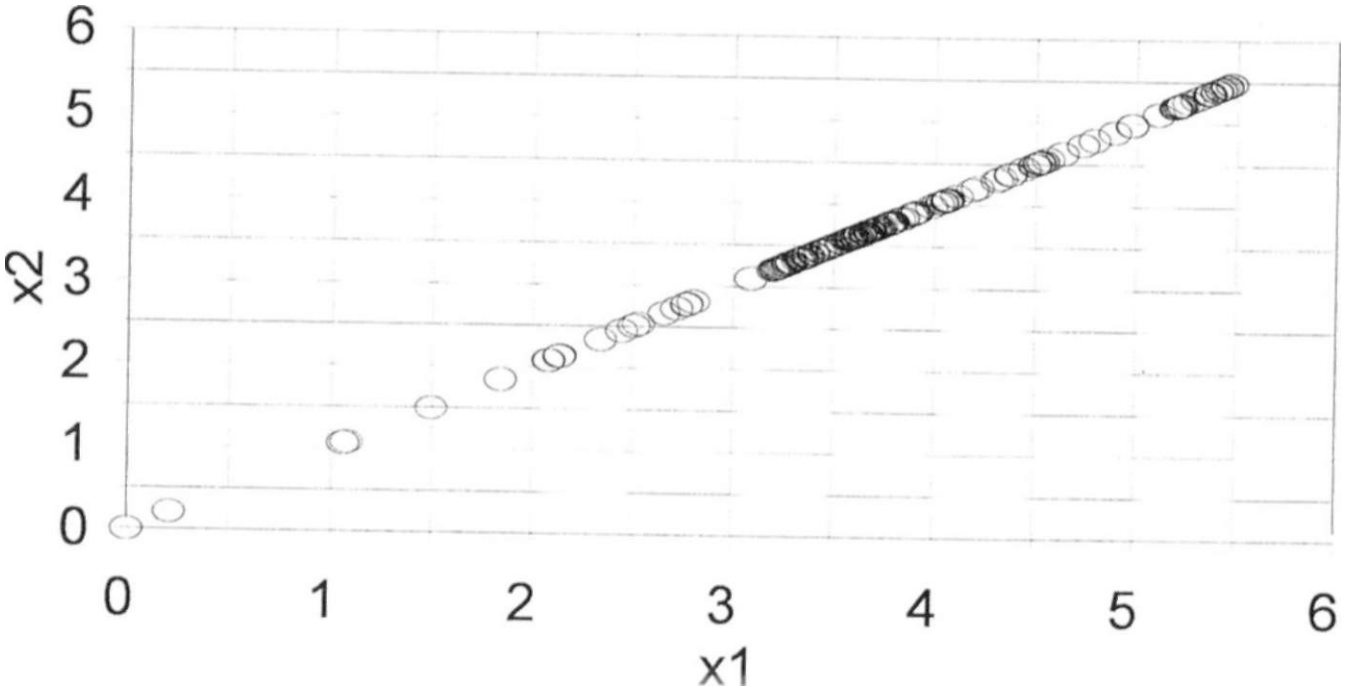
Nuvem de pontos

Primeira rodada de simulação

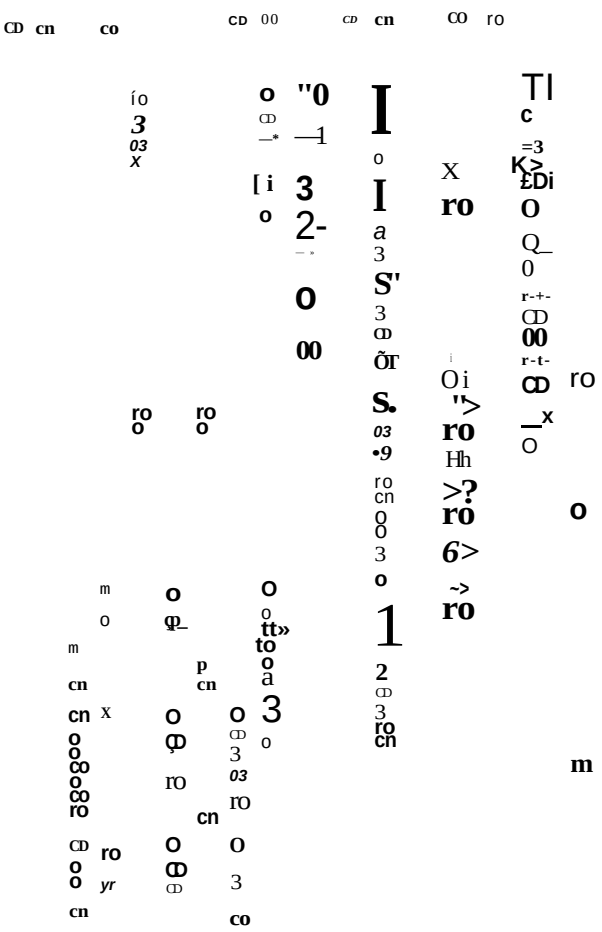


Nuvem de pontos

Vigésima rodada de simulação



Anexo L



Esse método não precisa da informação sobre o gradiente de f(x), entretanto precisa da **hipótese** dc convexidade para a função objetivo, de modo a se poder garantir que o ponto iterado x ' não seja pior do que x^k .

Esse método utiliza estimativas quadráticas a partir dc informações dc determinados pontos c valores da função nesses pontos.

Os dois métodos acima usam a busca unidirccional, outros métodos que usam a busca unidirccional são método dc **Goldstein** c o dc Armijo esses porém fazem uso dc derivadas.

iii) Método de Cauchy

E um método derivado do método do gradiente, no qual o gradiente é aproximado através de diferenças finitas. O algoritmo efetua um cálculo aproximado do gradiente da função usando vetores canónicos Q € R" , **i** = 1, 2, 3, n. A direção dc busca é essa aproximação do gradiente com sinal trocado. Então ***i\x)*** deve ser, pelo menos, continuamente difcrcnciável.

iv) Método dc Powcll

Atinge o mínimo dc uma função quadrática com Ilessiana positiva definida, em no máximo n iterações, através de sucessivas buscas unidirecionais ao longo de uma série dc direções conjugadas partindo de um ponto inicial x_0.

O método sc fundamenta no fato dc que o mínimo de uma função quadrática é encontrado ao iongo de cada p (p < n) direções conjugadas cm um estágio dc procura; é fácil ver então que sc é necessário apenas um passo cm cada direção as n direções serão alcançadas em n passos, ver |1 limmclblauj para maiores detalhes.

v) Método do poliedro flexível (Neldcr - Mead)

A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	18.43385	4.197552	7.957879	18.1325	14.48885	9.663457	5.805336	19.988	Resultado da simulação		
24	x2	6.057911	11.7886	15.8051	4.998361	12.03108	10.60922	5.926791		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1 ,x2)	-721.876	-36.0836	-131.136	-690.854	-55912.3				4.77054	6.30712	0.30493
26	(x1-x1fk)^3	2550.748	-0.18812	32.38058	2385.671	0.229693				a1k	a2k	fmax
27	(x2-x2fk)^3	-0.01548	164.6999	856.8296	-2.2417	-0.46208				0.229693	-0.46208	-0.98979
28	Fk(x1 ,x2)	0	0	0	0	2.078113				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					4.540847	6.769202	
30	Curare	1.008049	1.026687	1.02138	1.00831							
31	x1Fk-a1k-x1ik	-13.893	0.343295	-3.41703	-13.5917							
32	x2Fk-a2k-x2ik	0.711291	-5.0194	-9.0359	1.770841							
33	f(x1Fk-a1k,x2ik)	-0.84664	-34.3512	-96.9834	-1.84657							
34	f(x1ik,x2Fk-a2k)	-722.465	-3.16736	-35.5879	-690.442							
35	Sinal 1	1	1	1	1							
36	Sinal 2	-1	1	1	1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	18.38667	4.198739	7.946121	18.08633							
39	Cálculo 2	6.055495	11.77124	15.77401	5.004377							
40												
41	Iteração	2										
42	x1	18.38667	4.198739	7.946121	18.08633	14.45434	9.64619	5.78543	19.93564	Resultado da simulação		
43	x2	6.055495	11.77124	15.77401	5.004377	12.01348	10.59556	5.906935		x1Fk	x2Fk	f(x1Fk,x2F
44	f(x1,x2)	-716.814	-35.8753	-130.25	-685.999	-55525.1				4.69348	6.380561	0.520645^
45	(x1-x1fk)^3	2567.518	-0.1211	34.41189	2402.254	0.167917				a1k	a2k	fmax
46	(x2-x2fk)^3	-0.03435	156.6498	828.8494	-2.60633	0.381915				0.167917	0.381915	-0.98979
47	Fk(x1 ,x2)	0	0	0	0	1.73261				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					4.525563	5.998646	
49	Curare	1.00809	1.026684	1.021433	1.008352							
50	x1Fk-a1k-x1ik	-13.8611	0.326824	-3.42056	-13.5608							
51	x2Fk-a2k-x2ik	-0.05685	-5.77259	-9.77537	0.994269							
52	f(x1Fk-a1k,x2ik)	-0.90344	-34.2075	-96.4317	-1.89163							
53	f(x1ik,x2Fk-a2k)	-716.811	-2.56808	-34.7185	-685.008							
54	Sinal 1	1	1	1	1							
55	Sinal 2	1	1	1	1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	17.81317	4.212511	7.802724	17.52511							

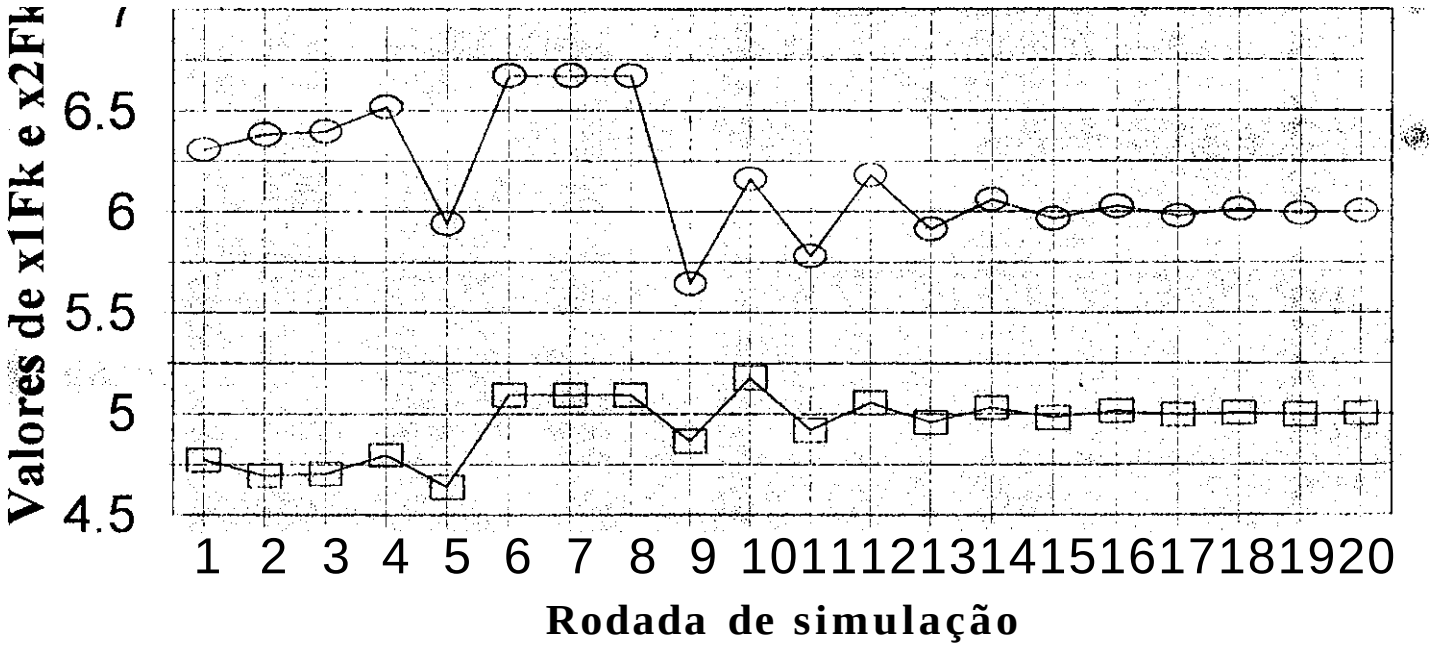
A:A22: **Iteração**
A:B22: 1
A:A23: **x1**
A:B23: **+$B$9+($B$10-$B$9)*@RAND**
A:C23: **+$B$9+($B$10-$B$9)*@RAND**
A:A24: **x2**
A:B24: **+$B$11+($B$12-$B$11)*@RAND**
A:C24: **+$B$11 +($B$12-$B$11)*@RAND**
A:A25: **f(x1 ,x2)**
A:B25: **-4*(B23-5)^2-(B24-6)^2**
A:C25: **-4*(C23-5)^2-(C24-6)^2**
A:A26: **'(x1-x1fk)^3**
A:B26: **(B23-$GY25)^3**
A:C26: **(C23-$GY25)^3**
A:A27: **'(x2-x2fk)^3**
A:B27: **(B24-$GZ25)^3**
A:C27: **(C24-$GZ25)^3**
A:A28: **Fk(x1 ,x2)**
A:B28: **@EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))**
A:C28: **@EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))**
A:A29: **Fk/SumFk**
A:B29: **+B28/($D$14+$GU28)**
A:C29: **+C28/($D$14+$GU28)**
A:A30: **Curare**
A:B30: **($HA27-B25)^(1 /($F$10+($HA27-B25)))**
A:C30: **($HA27-C25)^(1 /($F$10+($HA27-C25)))**
A:A31: **x1Fk-a1k-x1ik**
A:B31: **+$GY29-B23**
A:C31: **+$GY29-C23**
A:A32: **x2Fk-a2k-x2ik**
A:B32: **+$GZ29-B24**
A:C32: **+$GZ29-C24**
A:A33: **f(x1Fk-a1k,x2ik)**
A:B33: **-4*($GY29-5)^2-(B24-6)^2**
A:C33: **-4*($GY29-5)^2-(C24-6)^2**
A:A34: **f(x1 ik,x2Fk-a2k)**
A:B34: **-4*(B23-5)^2-($GZ29-6)^2**
A:C34: **-4*(C23-5)^2-($GZ29-6)^2**
AA35: **Sinal** 1
A:B35: **(B33-B25)/($D$14+@ABS(B33-B25))**
A:C35: **(C33-C25)/($D$14+@ABS(C33-C25))**
A:A36: **Sinal 2**
A:B36: **(B34-B25)/($D$14+@ABS(B34-B25))**
A:C36: **(C34-C25)/($D$14+@ABS(C34-C25))**
A:A37: **Passo**
A:B37: **+$D$10*@EXP(-($E$10/$B22))**
A:C37: **+$D$10*@EXP(-($E$10/$B22))**
A:A38: **Cálculo 1**
A:B38: **+B23+B37*B30*B35*B31**
A:C38: **+C23+C37*C30*C35*C31**
A:A39: **Cálculo 2**
A:B39: **+B24+B37*B30*B36*B32**
A:C39: **+C24+C37*C30*C36*C32**
A:GU23: **@SUMPRODUCT(B23..GT23,B25..GT25)/$GU25**
A:GV23: **@AVG(B23..GT23)**
A:GW23: **@STD(B23..GT23)**
A:GX23: **@MAX(B23..GT23)**

A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: +4*($GY25-5)^2+($GZ25-6)^2
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

Rodada -->											
fmax	-0.98979	-0.98979	-0.98979	-0.90735	-0.80162	-0.4887	-0.4887	-0.4887	-0.19881	-0.15278	-0.07405
Delta x1Fk	-0.07706	0.011748	0.090657	-0.15733	0.456327	-1.4E-07	1.4E-07	-0.23005	0.313038	-0.25832	0.074048
Delta x2Fk	0.073441	0.016325	0.121053	-0.57776	0.732647	3.8E-07	-1.02741	0.516509	-0.38138	0.215716	-0.01556
Delta f	0.215716	-0.01556	-0.07017	0.091225	-0.03744	-6.2E-07	6.2E-07	-0.2899	-0.04603	-C.07873	0.015716

Função de teste 10

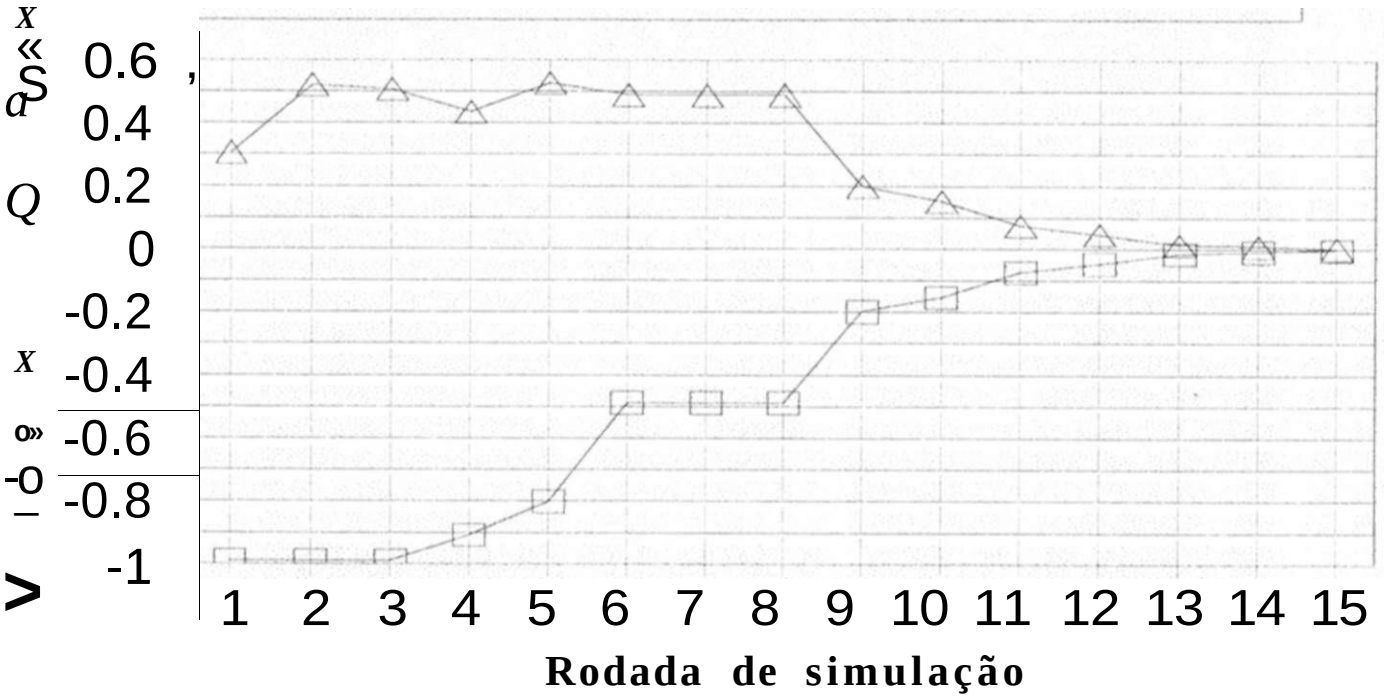
Comportamento de x1 Fk e x2Fk



i Valor de x1Fk Valor x2Fk

Função de teste 10

Comportamento de f(x1 F,x2F) e fmax

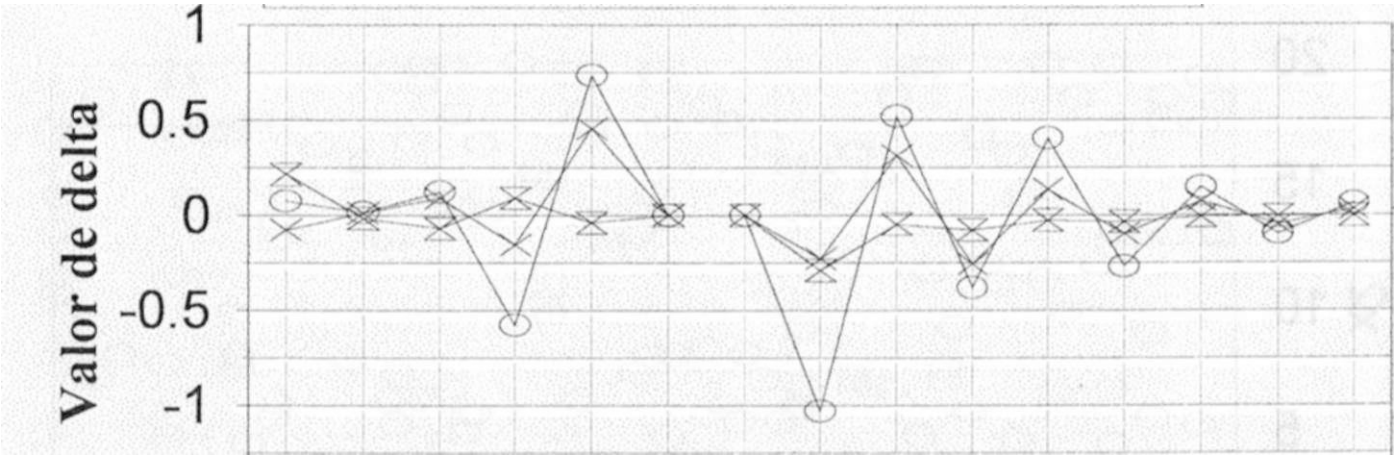


Valor de fmax

A Valor de f(x1Fk,x2Fk)

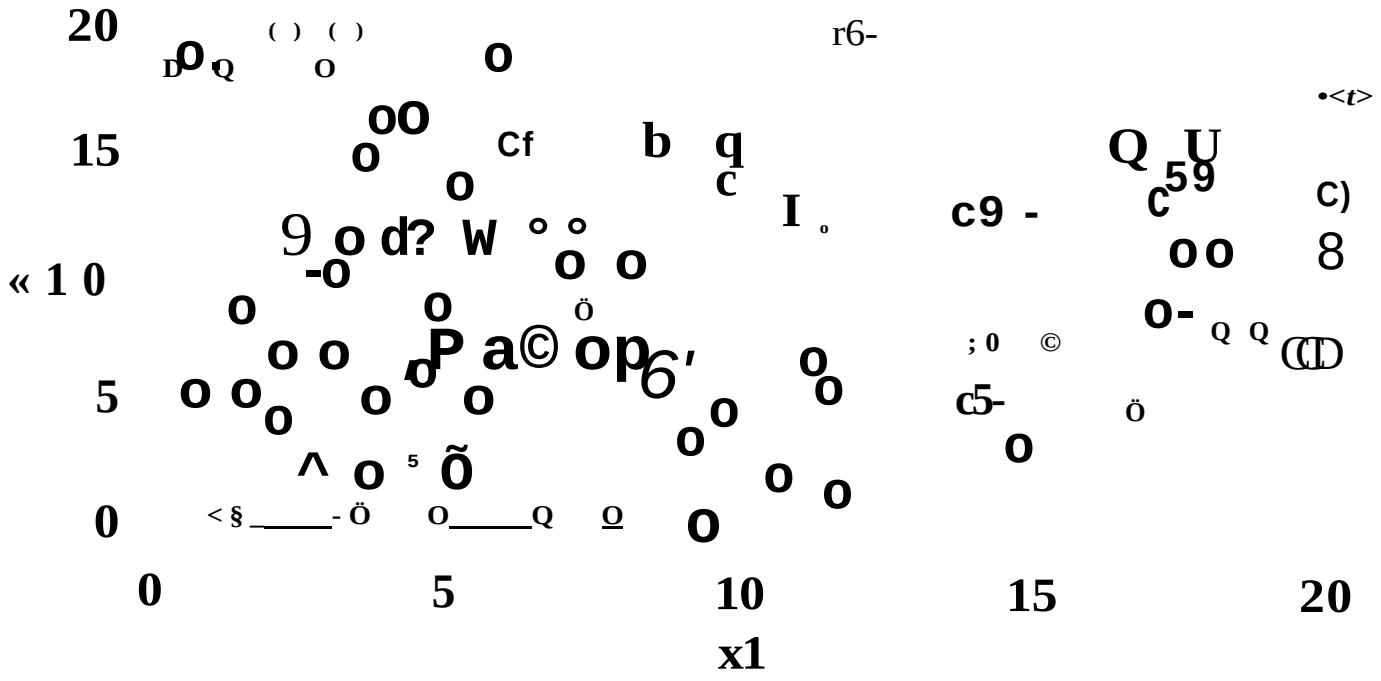
Função de teste 10

Comportamento dos Deltas



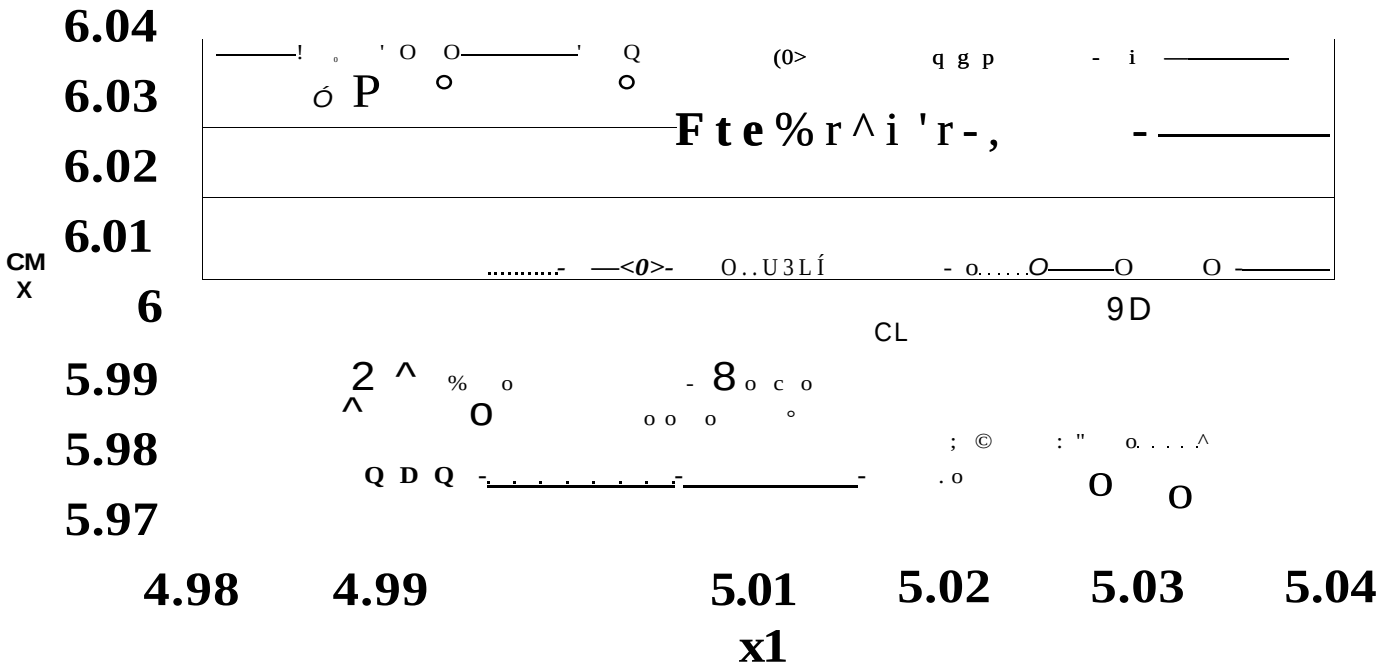
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação



Esse método apresenta uma das concepções mais criativas entre todos os que já descrevemos. A partir dos trabalhos dc Spcndlcy, I lext c I limsworth dc 1962, Ncldcr e Mead o formularam.

É um algoritmo que exige apenas a expressão analítica dc **f**ᵥx), c eficiente c dc làcil implementação computacional.

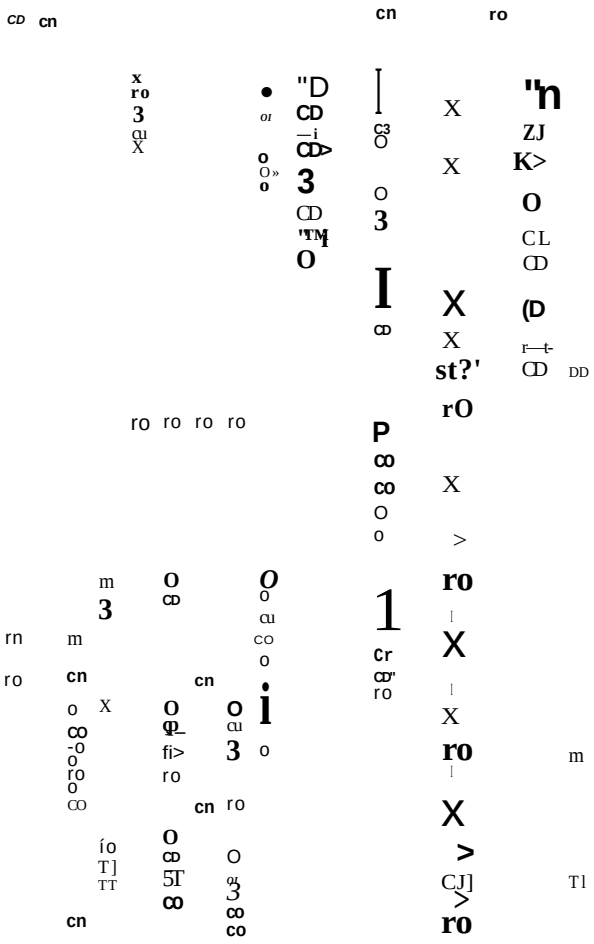
O algoritmo sc baseia no fato dc que um poliedro regular dc n I 1 vórtices cm R" c um simplex. Então dados n i 1 pontos formando um simplex, podemos avaliar f(x) cm cada um desses pontos c naquele vértice para o qual obtemos o maior valor dc f(x) é feita uma reflexão através do centroide do simplex. _sse vértice pode ser substituído pelo novo ponto e um novo simplex é obtido. Prosseguindo nesse processo c utilizando técnicas adequadas dc redução gradativa do simplex c de evitar ciclagcm na vizinhança do ponto de mínimo, obtém-se um método dc desempenho apenas razoável.

O trabalho dc Ncldcr - Mead foi exatamente dc introduzir alterações nesse método que o levasse a solucionar problemas práticos tais como. impossibilidade dc acelerar a busca do mínimo ou mesmo dc continua-la em certos casos. A alteração básica introduzida foi a possibilidade do simplex, durante a execução, sofrer variações cm sua forma deixando dc ser um simplex regular.

O algoritmo dc Ncldcr - Mead minimiza uma função f: R^n - > R usando n + 1 vértices dc um poliedro flexível, cm R^n. O vértice que maximiza \\\) é projetado através do centróide dos vértices restantes (c não do poliedro como antes), originando um novo ponto cm que f(x) pode ou não ter um valor menor. Isso acarreta movimentos dc reflexão, expansão, contração c redução para o poliedro e daí esses movimentos sugerirem o nome para o algoritmo.

2.3.1.2 Métodos com derivadas

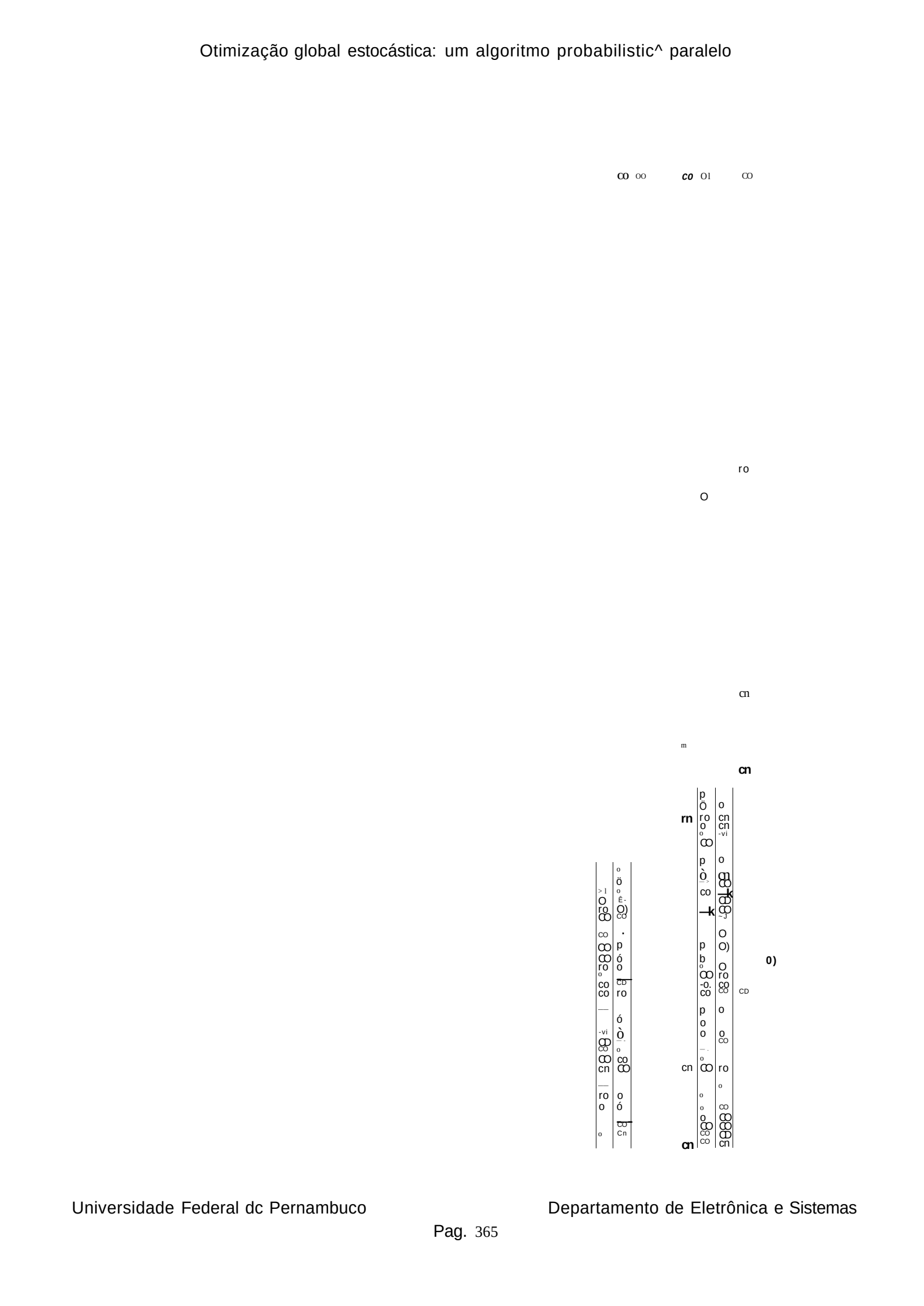
Anexo M



A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	-0.07137	-1.17394	-1.16842	-0.04658	-1.83719	0.054483	1.076888	1.995988	Resultado da simula*		j
24	x2	0.257625	0.580856	-1.13288	1.771127	0.842417	-0.06693	1.133451		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1,x2)	-0.00019	-1488.91	-26281.7	-0.01033	-3.7E+07				0.683919	-0.04993	26.89847
26	(x1-x1fk)^3	-0.43087	-6.41267	-6.35566	-0.38981	-0.32693				a1k	a2k	fmax
27	(x2-x2fk)^3	0.029091	0.250983	-1.27006	6.039065	-0.38914				-0.32693	-0.38914	-1.1E-09
28	Fk(x1 ,x2)	1	0	0	0.99971	107.6623				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0.009288	0	0	0.009286					1.010845	0.339216	
30	Curare	0.918093	1.004609	1.000386	0.955308							
31	x1Fk-a1k-x1ik	1.082219	2.184786	2.179265	1.057423							
32	x2Fk-a2k-x2ik	0.08159	-0.24164	1.472094	-1.43191							
33	f(x1Fk-a1k,x2ik)	-9.4E-10	-4.8E-09	-1.8E-08	-4.4E-08							
34	f(x1 ik,x2Fk-a2k)	-0.00024	-153.186	-146.731	-0.00011							
35	Sinal 1	1	-1	1	1							
36	Sinal 2	-1	1	1	1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	-0.06803	-1.16655	-1.16107	-0.04317							
39	Cálculo 2	0.257373	0.580039	-1.12792	1.766519							
40												
41	Iteração	2										
42	x1	-0.06803	-1.16655	-1.16107	-0.04317	-1.82796	0.057685	1.07325	1.992597	Resultado da simulação		
43	x2	0.257373	0.580039	-1.12792	1.766519	0.844361	-0.06586	1.129764		x1Fk	x2Fk	f(x1Fk.x2F
44	f(x1,x2)	-0.00018	-1399.83	-24517.4	-0.00821	-3.5E+07				0.653485	-0.07693	25.5191
45	(x1-x1fk)^3	-0.3756	-6.02888	-5.97467	-0.33811	-0.32022				a1k	a2k	fmax
46	(x2-x2fk)^3	0.037362	0.283555	-1.16089	6.264619	-0.41226				-0.32022	-0.41226	-1.1E-09
47	Fk(x1,x2)	1	0	0	0.999502	101.8329				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0.00982	0	0	0.009815					0.97371	0.335329	
49	Curare	0.917204	1.004842	1.000411	0.953118							i
50	x1Fk-a1k-x1ik	1.041736	2.140256	2.134785	1.016884							
51	x2Fk-a2k-x2ik	0.077956	-0.24471	1.463246	-1.43119							
52	f(x1Fk-a1k,x2ik)	-3E-08	-1.5E-07	-5.8E-07	-1.4E-06							
53	f(x1 ik,x2Fk-a2k)	-0.00022	-137.589	-131.807	-9E-05							
54	Sinal 1	1	1	1								
55	Sinal 2	-1	1	1	1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	-0.02881	-1.07828	-1.07342	-0.0034							
58	Cálculo 2	0.254438	0.569947	-1.06784	1.710533							

A:A22: Iteração
A:B22: 1
A:A23: x1
A:B23: +$B$9+($B$10-$B$9)*@RAND
A:C23: +$B$9+($B$10-$B$9)*@RAND
A:A24: x2
A:B24: +$B$11+($B$12-$B$11)*@RAND
A:C24: +$B$11+($B$12-$B$11)*@RAND
A:A25: f(x1,x2)
A:B25: -(B23*B24)^2*(1-B23)^2*(1-B23-B24*(1-B23)^5)^2
A:C25: -(C23*C24)^2*(1-C23)^2*(1-C23-C24*(1-C23)^5)^2
A:A26: '(x1-x1fk)^3
A:B26: (B23-$GY25)^3
A:C26: (C23-$GY25)^3
A:A27: '(x2-x2fk)^3
A:B27: (B24-$GZ25)^3
A:C27: (C24-$GZ25)^3
A:A28: Fk(x1,x2)
A:B28: @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28: @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29: Fk/SumFk
A:B29: +B28/($D$14+$GU28)
A:C29: +C28/($D$14+$GU28)
A:A30: Curare
A:B30: ($HA27-B25)^(1 /($F$10+($HA27-B25)))
A:C30: ($HA27-C25)^(1/($F$10+($HA27-C25)))
A:A31: x1Fk-a1k-x1ik
A:B31: +$GY29-B23
A:C31: +$GY29-C23
A:A32: x2Fk-a2k-x2ik
A:B32: +$GZ29-B24
A:C32: +$GZ29-C24
A:A33: f(x1Fk-a1k,x2ik)
A:B33: -($GY29*B24)^2*(1-$GY29)^2*(1-$GY29-B24*(1-$GY29)^5)^2
A:C33: -($GY29*C24)^2*(1-$GY29)^2*(1-$GY29-C24*(1-$GY29)^5)^2
A:A34: f(x1ik,x2Fk-a2k)
A:B34: -(B23*$GZ29)^2*(1-B23)^2*(1-B23-$GZ29*(1-B23)^5)^2
A:C34: -(C23*$GZ29)^2*(1-C23)^2*(1-C23-$GZ29*(1-C23)^5)^2
A:A35: Sinal 1
A:B35: (B33-B25)/($D$14+@ABS(B33-B25))
A:C35: (C33-C25)/($D$14+@ABS(C33-C25))
A:A36: Sinal 2
A:B36: (B34-B25)/($D$14+@ABS(B34-B25))
A:C36: (C34-C25)/($D$14+@ABS(C34-C25))
A:A37: Passo
A:B37: +$D$10*@EXP(-($E$10/$B22))
A:C37: +$D$10*@EXP(-($E$10/$B22))
A:A38: Cálculo 1
A:B38: +B23+B37*B30*B35*B31
A:C38: +C23+C37*C30*C35*C31
A:A39: Cálculo 2
A:B39: +B24+B37*B30*B36*B32
A:C39: +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23.B25..GT25)/$GU25
A:GV23: @AVG(B23..GT23)
A:GW23: @STD(B23..GT23)
A:GX23: @MAX(B23..GT23)

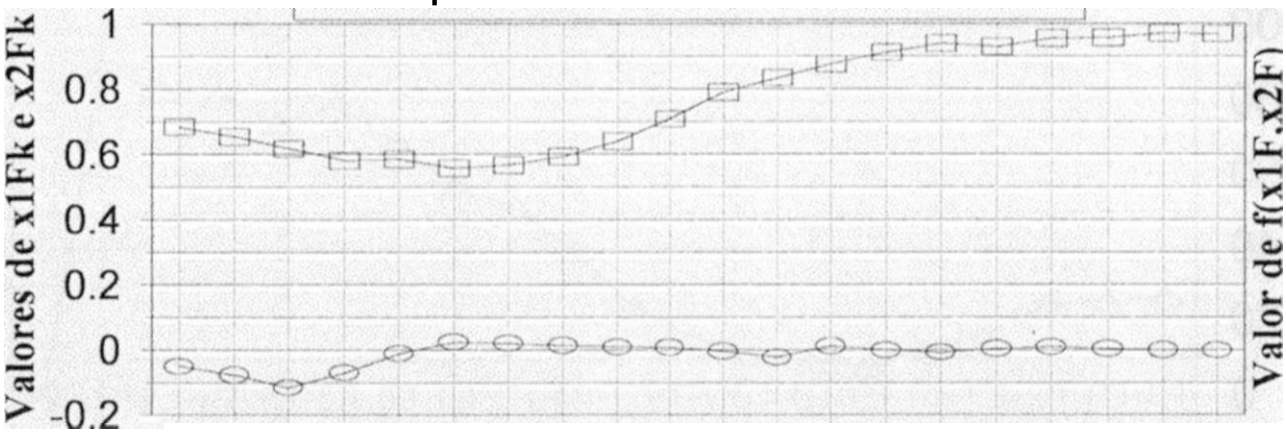
```
A:GY23:  Resultado da simulação
A:GU24:  @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24:  @AVG(B24..GT24)
A:GW24:  @STD(B24..GT24)
A:GY24:  x1Fk
A:GZ24:  x2Fk
A:HA24:  f(x1Fk,x2Fk)
A:GU25:  @SUM(B25..GT25)
A:GY25:  @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25:  @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25:  +  100*(GZ25-GY25^2)^2+(1-GY25)^2
A:GU26:  (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26,
         B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26:  a1k
AGZ26:   a2k
A:HA26:  fmax
A:GU27:  (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27,
         B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27:  +GU26
A:GZ27:  +GU27
A:HA27:  @MAX(B25..GT25)
A:GU28:  @SUM(B28..GT28)
A:GY28:  x1Fk-a1k
A:GZ28:  x2Fk-a2k
A:GY29:  +GY25-GY27
A:GZ29:  +GZ25-GZ27
```



V	X	N	AA	AB	AC	AD
0.833418	0.875284	0.911557	0.93914	0.92973	0.954936	0.956799	0.971321	0.970203
-0.02242	0.011864	-0.00085	-0.00726	0.003619	0.008964	0.003772	0.000302	-0.00153
-1.9E-16	-1.9E-16	-1.9E-16	[illegible]	-1.4E-16	[illegible]	-9E-20	-9E-20	-9E-20
51.43706	56.90604	69.19503	8,07933	74.09878	81.53209	83.12016	88.95644	1.7E-12
0.041865	0.036274	0.027583	[illegible] 00941	0.025207	0.001863	0.014522	-0.00112	-0.9702
0.03428	[illegible]	-0.00641	[illegible].01088	0.005344	-0.00519	-0.00347	-0.00183	0.001526
5.468977	12.289	9.884296	[illegible].98055	7.433307	1.588072	5.83628	-88.9564	-1.7E-12

Função de teste 11

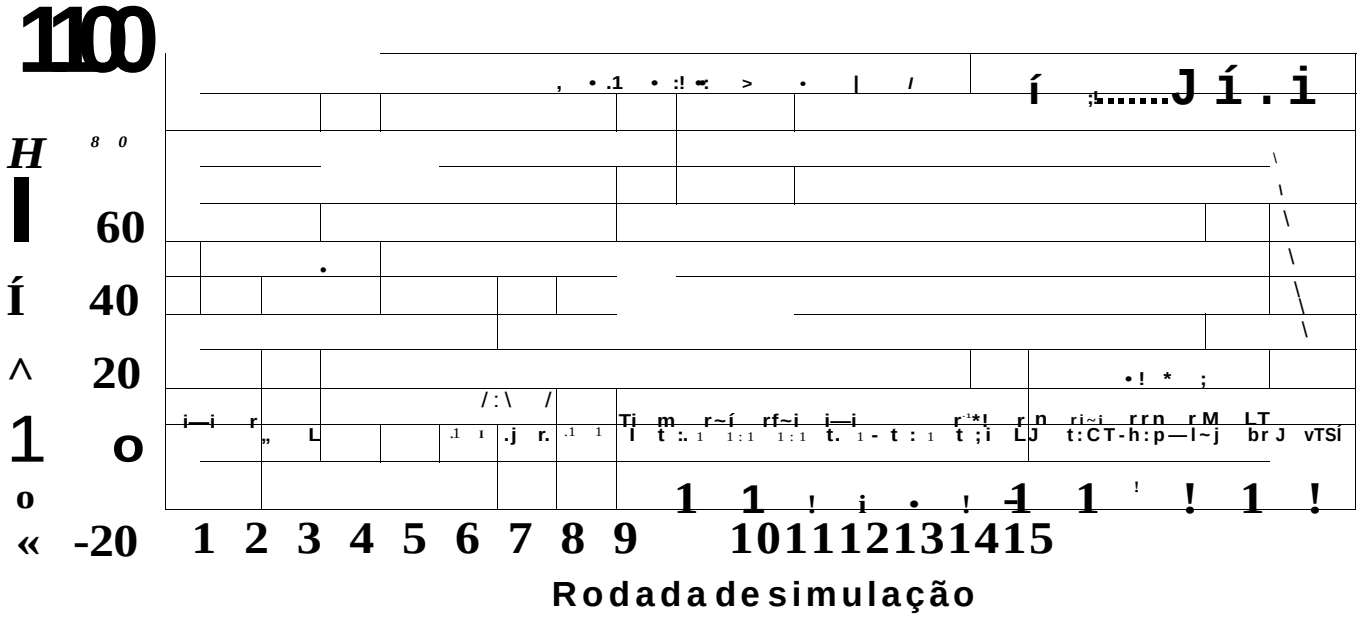
Comportamento de x1 Fk e x2Fk



i i Valor de x1Fk Valor x2Fk

Função de teste 11

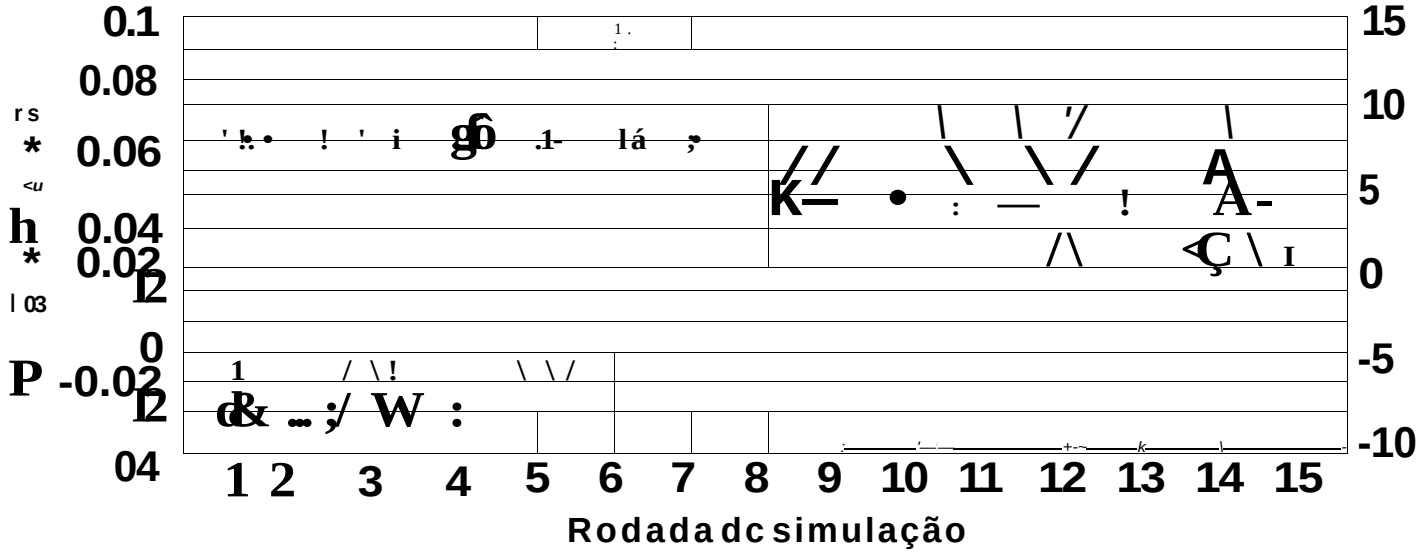
Comportamento de f(x1F,x2F) e fmax



i Valor de fmax **A Valor de f(x1Fk,x2Fk)**

Função de teste 11

Comportamento dos Deltas



M Delta x1 Fk ' Delta x2Fk >«• Delta f(x1 F,x2F)

Veremos agora alguns métodos que utilizam a derivada para a solução do problema de programação não linear sem vínculos. Aqui, de acordo com **[Ribeiro],** um algoritmo será considerado convergente se:

i) a sequencia gerada (X,) c finita c para seu último ponto x* é satisfeita a condição Vf(x*) 0, ou

ii) a seqüência (Xj) é infinita c qualquer um de seus pontos dc acumulação x* satisfeita a condição Vf(x*) - 0.

Alem disso algumas informações adicionais podem ser obtidas a partir dc certas condições :

I) sc f(x) c convexa c Vf(x*) - 0, então x* c ponto dc mínimo global.

2) se xo é ponto inicial e o conjunto V={x e R" : *í\x) < í{xo)}* for limitado então sempre haverá pontos dc acumulação pois as buscas unidirecionais somente fornecem pontos em V c portanto, as sequencias geradas são compactas.

Segundo |IIimmelblau| na prática os métodos que utilizam derivadas têm dois inconvenientes para implementação:

I) Problemas com médio número de variáveis torna-os muito trabalhosos ou as vezes impossível prover funções analíticas para as derivadas cm algoritmos de gradiente ou dc derivadas segunda;

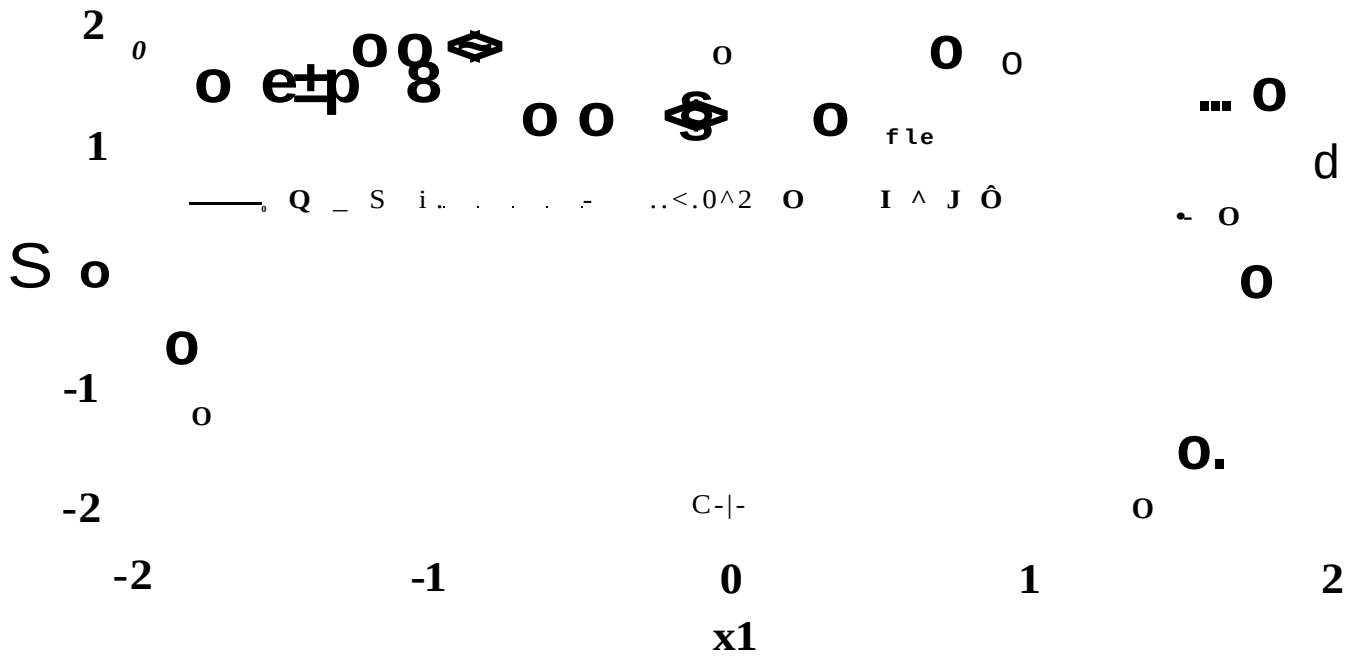
2) **Técnicas** dc otimização baseadas no uso de derivadas de primeira e segunda ordens requer uma quantidade relativamente grande de problemas dc preparação pelo usuário antes dc colocar o problema no algoritmo.

i) Método do gradiente

E um dos mais antigos processos dc otimização c também um dos mais simples. Se baseia no faio de que o gradiente calculado cm qualquer ponto do domínio de *í\)* aponta para a direção dc máximo crescimento inicial da função. Como estamos a procura do mínimo deveremos andar cm sentido contrário a direção do gradiente naquele ponto, esse algoritmo utiliza a derivada primeira da função objetivo f(-).

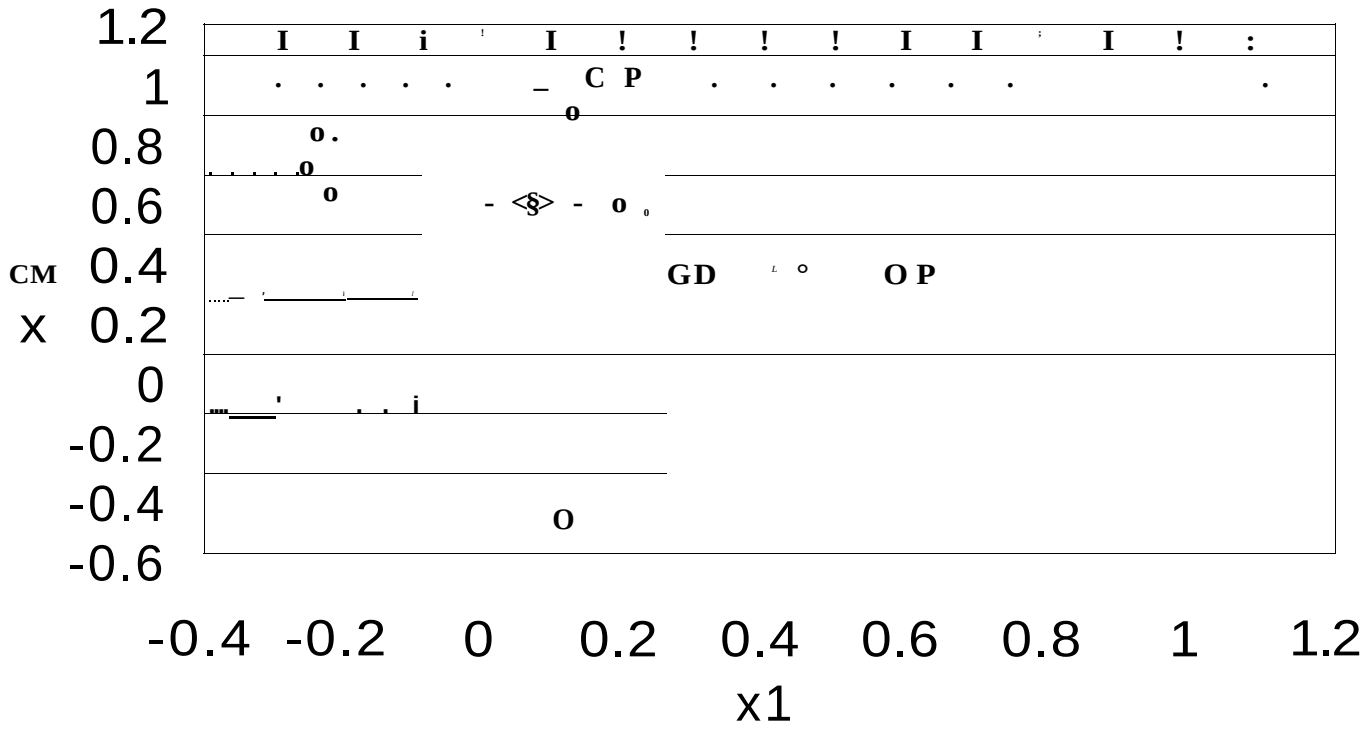
Nuvem de pontos

Primeira rodada de simulação



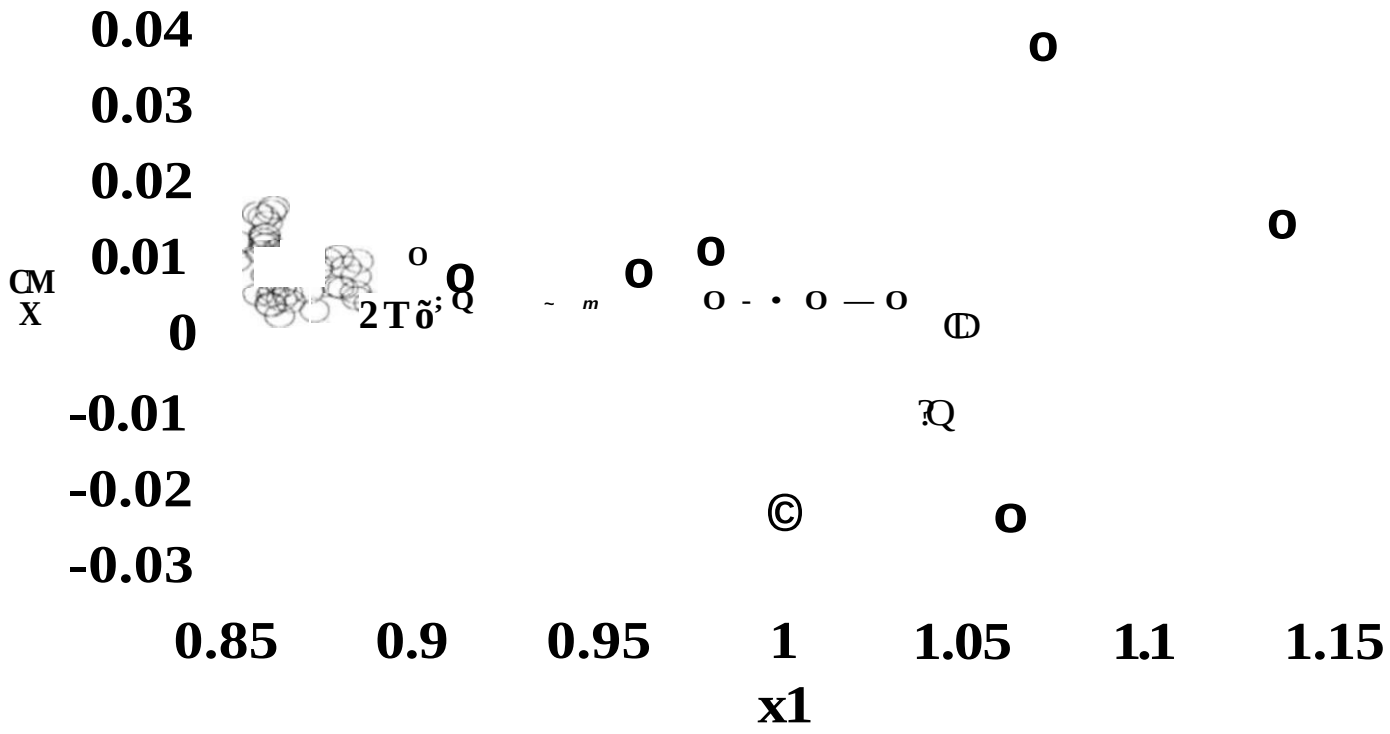
Nuvem de pontos

Décima rodada de simulação

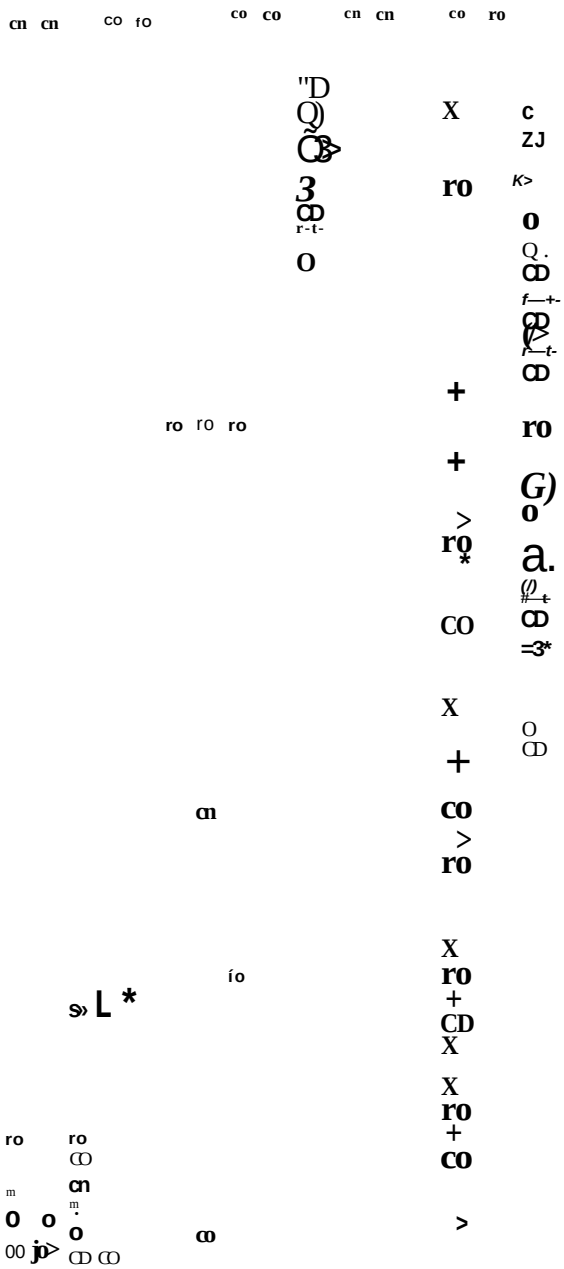


Nuvem de pontos

Vigésima rodada de simulação



Anexo N



A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	-1.70861	-1.90768	0.544585	-1.75841	-0.53987	0.086869	1.195949	1.969017	Resultado da simulação		
24	x2	-0.43745	1.7975	-1.92941	-1.71773	0.971389	0.020156	1.156725		x1Fk	x2Fk	f(x1Fk,x2f;
25	f(x1 ,x2)	-26560.9	-615881	-17999.7	-18889.8	-8889934				0.120516	-0.94824	-6.45286
26	(x1-x1fk)^3	-6.11969	-8.34313	0.076262	-6.63326	2.4E-49				a1k	a2k	fmax
27	(x2-x2fk)^3	0.133273	20.70041	-0.94457	-0.45563	-1E-45				2.4E-49	-1E-45	-6.45286
28	Fk(x1 ,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					0.120516	-0.94824	
30	Curare	1.000382	1.000022	1.000542	1.000519							
31	x1Fk-a1k-x1ik	1.829124	2.028195	-0.42407	1.878923							
32	x2Fk-a2k-x2ik	-0.5108	-2.74574	0.981171	0.769489							
33	f(x1Fk-a1k,x2ik)	-373.496	-135156	-41595.1	-10764.3							
34	f(x1ik,x2Fk-a2k)	-8370.3	-18533.2	-526.543	-10059.5							
35	Sinal 1	1	1	-1	1							
36	Sinal 2	1	1	1	1							
37	Passo	0.006738	0.006738	0.006738	0.006738							
38	Cálculo 1	-1.69628	-1.89401	0.547444	-1.74574							
39	Cálculo 2	-0.44089	1.778999	-1.9228	-1.71254							
40												
41	Iteração	2										
42	x1	-1.69628	-1.89401	0.547444	-1.74574	-0.52899	0.090075	1.193518	1.981632	Resultado da simulação		
43	x2	-0.44089	1.778999	-1.9228	-1.71254	0.955445	0.018186	1.152335		x1Fk	x2Fk	f(x1Fk,x2F
44	f(x1,x2)	-24928.9	-592217	-16979.3	-18402.8	-8573112				0.120516	-0.94824	-6.45286
45	(x1-x1fk)^3	-5.99677	-8.17561	0.077815	-6.50001	2.4E-49				a1k	a2k	fmax
46	(x2-x2fk)^3	0.130596	20.28478	-0.92559	-0.44647	-1E-45				2.4E-49!	-1E-45	-6.45286!
47	Fk(x1,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					0.120516	-0.94824	
49	Curare	1.000405	1.000022	1.000571	1.000531							
50	x1Fk-a1k-x1ik	1.816795	2.014528	-0.42693	1.866256							
51	x2Fk-a2k-x2ik	-0.50735	-2.72724	0.974556	0.764302							
52	f(x1Fk-a1k,x2ik)	-368.969	-130632	-40039.1	-10372.3							
53	f(x1ik,x2Fk-a2k)	-8014.99	-17482.2	-537.64	-9588.14							
54	Sinal 1	1	1	-1	1							
55	Sinal 2	1	1	1	1							
56	Passo	0.082085	0.082085	0.082085	0.082085							
57	Cálculo 1	-1.54709	-1.72865	0.582508	-1.59247							
58	Cálculo 2	-0.48255	1.555128	-1.84276	-1.64977							

```
A:A22:    Iteração
A:B22:    1
A:A23:    x1
A:B23:     +$B$9+($B$10-$B$9)*@RAND
A:C23:     +$B$9+($B$10-$B$9)*@RAND
A:A24:    x2
A:B24:     +$B$11+($B$12-$B$11)*@RAND
A:C24:    +$B$11 +($B$12-$B$11 )*@RAND
A:A25:    f(x1,x2)
A:B25:       -(1+(B23+B24+1)^2*(19-14*B23+3*B23^2-14*B24+6*B23*B24+3*B24^2))*(30+(2*B23-3*B24)^2*(
          18-32*B23+12*B23^2+48*B24-36*B23*B24+27*B24^2))
A.C25:     -(1+(C23+C24+1)^2*(19-14*C23+3*C23^2-14*C24+6*C23*C24+3*C24^2))*(30+(2*C23-3*C24)^2*
          (18-32*C23+12*C23^2+48*C24-36*C23*C24+27*C24^2))
A:A26:    '(x1-x1fk)^3
A:B26:    (B23-$GY25)^3
A:C26:    (C23-$GY25)^3
A:A27:    '(x2-x2fk)^3
A:B27:    (B24-$GZ25)^3
A:C27:    (C24-$GZ25)^3
A:A28:    Fk(x1,x2)
A:B28:   @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28:     @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29:    Fk/SumFk
A:B29:    +B28/($D$14+$GU28)
A:C29:    +C28/($D$14+$GU28)
A:A30:    Curare
A:B30:    ($HA27-B25)^(1/($F$10+($HA27-B25)))
A:C30:    ($HA27-C25)^(1/($F$10+($HA27-C25)))
A:A31:    x1Fk-a1k-x1ik
A:B31:    +$GY29-B23
A:C31:    +$GY29-C23
A:A32:    x2Fk-a2k-x2ik
A:B32:    +$GZ29-B24
A:C32:    +$GZ29-C24
A:A33:    f(x1Fk-a1k,x2ik)
A:B33:     -(1+($GY29+B24+1)^2*(19-14*$GY29+3*$GY29^2-14*B24+6*$GY29*B24+3*B24^2))*(30+(2*$G
          Y29-3*B24)^2*(18-32*$GY29+12*$GY29^2+48*B24-36*$GY29*B24+27*B24^2))
A:C33:    -(1+($GY29+C24+1)^2*(19-14*$GY29+3*$GY29^2-14*C24+6*$GY29*C24+3*C24^2))*(30+(2*$G
          Y29-3*C24)^2*(18-32*$G Y29+12*$G Y29^2+48*C24-36*$G Y29*C24+27*C24^2))
A:A34:    f(x1ik,x2Fk-a2k)
A:B34:     -(1+(B23+$GZ29+1)^2*(19-14*B23+3*B23^2-14*$GZ29+6*B23*$GZ29+3*$GZ29^2))*(30+(2*B23
          -3*$GZ29) ^2*(18-32*B23+12*B23^2+48*$GZ29-36*B23*$GZ29+27*$GZ29^2))
A:C34:    -(1+(C23+$GZ29+1)^2*(19-14*C23+3*C23^2-14*$GZ29+6*C23*$GZ29+3*$GZ29^2))*(30+(2*C2
          3-3*$GZ29)^2*(18-32*C23+12*C23^2+48*$GZ29-36*C23*$GZ29+27*$GZ29^2))
A:A35:    Sinal 1
A:B35:    (B33-B25)/($D$14+@ABS(B33-B25))
A:C35:     (C33-C25)/($D$14+@ABS(C33-C25))
A:A36:    Sinal 2
A:B36:    (B34-B25)/($D$14+@ABS(B34-B25))
A:C36:   (C34-C25)/($D$14+@ABS(C34-C25))
A:A37:    Passo
A:B37:   +$D$10*@EXP(-($E$10/$B22))
A:C37:   +$D$10*@EXP(-($E$10/$B22))
A:A38:    Cálculo 1
A:B38:    +B23+B37*B30*B35*B31
A:C38:   +C23+C37*C30*C35*C31
A:A39:    Cálculo 2
```

A:B39: +B24+B37*B30*B36*B32
A:C39: +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23: @AVG(B23..GT23)
A:GW23: @STD(B23..GT23)
A:GX23: @MAX(B23..GT23)
A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A.GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: -(1+($GY25+$GZ25+1)^2*(19-14*$GY25+3*$GY25^2-14*$GZ25+6*$GY25*$GZ25+3*$GZ25^2))*(30+(2*$GY25-3*$GZ25)^2*(18-32*$GY25+12*$GY25^2+48*$GZ25-36*$GY25*$GZ25+27*$GZ25^2))
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A.GZ29: +GZ25-GZ27


KJ
CD
oo
o
CA)
o
Cn
cn
o
30
CO
CD
cn
CO

[illegible]	[illegible]		X	[illegible]	N	AA	AB	AC	AD	[illegible]	AF	AG
	[illegible]	[illegible]		[illegible]	[illegible]	-4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	0.000442	0.001664	0.001136	0.000712	0.000439	0.00051	0.000351	0.00037	0.00028	0.000195	0.000143	8.7E-05
[illegible]	-0.99868	-0.99898	-0.99903	-0.99914	-0.999361	-0.99958	-0.99986	-0.99982	-0.99991	-0.9999	-0.99994	-0.99996
[illegible]	-3.00051	[illegible]	-3.00034	-3.00023	-3.00007	-3.00007	-3.00002	-3.000011	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	-3.00067	-3.00078	-3.00049	-3.00031	-3.00016	-3.0001	-3.00003	-3.00003	-3.00002	-3.00001	[illegible]	[illegible]
[illegible]	0.001223		-0.00042	-0.00027	7.1E-05	-0.00016	[illegible]	[illegible]		-5.2E-05	-5.6E-05	-1.9E-05
[illegible]	-0.0003	[illegible]	-0.00011	-0.00022	-0.00022	-0.00028	[illegible]	[illegible]	1.3E-05	-4.7E-05	-2.1E-05	-5.7E-06
	-0.0001	0.000287	0.000178	0.00015	6.9E-05	6.7E-05	-5.1E-06	1.6E-05	[illegible]	5.1E-06	3E-06	[illegible]

A direção Si (i indicando a i-esima iteração) c determinada por Sj - Vf(X,) c o novo ponto c obtido através da relação

Xj, i X, i A/S, X, - Wf(Xj), onde ***X'c*** o passo c é obtido pela busca direciona!.

ii) Método ***steepest descenl***

Como já dito anteriormente a direção contrária a do gradiente é dc descida ***(steepest descenl).*** O gradiente unitário (normalizado) dc isto é, a direção é definida em x^k por

s^k ----, nos fornece a transição dc x^k para $x^{k'1}$ dada por

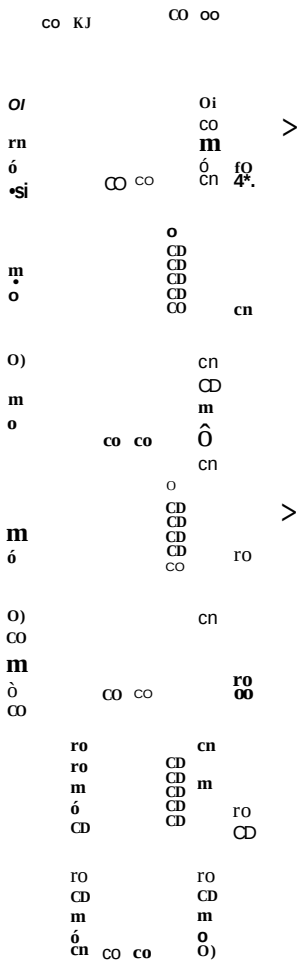
$$\frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}$$

$$x^{k+1} = x^k - \alpha^k \frac{\nabla f(x^k)}{\|\nabla f(x^k)\|}$$

onde α^k é o tamanho ótimo do passo (para demonstrar sua utilização [Himmeiblau] implementa este algoritmo para resolver a função dc Koscnbrock partindo dc [-0.5 ; 0.51', evidentemente o algoritmo não alcança o ponto dc ótimo),

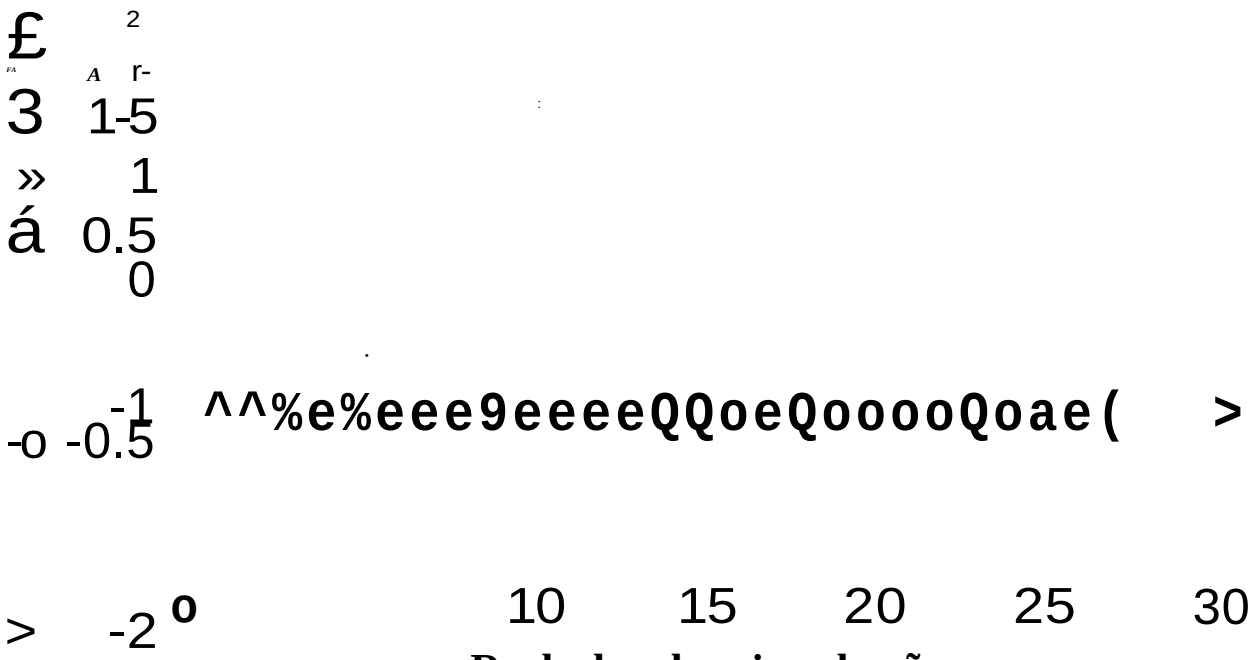
iii) Método dc Newton

A direção dc procura no método ***steepest descenl*** pode ser interpretada como produzindo uma aproximação linear da função objetivo.



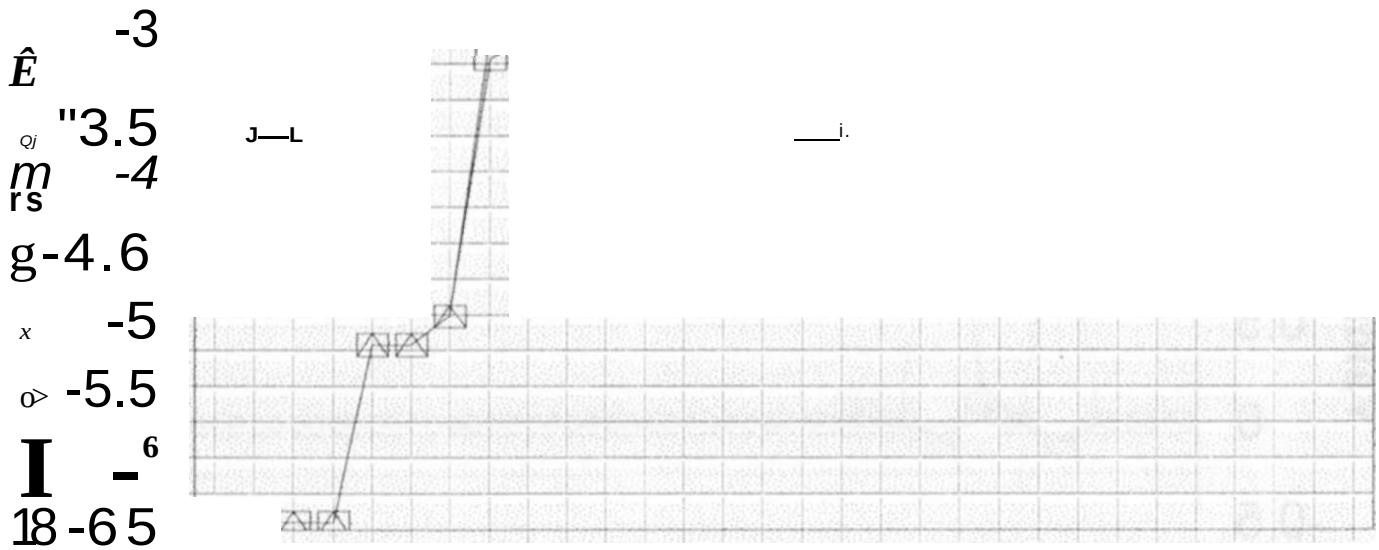
Função de teste 12

Comportamento de x1Fk e x2Fk



Função de teste 12

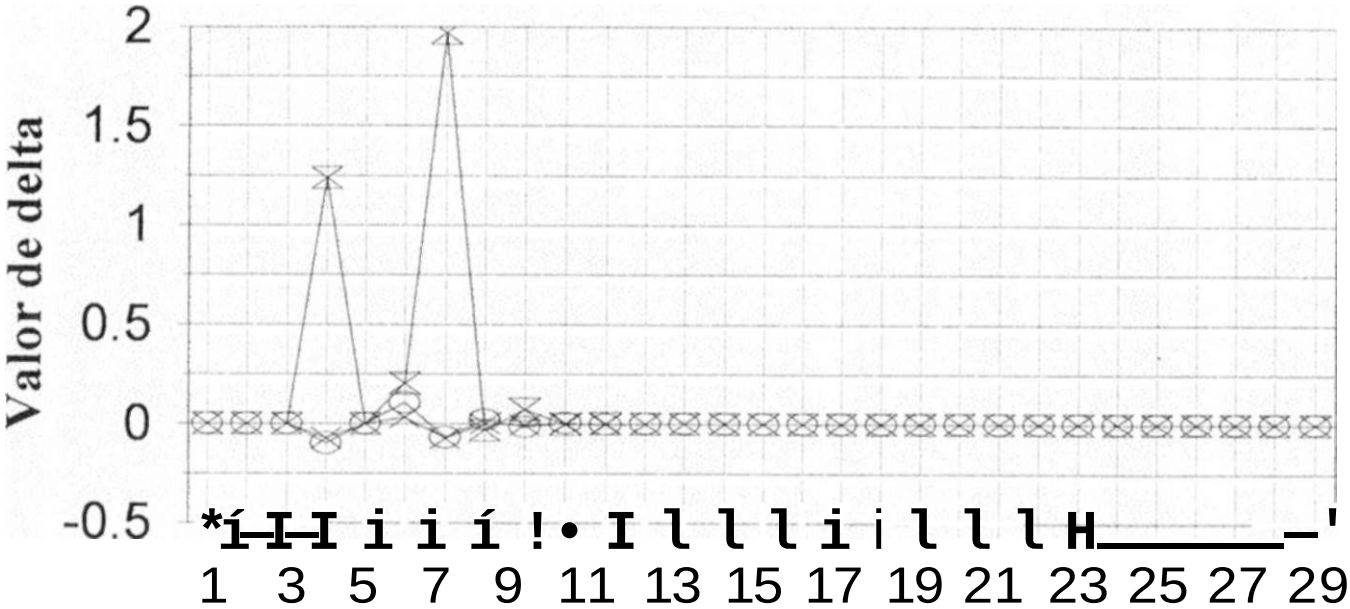
Comportamento de f(x1F,x2F) e fmax



-B Valor de fmax Valor de f(x1Fk,x2Fk)

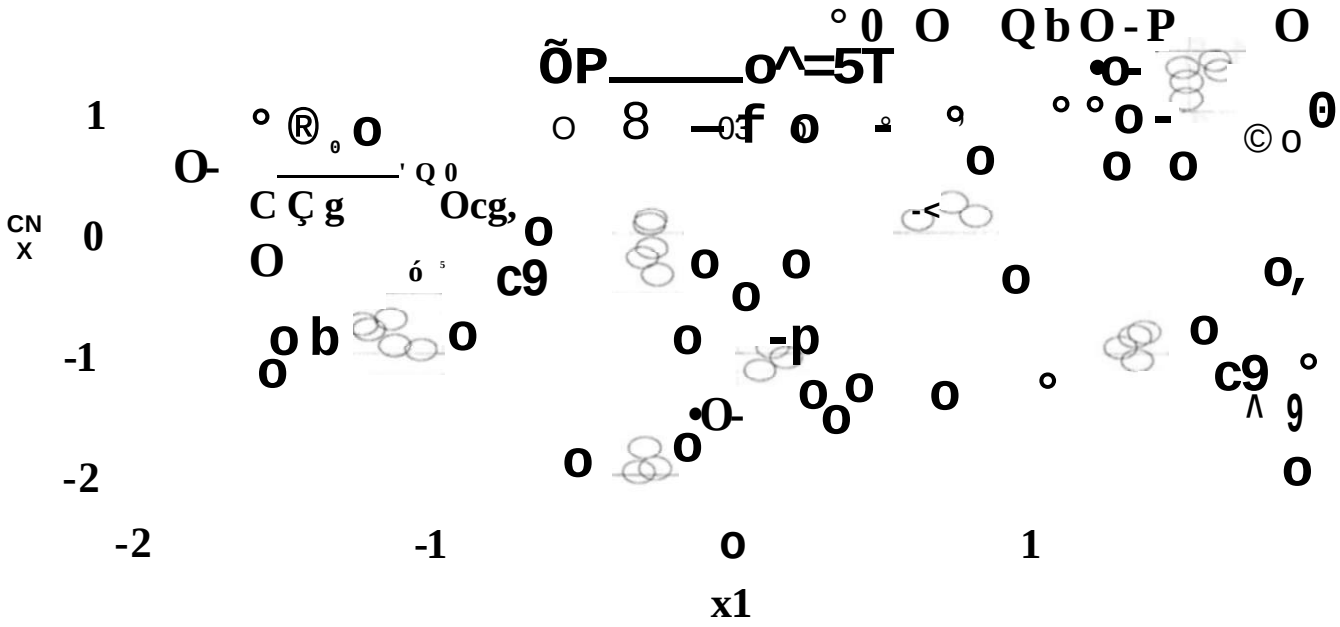
Função de teste 12

Comportamento dos Deltas



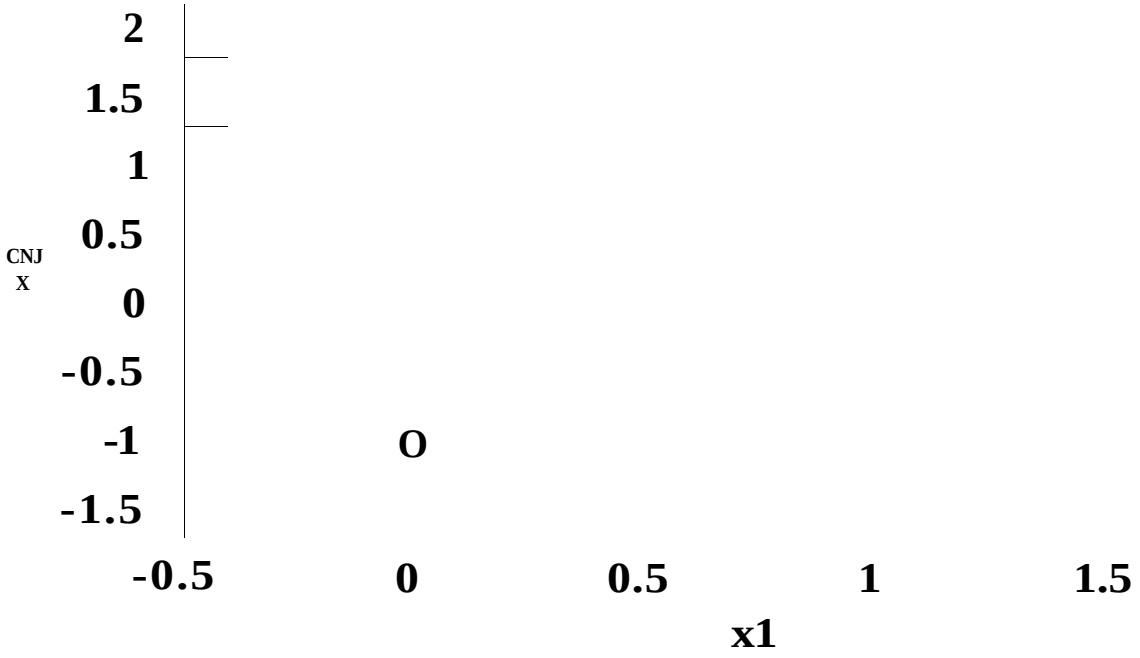
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Trigésima rodada de simulação



Anexo O

Função de teste 13

Proposto em Himrrx problem 28 pag. 42J

Parâmetros

Da função

x 3	o
xlmax	o
3	o
3 x	o

Do algoritmo

Gama 1	Gama 2
o	Oi
Delta 1	Delta 2
Erro	x
1E-15	3.014116
0.006646	

Gama 3

Delta 3

1.98666

A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										1 i
23	x1	1.040716	2.189734	8.472925	4.988509	6.806125	5.178703	2.844604	9.989737	Resultado da simulação		
24	x2	6.273924	5.368415	1.150026	1.872097	6.859425	5.328033	2.792747		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1 ,x2)	-1129.02	-577.162	-3844.44	-250.523	-762939				2.69025	2.304119	3.125466!
26	(x1-x1fk)^3	-4.48832	-0.12539	193.3688	12.1394	0.119756				a1k	a2k	fmax
27	(x2-x2fk)^3	62.56158	28.77347	-1.53717	-0.08063	-0.23818				0.119756	-0.23818	-3.31424
28	Fk(x1,x2)	0	0	0	0	1.014164				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					2.570494	2.542295	
30	Curare	1.005749	1.009472	1.002096	1.015997							
31	x1Fk-a1k-x1ik	1.529779	0.380761	-5.90243	-2.41801							
32	x2Fk-a2k-x2ik	-3.73163	-2.82612	1.392269	0.670198							
33	f(x1Fk-a1k,x2ik)	-1223.83	-595.843	-20.1671	-7.20791							
34	f(x1 ik,x2Fk-a2k)	-54.639	-16.1483	-4074.02	-289.682							
35	Sinal 1	-1	-1	1	1							
36	Sinal 2	1	1	-1	-1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	1.035532	2.188439	8.452998	4.980233							
39	Cálculo 2	6.26128	5.358804	1.145325	1.869803							
40												
41	Iteração	2										
42	x1	1.035532	2.188439	8.452998	4.980233	6.790349	5.168529	2.837103	9.964714	Resultado da simulação		
43	x2	6.26128	5.358804	1.145325	1.869803	6.844473	5.317274	2.785387		x1Fk	x2Fk	f(x1Fk,x2F
44	f(x1 ,x2)	-1118.28	-572.185	-3802.02	-247.808	-755210				2.683208	2.318124	3.31418
45	(x1-x1fk)^3	-4.47317	-0.12112	192.0791	12.11984	0.00804				a1k	a2k	fmax
46	(x2-x2fk)^3	61.3101	28.11331	-1.61313	-0.09011	-0.01599				0.00804	-0.01599Í	-3.31424
47	Fk(x1 ,x2)	0	0	0	0	1.000004				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					2.675168	2.334115	
49	Curare	1.005792	1.009529	1.002116	1.016091							
50	x1Fk-a1k-x1ik	1.639636	0.486729	-5.77783	-2.30506							
51	x2Fk-a2k-x2ik	-3.92717	-3.02469	1.188789	0.464312							
52	f(x1Fk-a1k,x2ik)	-1222.38	-597.263	-16.3586	-4.58209							
53	f(x1ik,x2Fk-a2k)	-57.9288	-15.4334	-3989.87	-272.151							
54	Sinal 1	-1	-1	1	1							
55	Sinal 2	1	1	-1	-1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	0.967848	2.168272	8.21536	4.884105							1
58	Cálculo 2	6.099166	5.23348	^.096431	1.850439							

```
A:A22:    Iteração
A:B22:    1
A:A23:    x1
A:B23:    +$B$9+($B$10-$B$9)*@RAND
A:C23:    +$B$9+($B$10-$B$9)*@RAND
A:A24:    x2
A:B24:    +$B$11 +($B$12-$B$11 )*@RAND
A:C24:    +$B$11+($B$12-$B$11)*@RAND
A:A25:    f(x1 ,x2)
A:B25:    -(B23^2+B24-11 )^2-(B23+B24^2-7)^2
A:C25:    -(C23^2+C24-11 )^2-(C23+C24^2-7)^2
A:A26:    '(x1-x1fk)^3
A:B26:    (B23-$GY25)^3
A:C26:    (C23-$GY25)^3
A:A27:    '(x2-x2fk)^3
A:B27:    (B24-$GZ25)^3
A:C27:    (C24-$GZ25)^3
A:A28:    Fk(x1,x2)
A:B28:    @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28:    @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29:    Fk/SumFk
A:B29:    +B28/($D$14+$GU28)
A:C29:    +C28/($D$14+$GU28)
A:A30:    Curare
A:B30:    ($HA27-B25)^(1/($F$10+($HA27-B25)))
A:C30:    ($HA27-C25)^(1/($F$10+($HA27-C25)))
A:A31:    x1Fk-a1k-x1ik
A:B31:    +$GY29-B23
A:C31:    +$GY29-C23
A:A32:    x2Fk-a2k-x2ik
A:B32:    +$GZ29-B24
A:C32:    +$GZ29-C24
A:A33:    f(x1Fk-a1k,x2ik)
A:B33:    -($GY29^2+B24-11)^2-($GY29+B24^2-7)^2
A:C33:    -($GY29^2+C24-11 )^2-($GY29+C24^2-7)^2
A:A34:    f(x1ik,x2Fk-a2k)
A:B34:    -(B23^2+$GZ29-11 )^2-(B23+$GZ29^2-7)^2
A:C34:    -(C23^2+$GZ29-11 )^2-(C23+$GZ29^2-7)^2
A:A35:    Sinal 1
A:B35:    (B33-B25)/($D$14+@ABS(B33-B25))
A:C35:    (C33-C25)/($D$14+@ABS(C33-C25))
A:A36:    Sinal 2
A:B36:    (B34-B25)/($D$14+@ABS(B34-B25))
A:C36:    (C34-C25)/($D$14+@ABS(C34-C25))
A:A37:    Passo
A:B37:    +$D$10*@EXP(-($E$10/$B22))
A:C37:    +$D$10*@EXP(-($E$10/$B22))
AA38:     Cálculo 1
A:B38:    +B23+B37*B30*B35*B31
A:C38:    +C23+C37*C30*C35*C31
A:A39:    Cálculo 2
A:B39:    +B24+B37*B30*B36*B32
A:C39:    +C24+C37*C30*C36*C32
A:GU23:   @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23:   @AVG(B23..GT23)
A:GW23:   @STD(B23..GT23)
A:GX23:   @MAX(B23..GT23)
```

O método de Newton (assim denominado porque a solução do conjunto de equações Vf\x) 0 pelo método de Newton produz a equação 2.3.1.2.3 abaixo) utiliza a derivada segunda realizando uma aproximação quadrática de í^x) utilizando a expansão em série de Taylor da equação 2.3.1.2.1 abaixo

l-q. 2.3.1.2.1 f(x) - fi>^{k}) » $V^{T}f(x^{k})$ (x - x^{k}) f *Vi* (x - x^{k}*jV*^{2}\\x^{k}) (x - x^{k})

onde $V^{2}f(x^{k})$ é a matriz Hessiana de f(x) cm x^{k}

A direção de procura S do método de Newton é escolhida substituindo (x - x^{k}) da eq. 2.3.1.2.1 acima por Ax^{k} x^{k+1} - x^{k} e a aproximação quadrática de f(x) em termos de Ax^{k} é

Eq. 2.3.1.2.2 $f(x^{k+1})$ - n>^{k}) + $V^{T}f(x^{k})$ Ax^{k} + '/$_{2}$ [$Ax^{T}V^{2}$**fi>^{k})** Ax^{k}

O mínimo de *i\x)* na direção de Ax é obtido diferenciando f(x) em relação a cada componente dc Ax e igualando o resultado a zero

Eq. 2.3.1.2.3 Ax^{k} -[V^x^{1})]"^{1} V^x*)

onde I $V^{2}f(x^{k})$| representa a inversa da matriz 1 lessiana em x^{k}. Substituindo a equação 2.3.1.2.3 na equação 2.3.1.2.2 obtemos.

Eq. 2.3.1.2.4 x^{k}" x^{k} - [V^{2}^)]"' Vn>^{k})

se f(x) for quadrática o mínimo será alcançado cm um único passo. Se f^x) não for quadrática (o que acontece na maioria dos casos) o mínimo isso não ocorrerá c a equação 2.3.1.2.4 acima será modificada com o intuito de introduzir o parâmetro para o tamanho do passo *X* ficando como abaixo

Eq. 2.3.1.2.5 x^{k+1} =x^{k} -X^{k} $\dfrac{\text{v/(,<)]}}{|\text{v}^{2}/(,')\,\Gamma}$

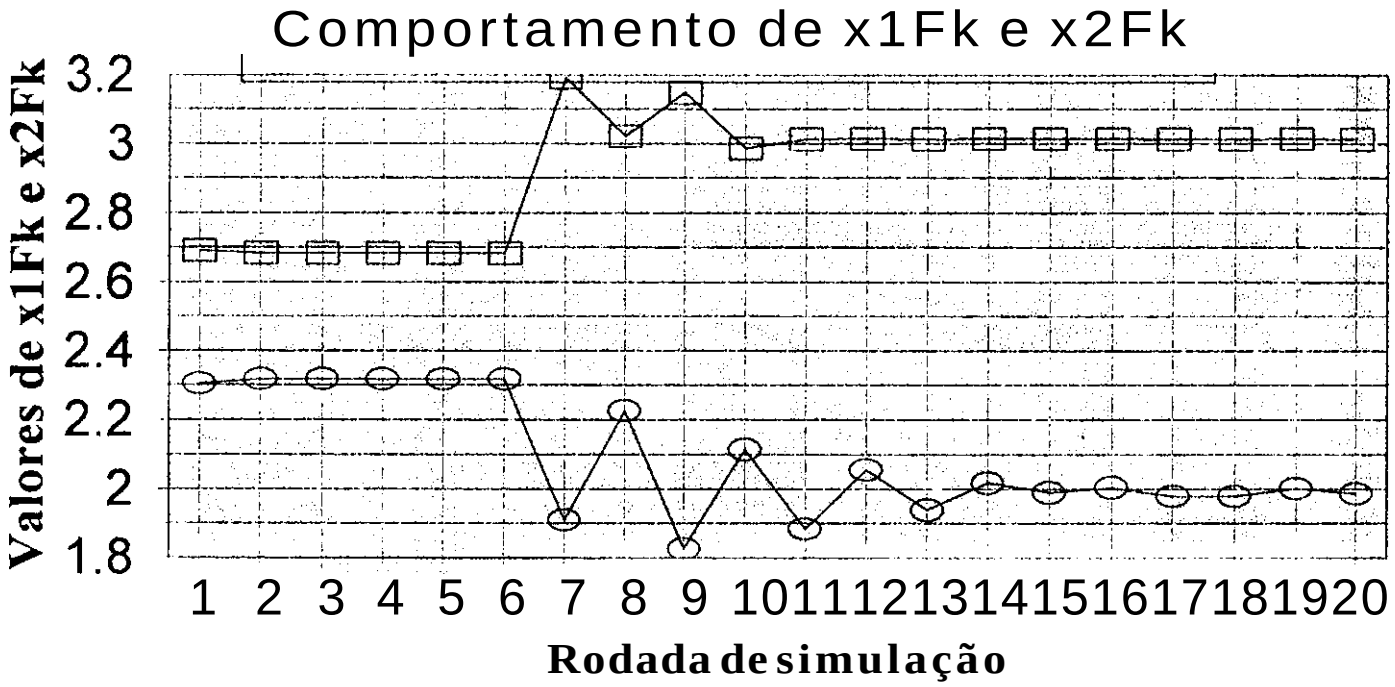
ou como é mais freqüentemente escrita

Eq. 2.3.1.2.6 * ^ " x^{k} - l^{H} *II* ^{1} [x*)vftx^{k})
\ *I* \ *I*

a equação 2.3.1.2.6 é aplicada seqüencialmente até atingirmos um critério de parada preestabelecido.

A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: (GY25^2+GZ25-11)^2+(GY25+GZ25^2-7)^2
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

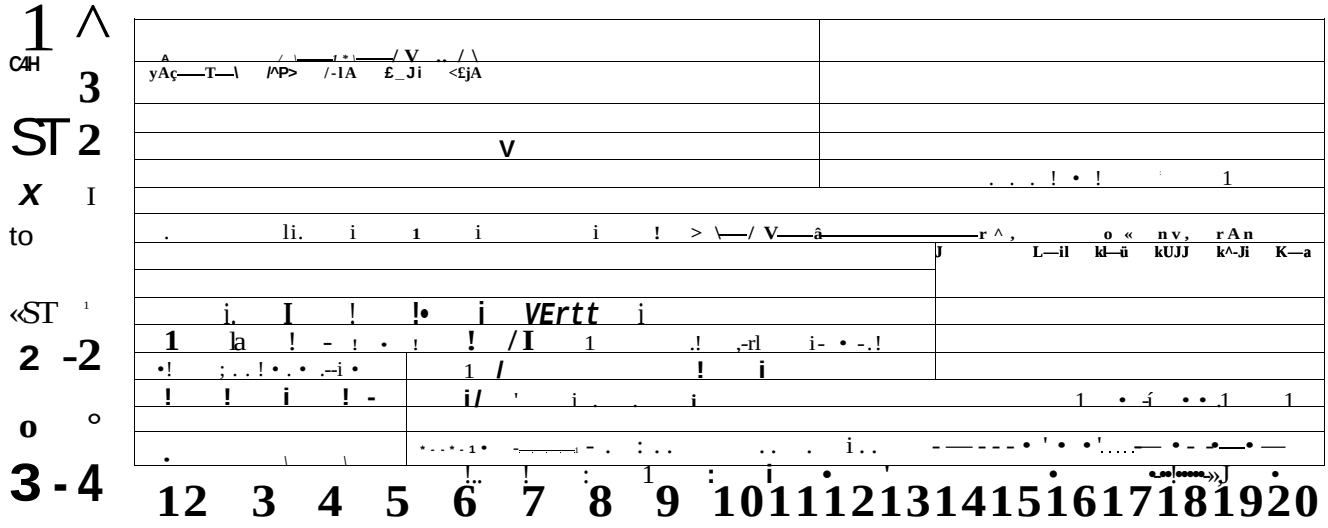
Função de teste 13



Valor de x1Fk **-e-** Valor x2Fk

Função de teste **13**

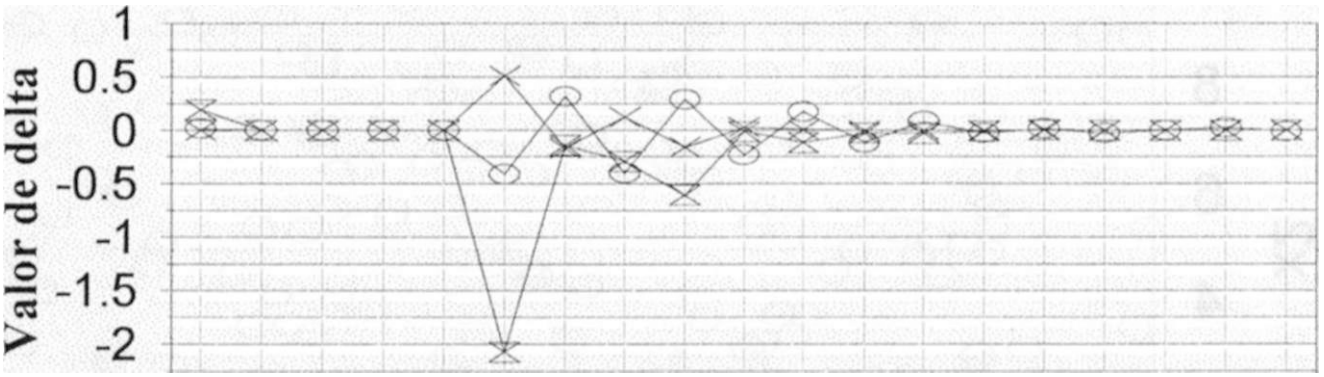
Comportamento de f(x1F,x2F) e fmax



-e- Valor de fmax **£r Valor de f(x1Fk,x2Fk)**

Função de teste 13

Comportamento dos Deltas

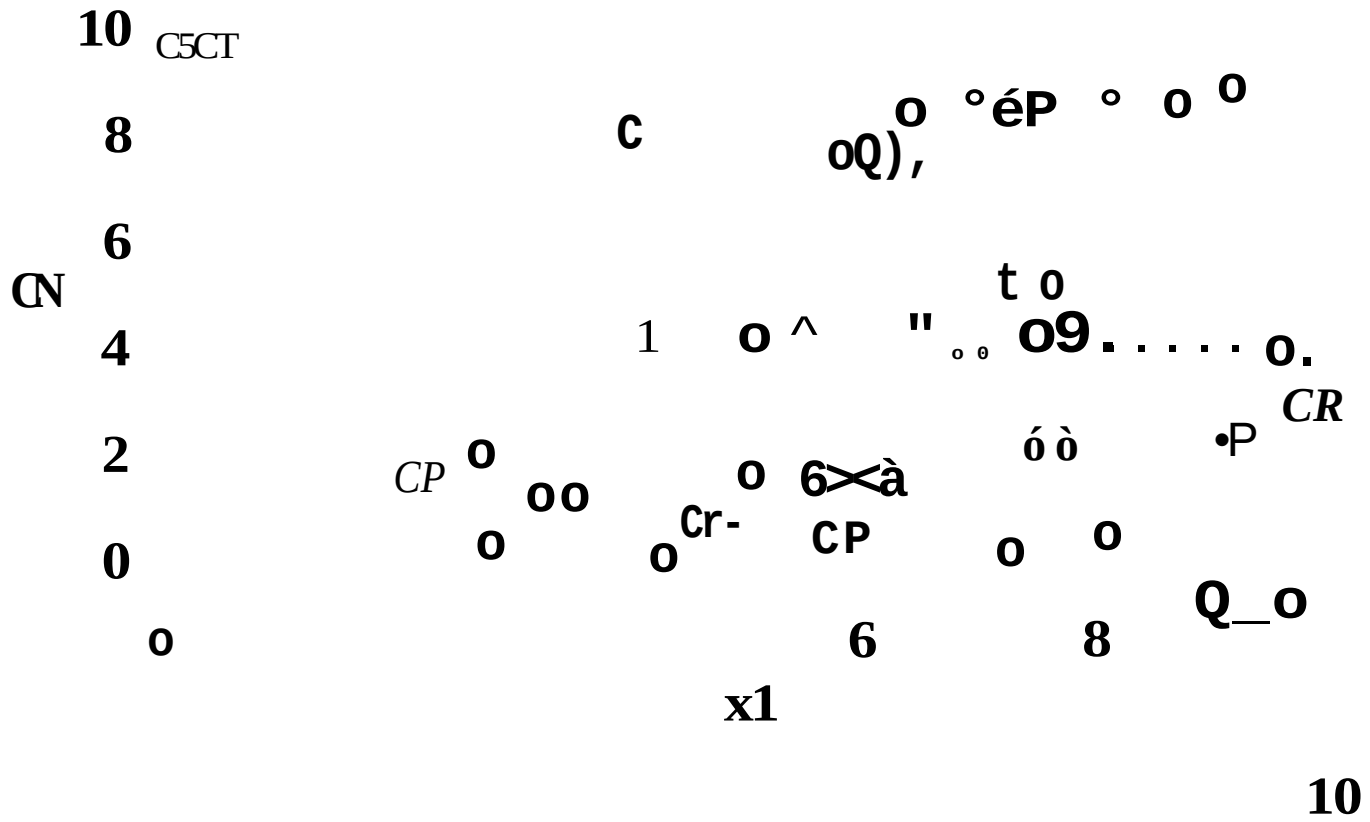


$$x \quad \text{Delta } x1\,Fk \quad - \quad e \quad - \text{Delta } x2Fk$$

$$\text{Delta } f(x1\,Fk, x2Fk)$$

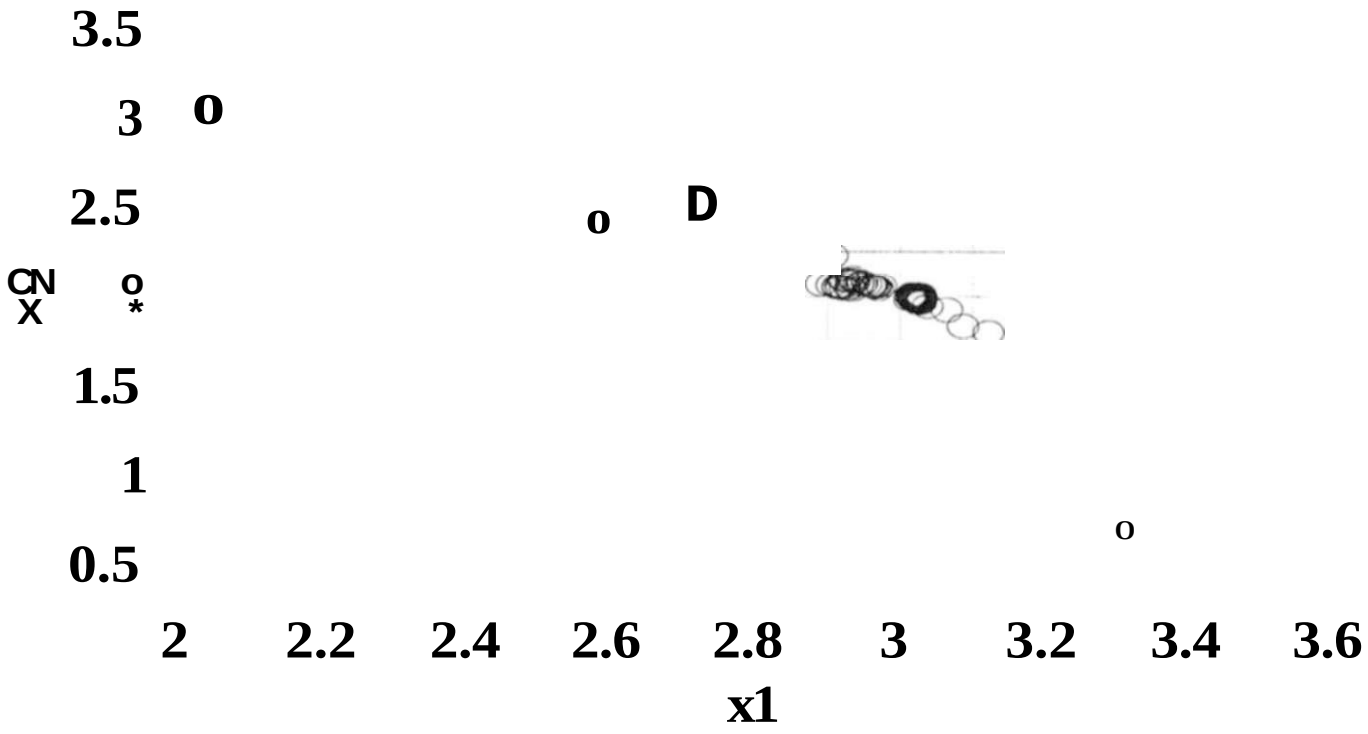
Nuvem de pontos

Primeira rodada de simulação

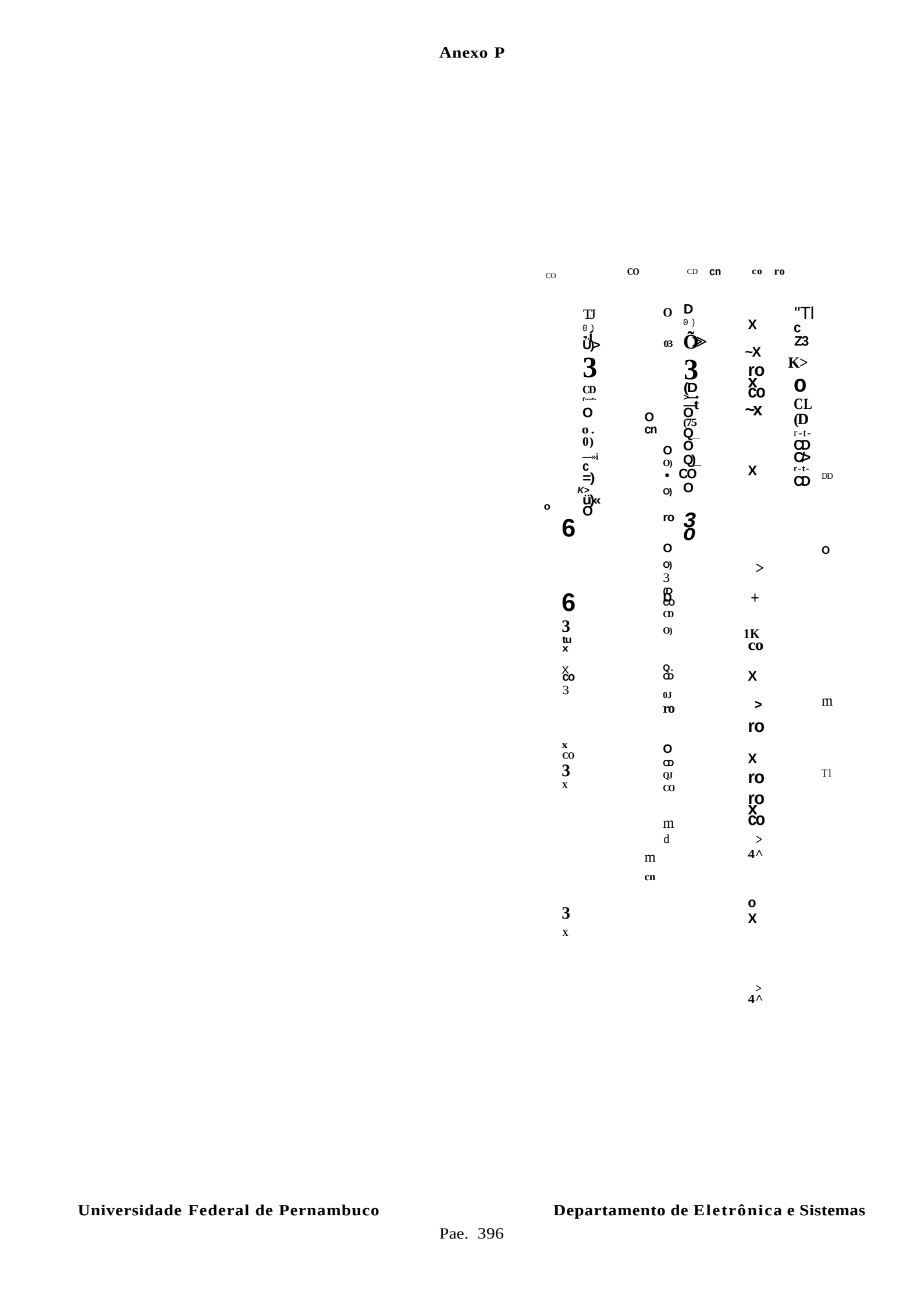


Nuvem de pontos

Vigésima rodada de simulação



Anexo P



iv) Método de Fletcher - Reeves (Gradientes conjugados)

Em termos dc convergência é inferior ao de Newton, todavia o fato dc não requerer o uso dc derivadas segunda, o que evita dc termos dc trabalhar com matriz Ilcssiana, que cm espaços dc dimensão elevada dificulta sobremaneira sua execução, faz sua eficiência ser equivalente aquele. Esse algoritmo possui a característica peculiar dc minimizar uma função quadrática em, no máximo n iterações.

O algoritmo c operacionalizado com a geração dc uma seqüência dc direções **Si** que são combinações lineares entre -Vf(Xj) e as direções anteriores de tal forma que, sc a função objetivo for quadrática, as direções geradas pelo algoritmo são conjugadas, ver |Ribeiro| c [Avricl|.

Seja **X(,** o ponto inicial c So < R^n a primeira direção dc busca dada por

So - $Vf(x_{()})$, define-se então as direções Si, i - 1 , 2, ... dc forma recursiva como

Spi - Vf(xj l i) i w, Si, onde Wj é dado por

$$\frac{(v/(*_{,,}),v/(*_{,,}))}{(V/(x,).V/(x,))}$$

Demonstra-se que esse método converge para funções estritamente convexas e duas vezes diferenciáveis.

iii) Método dc Davidon - Fletcher - Powell

Esse método aproxima a inversa da matriz Hessiana da função, diminuindo drasticamente , o volume dc cálculos envolvidos na obtenção da matriz I lessiana c posteriormente cm sua inversa.

Apresenta as vantagens dc ser altamente eficiente e de ter uma boa estabilidade computacional, em contrapartida possui a desvantagem dc utilizar muita memória pois as matrizes envolvidas, normalmente.

Cálculo 4	2.278283	6.605703	[illegible]	-7.814	
[illegible]					-406.591267965419[illegible]
[illegible]					406.591267965417

```
A:A15:    Iteração
A:B15:    1
A:A16:    x1
A:B16:     +$A$13+($B$13-$A$13)*@RAND
A:C16:     +$A$13+($B$13-$A$13)*@RAND
A:A17:    x2
A:B17:     +$C$13+($D$13-$C$13)*@RAND
A:C17:     +$C$13+($D$13-$C$13)*@RAND
A:A18:    x3
A:B18:    +$E$13+($F$13-$E$13)*@RAND
A:C18:    +$E$13+($F$13-$E$13)*@RAND
A:A19:    x4
A:B19:     +$G$13+($H$13-$G$13)*@RAND
A:C19:     +$G$13+($H$13-$G$13)*@RAND
A:A20:    f(x1 ,x2,x3,x4)
A:B20:      -(B16+10*B17)^2-5*(B18-B19)^2-(B17-2*B18)^4-10*(B16-B19)^4
A:C20:    -(C16+10*C17)^2-5*(C18-C19)^2-(C17-2*C18)^4-10*(C16-C19)^4
A:A21:    a1
A:B21:    (B16-$IC17)^3
A:C21:    (C16-$IC17)^3
A:A22:    a2
A:B22:    (B17-$ID17)^3
A:C22:    (C17-$ID17)^3
A:A23:    a3
A:B23:    (B18-$IE17)^3
A:C23:    (C18-$IE17)^3
A:A24:    a4
A:B24:    (B19-$IF17)^3
A:C24:    (C19-$IF17)^3
A:A25:    Fk
A:B25:   @EXP(-$D$8*@EXP($F$8*$B15)*(B20-$IG19)^(2*$E$8))
A:C25:     @EXP(-$D$8*@EXP($F$8*$B15)*(C20-$IG19)^(2*$E$8))
A:A26:    F/SumF
A:B26:    +B25/($G$8+$IA25)
A:C26:    +C25/($G$8+$IA25)
A:A27:    Curare
A:B27:    ($IG19-B20)^(1/(($IG19-B20)+$C$8))
A:C27:    ($IG19-C20)^(1/(($IG19-C20)+$C$8))
A:A28:    x1Fk-a1-x1
A:B28:    +$IC21-B16
A:C28:    +$IC21-C16
A:A29:    x2Fk-a2-x2
A:B29:    +$ID21-B17
A:C29:    +$ID21-C17
A:A30:    x3Fk-a3-x3
A:B30:    +$IE21-B18
A:C30:    +$IE21-C18
A:A31:    x4Fk-a4-x4
A:B31:    +$IF21-B19
A:C31:    +SIF21-C19
A:A32:    f(x1Fk-a1 ,x2,x3,x4)
A:B32:    -($IC21 + 10*B17)^2-5*(B18-B19)^2-(B17-2*B18)^4-10*($IC21-B19)^4
A:C32:    -($IC21+10*C17)^2-5*(C18-C19)^2-(C17-2*C18)^4-10*($IC21 -C19)^4
A:A33:    f(x1 ,x2Fk-a2,x3,x4)
A:B33:    -(B16+10*$ID21)^2-5*(B18-B19)^2-($IC21  -2*B18)^4-10*(B16-B19)^4
A:C33:    -(C16+10*$ID21 )^2-5*(C18-C19)^2-($IC21 -2*C18)^4-10*(C16-C19)^4
A:A34:    f(x1 ,x2,x3Fk-a3,x4)
```

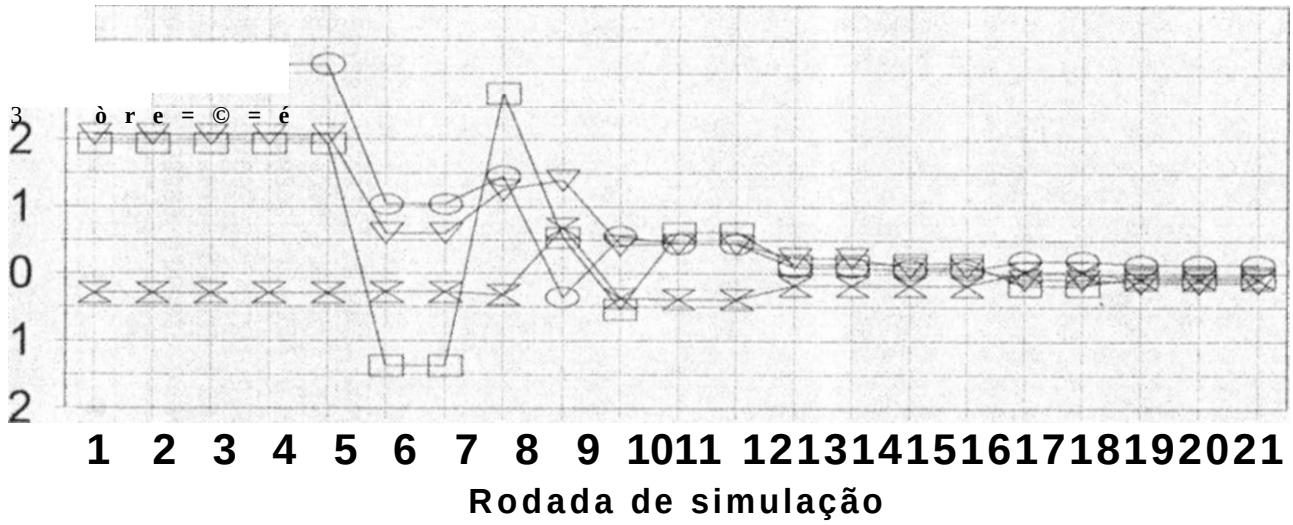
```
A:B34:       -(B16+10*B17)^2-5*($IE21-B19)^2-(B17-2*B18)^4-10*(B16-B19)^4
A:C34:      -(C16+10*C17)^2-5*($IE21 -C19)^2-(C17-2*C18)^4-10*(C16-C19)^4
A:A35:      f(x1 ,x2,x3,x4Fk-a4)
A:B35:       -(B16+10*B17)^2-5*(B18-$IF21)^2-(B17-2*B18)^4-10*(B16-$IF21)^4
A:C35:      -(C16+10*C17)^2-5*(C18-$IF21)^2-(C17-2*C18)^4-10*(C16-$IF21)^4
A:A36:      Sinal 1
A:B36:      (B32-B20)/($G$8+@ABS(B32-B20))
A:C36:     (C32-C20)/($G$8+@ABS(C32-C20))
A:A37:      Sinal 2
A:B37:      (B33-B20)/($G$8+@ABS(B33-B20))
A:C37:     (C33-C20)/($G$8+@ABS(C33-C20))
A:A38:      Sinal 3
A:B38:      (B34-B20)/($G$8+@ABS(B34-B20))
A:C38.     (C34-C20)/($G$8+@ABS(C34-C20))
A:A39:      Sinal 4
A:B39:      (B35-B20)/($G$8+@ABS(B35-B20))
A:C39:     (C35-C20)/($G$8+@ABS(C35-C20))
A:A40:      Passo
A:B40:      +$A$8*@EXP(-$B$8/$B15)
A:C40:      +$A$8*@EXP(-$B$8/$B15)
A:A41:      Cálculo 1
A:B41:       +B16+(B40*B27*B36*B28)
A:C41:      +C16+(C40*C27*C36*C28)
A:A42:      Cálculo 2
A:B42:       +B17+(B40*B27*B37*B29)
A:C42:      +C17+(C40*C27*C37*C29)
A:A43:      Cálculo 3
A:B43:       +B18+(B40*B27*B38*B30)
A:C43:      +C18+(C40*C27*C38*C30)
A:A44:      Cálculo 4
A:B44:       +B19+(B40*B27*B39*B31)
A:C44:      +C19+(C40*C27*C39*C31)
A:IC16:     x1Fk
A:ID16:    x2Fk
A:IE16:     x3Fk
A:IF16:     x4Fk
A:IG16:    f(x1Fk,x2Fk,x3Fk,x4Fk)
A:IC17:      @SUMPRODUCT(B16..HZ16,B26..HZ26)
A:ID17:   @SUMPRODUCT(B17..HZ17.B26..HZ26)
A:IE17:      @SUMPRODUCT(B18..HZ18,B26..HZ26)
A:IF17:      @SUMPRODUCT(B19..HZ19,B26..HZ26)
A:IG17:     (IC17+10*ID17)^2+5*(IE17-IF17)^2+(ID17-2*IE17)^4+10*(IC17-IF17)^4
A:IC18:     a1
A:ID18:     a2
A:IE18:     a3
A:IF18:     a4
A:IG18:     fmax
A:IC19:     +IA21
A:ID19:     +IA22
A:IE19:     +IA23
A:IF19:     +IA24
A:IG19:      @MAX(B20..HZ20)
A:IA20:     @SUM(B20..HZ20)
A:IC20:     x1Fk-a1
A:ID20:     x2Fk-a2
A:IE20:     x3Fk-a3
A:IF20:     x4Fk-a4
```

```
A:IA21:  (@SUMPRODUCT(B21..HZ21,B26..HZ26)/($G$8+@ABS(@SUMPRODUCT(B21..HZ21.B26..H
         Z26)))*@ABS(@SUMPRODUCT(B21..HZ21,B26..HZ26))^(1/3))
A:IC21:  +IC17-IC19
A:ID21:  +ID17-ID19
A:IE21:  +IE17-IE19
A:IF21:  +IF17-IF19
A:IA22:  (@SUMPRODUCT(B17..HZ17,B26..HZ26)/($G$8+@ABS(@SUMPRODUCT(B17..HZ17,B26..H
         Z26)))*@ABS(@SUMPRODUCT(B17..HZ17,B26..HZ26))^(1/3))
A:IA23:  (@SUMPRODUCT(B18..HZ18,B26..HZ26)/($G$8+@ABS(@SUMPRODUCT(B18..HZ18,B26..H
         Z26)))*@ABS(@SUMPRODUCT(B18..HZ18,B26..HZ26))^(1/3))
A:IA24:  (@SUMPRODUCT(B19..HZ19,B26..HZ26)/($G$8+@ABS(@SUMPRODUCT(B19..HZ19,B26..H
         Z26)))*@ABS(@SUMPRODUCT(B19..HZ19,B26..HZ26))^(1/3))
A:IA25:  @SUM(B25..HZ25)
A:IA26:  @SUM(B26..HZ26)
```

^	GE	GF	GG	GH	O	Oc	O *	GL	GM	GN	GO	GP
ro	Rodada		ro	co		cn	o)	J	co	cn	o	
00	X	3.133586	3.133586	3.133586	3.133586	3.133586	1.042202	1.042202	1.454002	-0.35027	0.554892	0.446719
		1 -0.28257	-0.28257	-0.28257	-0.28257	-0.28257	-0.26694!	-0.26694!	-0.30837	0.685884	-0.36798	-0.37448
cn	X co	2.076064	2.076064	2.076064	ro o Bo0	2.076064	0.603403	0.603402	1.254271	00E5E0	0.425877	0.451051
o	X	1.949719	1.949719	1.949719	1.949719	1.949719	-1.3659	-1.3659	2.703774	0.535452	o5 8 e88	0.620144
04	X "1 b ?n	1406.5913	406.5913	406.5913	406.5913	406.5913	363.0347	363.0347	100.5211	71.85323	30.5452	13.68492
00	fmax	-406.591	-406.591	-406.591	-406.591	-406.591	-363.035	-363.035	-100.521	-71.8532	-30.5452	-13.6849

Função de teste **14**

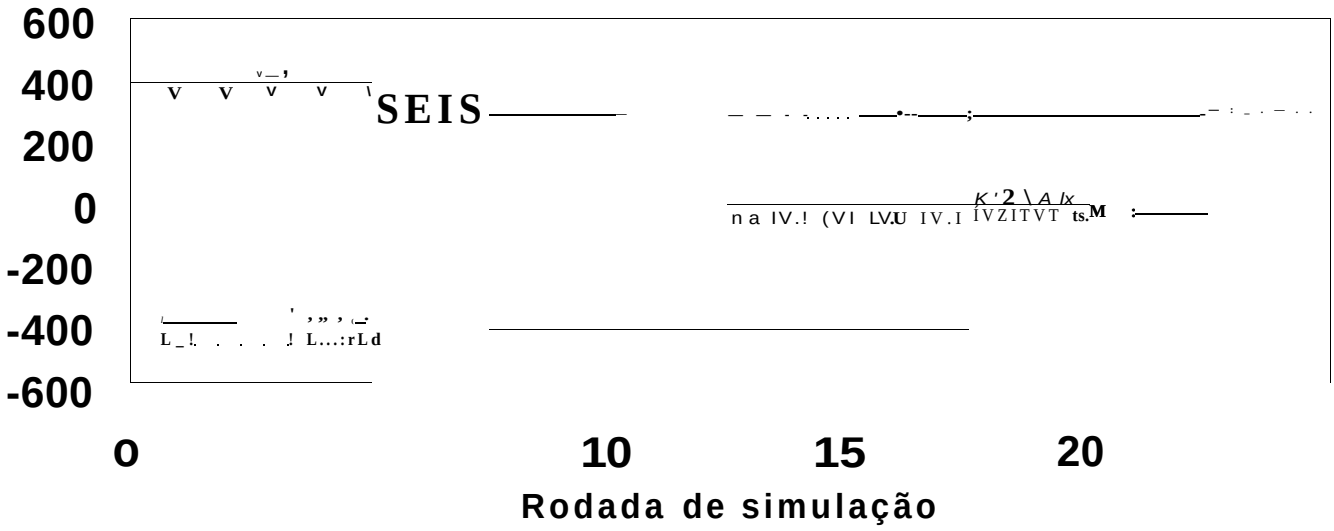
Comportamento de **x1F, x2F, x3F** e **x4F**



x1F ^ x2F v x3F è x4F

Função de teste 14

Comportamento de f(x1 ,x2,x3,x4) e fmax



f(x1,x2,x3,x4) i > fmax

Anexo Q

Função de teste 15
Parâmetros
Da função
Do algoritmo
Delta 1
Delta 2
Delta 3
Gama 1
Gama 2
Gama 3
Erro
1E-15
0.003557
0.019805

tem ordem elevadas. Esse método tem a importante propriedade de, cm n passos, a matriz direciona] tornar-sc igual a matriz inversa da 1 lessiana para funções quadráticas com hessiana positiva definida.

A matriz direcional inicial, geralmente, é escolhida igual à matriz identidade. Transformações graduais dc direção de gradiente vão se processando a cada passo, para direções de Newton, extraindo-sc desse fato as boas fases de comportamento do método dc Cauchy (longe do ponto de ótimo) c de Newton (na vizinhança do ótimo).

O método dc Davidon - Fletcher - Powcll pertence a classe dos métodos dc métrica variável, no entanto pode ser enquadrado na categoria dos que usam direções conjugadas. Para uma função qualquer é esse fato, mais que o dc aproximar a inversa da matriz I lessiana, a razão dc sua grande eficiência.

A convergência para esse algoritmo é garantida para função objetivo quadrática' com I lessiana positiva definida, c para funções necessariamente quadráticas e estritamente convexas.

iv) Método dc Broydcn

Rssc método, como o anterior, é da classe dos métodos dc métrica variável, a diferença entre cies está no processo dc geração da matriz direcional. Sua convergência é demonstrada apenas para funções quadráticas com I lessiana positiva definida. Durante sua execução deve-sc tomar algumas precauções para que a convergência não seja afetada pela manipulação de precisões nas buscas unidirecionais, embora sc possa afetar a velocidade de convergência, ver [Ribeiro].

2.3.2 Otimização vinculada

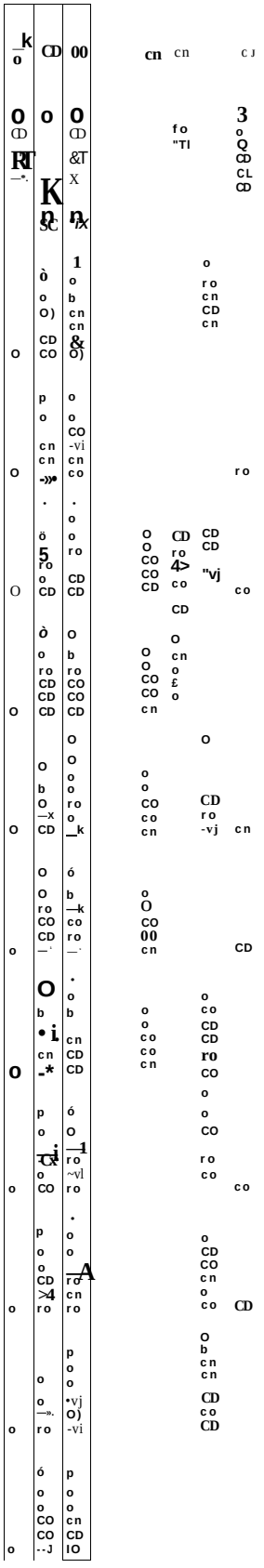
2.3.2.1 Sem derivadas

i) Método dc tolerâncias flexíveis

Cálculo 2 0.255649 1.594597 1.067683 0.716224 0.847906

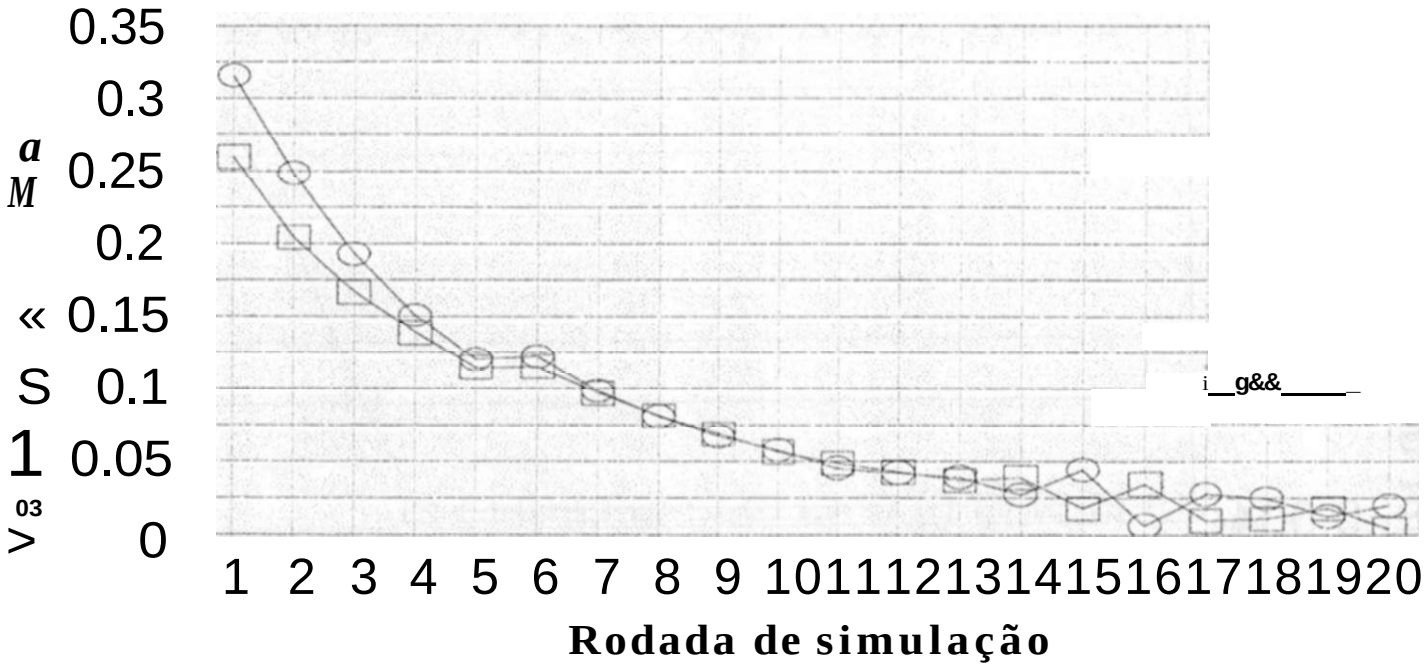
```
A:A22:    Iteração
A:B22:    1
A:A23:    x1
A:B23:      +$B$9+($B$10-$B$9)*@RAND
A:C23:      +$B$9+($B$10-$B$9)*@RAND
A:A24:    x2
A:B24:      +$B$11+($B$12-$B$11)*@RAND
A:C24:    +$B$11  +($B$12-$B$11 )*@RAND
A:A25:    f(x1 ,x2)
A.B25:    -0.5*B23^2-0.5*(1    -@COS(2*B23))-B24^2
A:C25:    -0.5*C23^2-0.5*(1    -@COS(2*C23))-C24^2
A:A26:    '(x1-x1fk)^3
A:B26:    (B23-$GY25)^3
A:C26:    (C23-$GY25)^3
A:A27:    '(x2-x2fk)^3
A:B27:    (B24-$GZ25)^3
A:C27:    (C24-$GZ25)^3
A:A28:    Fk(x1,x2)
A:B28:      @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28:      @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29:    Fk/SumFk
A:B29:      +B28/($D$14+$GU28)
A:C29:      +C28/($D$14+$GU28)
A:A30:    Curare
A.B30:      ($HA27-B25)^(1/($F$10+($HA27-B25)))
A.C30:    ($HA27-C25)^(1    /($F$10+($HA27-C25)))
A:A31:    x1Fk-a1k-x1ik
A:B31:    +$GY29-B23
A:C31:    +$GY29-C23
A:A32:    x2Fk-a2k-x2ik
A:B32:    +$GZ29-B24
A:C32:    +$GZ29-C24
A:A33:    f(x1Fk-a1k,x2ik)
A:B33:    -0.5*$GY29^2-0.5*(1    -@COS(2*$GY29))-B24^2
A:C33:    -0.5*$GY29^2-0.5*(1    -@COS(2*$GY29))-C24^2
A:A34:    f(x1 ik,x2Fk-a2k)
A:B34:    -0.5*B23^2-0.5*(1    -@COS(2*B23))-$GZ29^2
A:C34:    -0.5*C23^2-0.5*(1    -@COS(2*C23))-$GZ29^2
A:A35:    Sinal 1
A:B35:      (B33-B25)/($D$14+@ABS(B33-B25))
A:C35:    (C33-C25)/($D$14+@ABS(C33-C25))
A:A36:    Sinal 2
A:B36:      (B34-B25)/($D$14+@ABS(B34-B25))
A:C36:    (C34-C25)/($D$14+@ABS(C34-C25))
A:A37:    Passo
A:B37:    +$D$10*@EXP(-($E$10/$B22))
A:C37:    +$D$10*@EXP(-($E$10/$B22))
A:A38:    Cálculo 1
A:B38:      +B23+B37*B30^B35*B31
A:C38:    +C23+C37*C30*C35*C31
A:A39:    Cálculo 2
A:B39:      +B24+B37*B30*B36*B32
A:C39:    +C24+C37*C30*C36*C32
A:GU23:   @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23:   @AVG(B23..GT23)
A:GW23:   @STD(B23..GT23)
A:GX23:   @MAX(B23..GT23)
```

A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A.GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: +0.5*$GY24^2+0.5*(1 -@COS(2*$G Y24))+$GZ24^2
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT 26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT 27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27



Função de teste 15

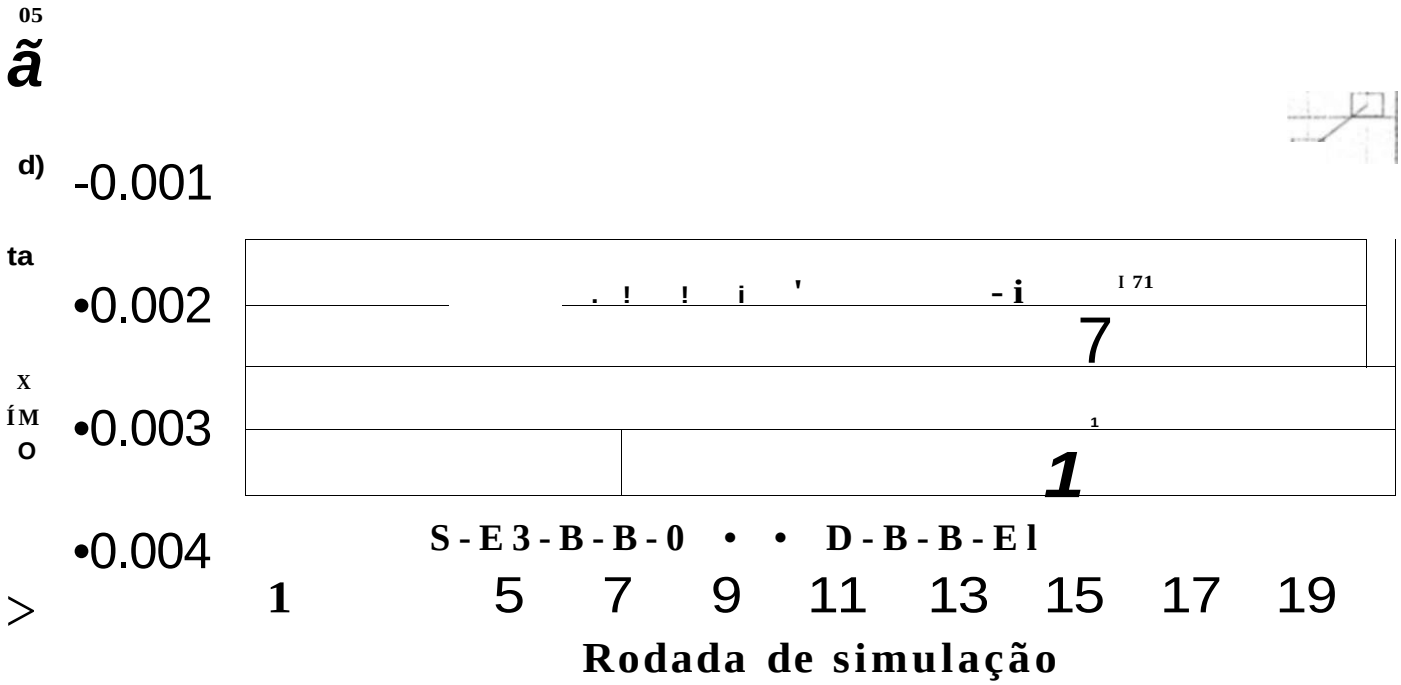
Comportamento de x1 Fk e x2Fk



-ra Valor de x1Fk -e- Valor x2Fk

Função de teste 15

Comportamento de f(x1Fk,x2Fk) e fmax



e- Valor de fmax **A** Valor de f(x1Fk,x2Fk)

Função de teste 15

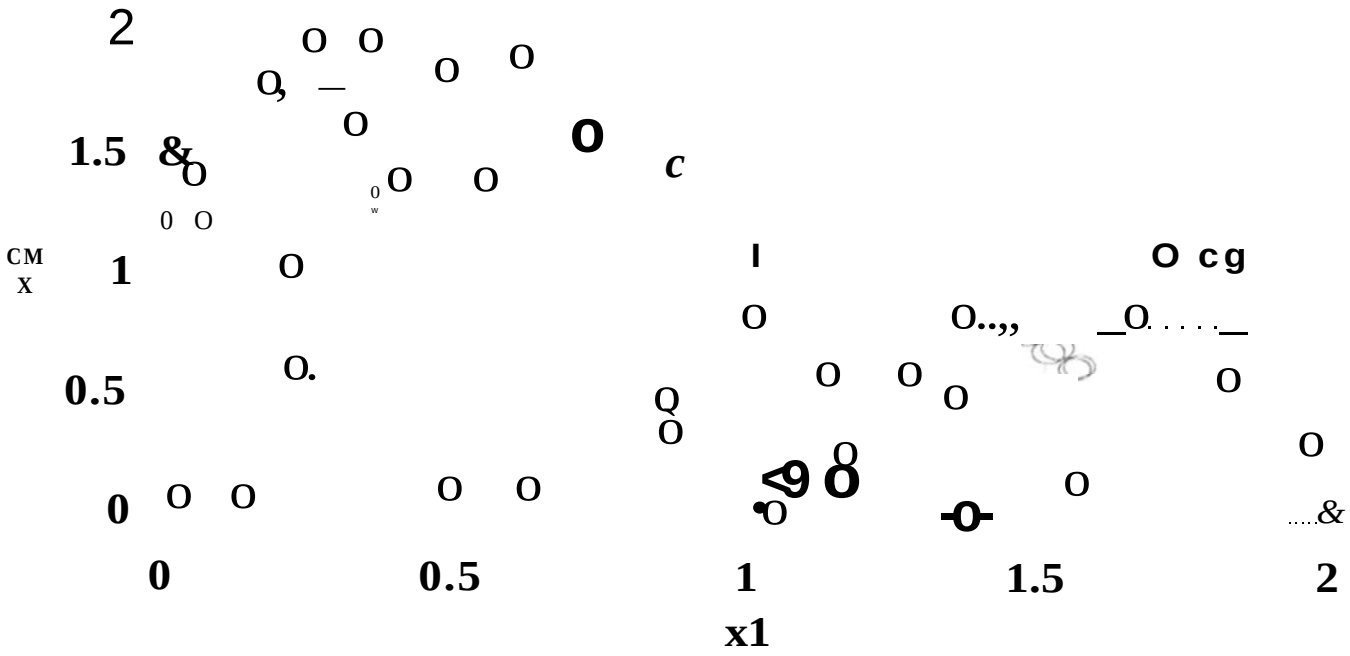
Comportamento dos Deltas

Delta x1 Fk Delta x2Fk

Delta f(x1Fk,x2Fk)

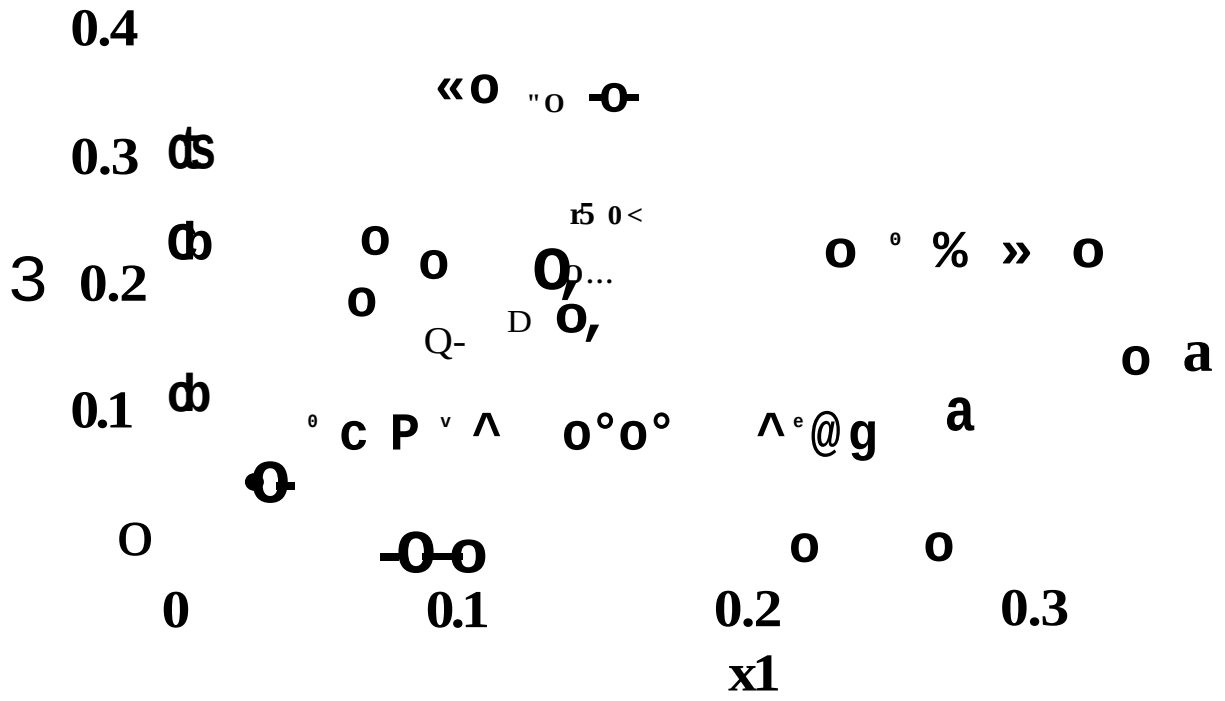
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Décima rodada de simulação



Esse processo c uma extensão do método do poliedro flexível já visto anteriormente e é devido a Paviani e Himmclblau Como não necessita de derivadas da função objetivo e das restrições agiliza bastante a preparação dos dados por parte do usuário comeomitantemente a um bom desempenho em relação aos diversos tipos de problemas que são utilizados para seu teste. **0** objetivo c resolver o problema 2**.1.1.1.**

Adola-se um critério de tolerância de violação de restrições durante sua execução, pois o poliedro flexível sofre contrações e expansões ao longo de seu processamento, todavia, tende a diminuir de tamanho a medida que se aproxima do ponto procurado, o algoritmo gera uma seqüência positiva não crescente {<l\} k c N que é utilizado como critério de tolerância c também como critério de parada.

Define-se o critério de tolerância flexível de viabilidade na k-ésima iteração como:

onde

<\\ é o valor do critério de tolerância na k-ésima iteração

$O_{\cdot}$.| é o valor do critério de tolerância na (k-l)-ésima iteração

1 é o número de restrições de igualdade

p = n-l número de graus de liberdade

Xj é o i-ésimo vértice do poliedro llexível

$\mathbf{Xp}_{)_2}$ é o centróide do poliedro flexível

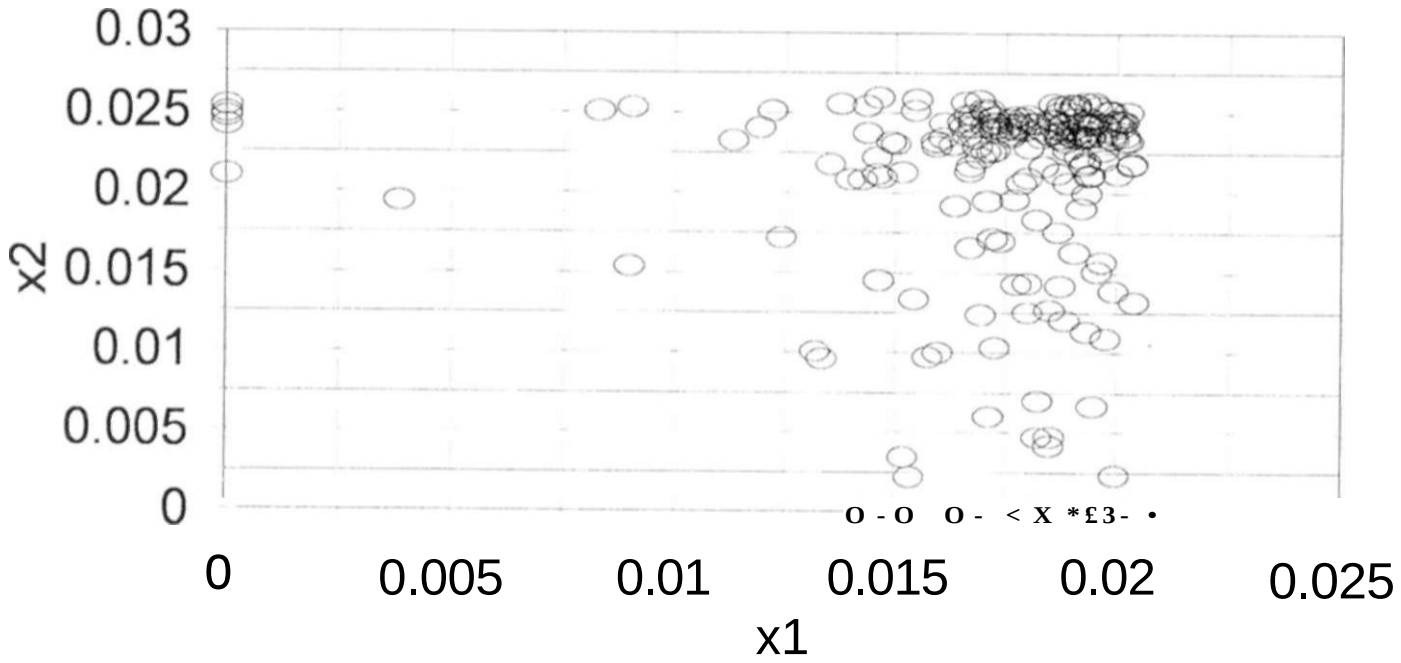
t tamanho inicial do poliedro

E o valor inicial $\Phi_{\text{„}}$ é calculado como O_{0} 2(1 H)t.

A seqüência (Ok) é então, claramente, positiva não crescente, isto é:

Nuvem de pontos

Vigésima rodada de simulação



Anexo R

Função de teste 16

Proposto em [Himmelblau] pag. 195tabela 5.2-1

Parâmetros

Da função

Do algoritmo

xlmax	3
Gama 1	0
Gama 2	0
Gama 3	0
Delta 1	0
Delta 2	0
Delta 3	0
Erro	**1E-15**
x1Fk	**0.988733**
K	**0.975323**

0.000132

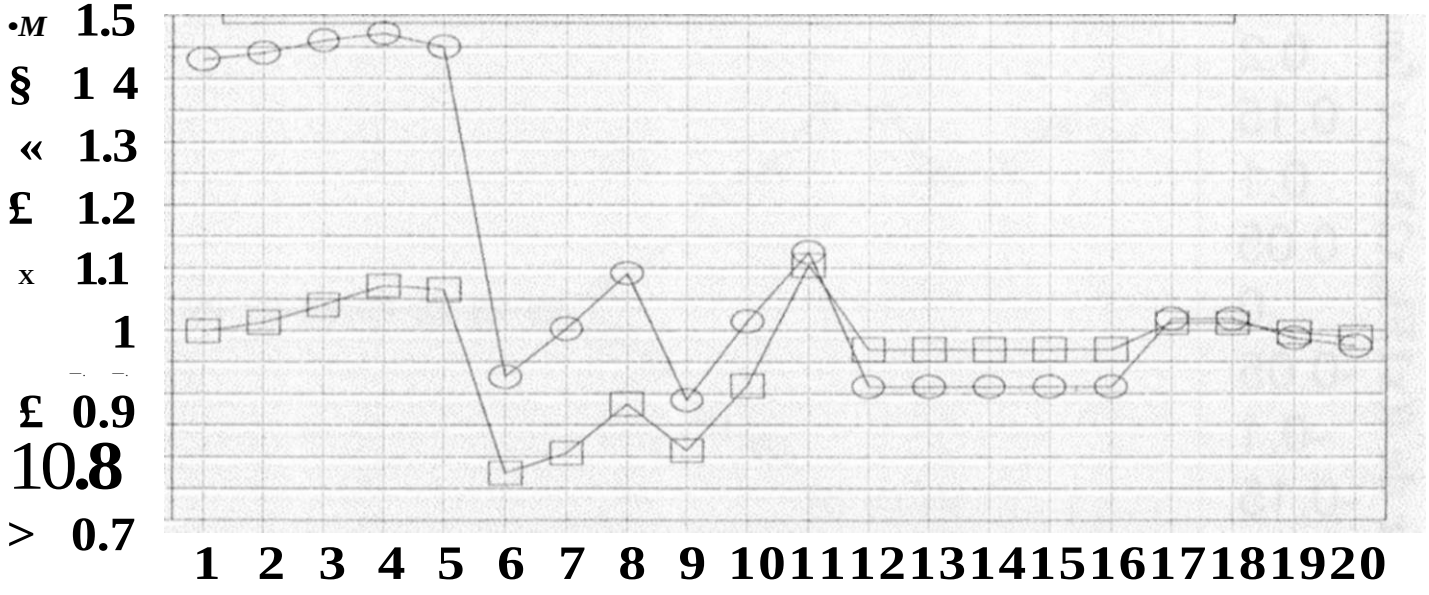
A:A22: Iteração
A:B22: 1
A:A23: x1
A:B23: +$B$9+($B$10-$B$9)*@RAND
A:C23: +$B$9+($B$10-$B$9)*@RAND
A:A24: x2
A:B24: +$B$11 +($B$12-$B$11)*@RAND
A:C24: +$B$11+($B$12-$B$11)*@RAND
A:A25: f(x1,x2)
A:B25: -(B24-B23^2)^2-(1-B23)^2
A:C25: -(C24-C23^2)^2-(1-C23)^2
A:A26: '(x1-x1fk)^3
A:B26: (B23-$GY25)^3
A:C26: (C23-$GY25)^3
A.A27: "(x2-x2fk)^3
A:B27: (B24-$GZ25)^3
A:C27: (C24-$GZ25)^3
A:A28: Fk(x1,x2)
A:B28: @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28: @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A.A29: Fk/SumFk
A:B29: +B28/($D$14+$GU28)
A:C29: +C28/($D$14+$GU28)
A:A30: Curare
A:B30: ($HA27-B25)^(1/($F$10+($HA27-B25)))
A:C30: ($HA27-C25)^(1 /($F$10+($HA27-C25)))
A:A31: x1Fk-a1k-x1ik
A:B31: +$GY29-B23
A:C31: +$GY29-C23
A:A32: x2Fk-a2k-x2ik
A:B32: +$GZ29-B24
A:C32: +$GZ29-C24
A:A33: f(x1Fk-a1k,x2ik)
A:B33: -(B24-$GY29^2)^2-(1-$GY29)^2
A:C33: -(C24-$GY29^2)^2-(1 -$GY29)^2
A:A34: f(x1ik,x2Fk-a2k)
A:B34: -($GZ29-B23^2)^2-(1-B23)^2
A:C34: -($GZ29-C23^2)^2-(1 -C23)^2
A:A35: Sinal 1
A:B35: (B33-B25)/($D$14+@ABS(B33-B25))
A:C35: (C33-C25)/($D$14+@ABS(C33-C25))
A:A36: Sinal 2
A:B36: (B34-B25)/($D$14+@ABS(B34-B25))
A:C36: (C34-C25)/($D$14+@ABS(C34-C25))
A:A37: Passo
A:B37: +$D$10*@EXP(-($E$10/$B22))
A:C37: +$D$10*@EXP(-($E$10/$B22))
A:A38: Cálculo 1
A:B38: +B23+B37*B30*B35*B31
A:C38: +C23+C37*C30*C35*C31
A:A39: Cálculo 2
A:B39: +B24+B37*B30*B36*B32
A:C39: +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23: @AVG(B23..GT23)
A:GW23: @STD(B23..GT23)
A:GX23: @MAX(B23. .GT23)

```
A:GY23:  Resultado da simulação
A:GU24:  @SUMPRODUCT(B24..GT24.B25..GT25)/$GU25
A:GV24:  @AVG(B24..GT24)
A:GW24:  @STD(B24..GT24)
A:GY24:  x1Fk
A:GZ24:  x2Fk
A:HA24:  f(x1Fk,x2Fk)
A:GU25:  @SUM(B25 .GT25)
A:GY25:  @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25:  @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25:  (GZ25-GY25^2)^2+(1 -GY25)^2
A:GU26:  (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT
         26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26:  a1k
A:GZ26:  a2k
A:HA26:  fmax
A:GU27:  (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT
         27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27:  +GU26
A.GZ27:  +GU27
A:HA27:  @MAX(B25..GT25)
A:GU28:  @SUM(B28..GT28)
A:GY28:  x1Fk-a1k
A:GZ28:  x2Fk-a2k
A:GY29:  +GY25-GY27
A:GZ29:  +GZ25-GZ27
```

	0.96984«	0.969848	0.969848	0.969848	1.012305	1.012305	0.996831	0.988733
	0.910423	0.910423	0.910423	0.910423	1.018724	1.018724	0.987491	0.975323
	-0.00182	-0.00182	-0.00182	-0.00182	-0.00019	-0.00019	-0.00019	-0.00015
	0.00182	0.00182	0.00182	0.001821	0.00182	0.000188	4.8E-05	0.000132
				-2.6E-15i		-8.1E-09	-0.0081	-0.98873
				1.6E-15		-1.1E-08	-0.97532	
						-2.7E-10	-0.00013	

Função de teste 16

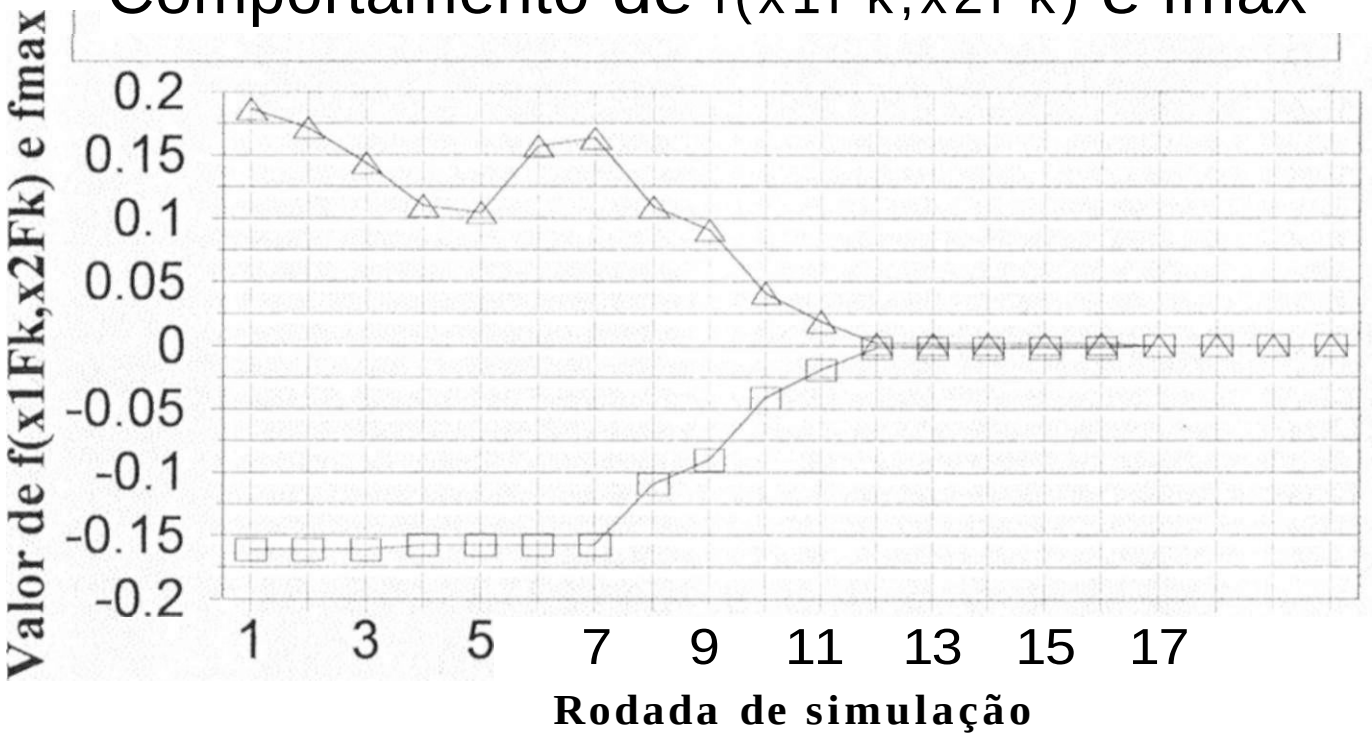
Comportamento de x1 Fk e x2Fk



s Valor de x1 Fk <> Valor x2Fk

Função de teste **16**

Comportamento de f(x1Fk,x2Fk) e fmax



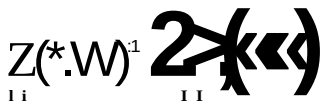
i ! Valor de fmax **A** Valor de f(x1Fk,x2Fk)

Oo > Oi > <D$_2$ > ...>O$_k$ > 0.

Como já dito, na vizinhança do ponto x* de f(x), o tamanho do poliedro tende para zero, o mesmo ocorrendo com O$_k$, e essa é a condição de parada do algoritmo.

Definc-sc, então, uma função T( ) que será usada para indicar se um ponto pertence ou não ao conjunto de pontos viáveis V.

Seja a função T:R" -> R definida como



onde Ui é o operador de I leaviside tal que Uj = 0 se g(x) < 0 e Uj - 1 se g(x) > 0.

Dessa forma T(-) é não negativa Vx e R\ Se além disso, ' ' e g( ) são convexas, T(-) é convexa com um ponto de mínimo global x, viável se T(x) = 0. Além disso, se T(x) > 0, x é não viável. Um conceito importante aqui, é o de quase viabilidade que ocorre quando T(x) = 0, nesse caso, x está muito próximo da região de viabilidade. Distingui-se os pontos por:

I) Viável se T(x) = 0

2) Não viável se T(x) > O$_k$

3) Quase viável se 0 < T(x) < (t>k

A região de quase viabilidade é, então V = { x:T(x) - O$_k$ < 0}.

A partir daí transformamos o problema 2.1.1.1 em um novo problema enunciado como

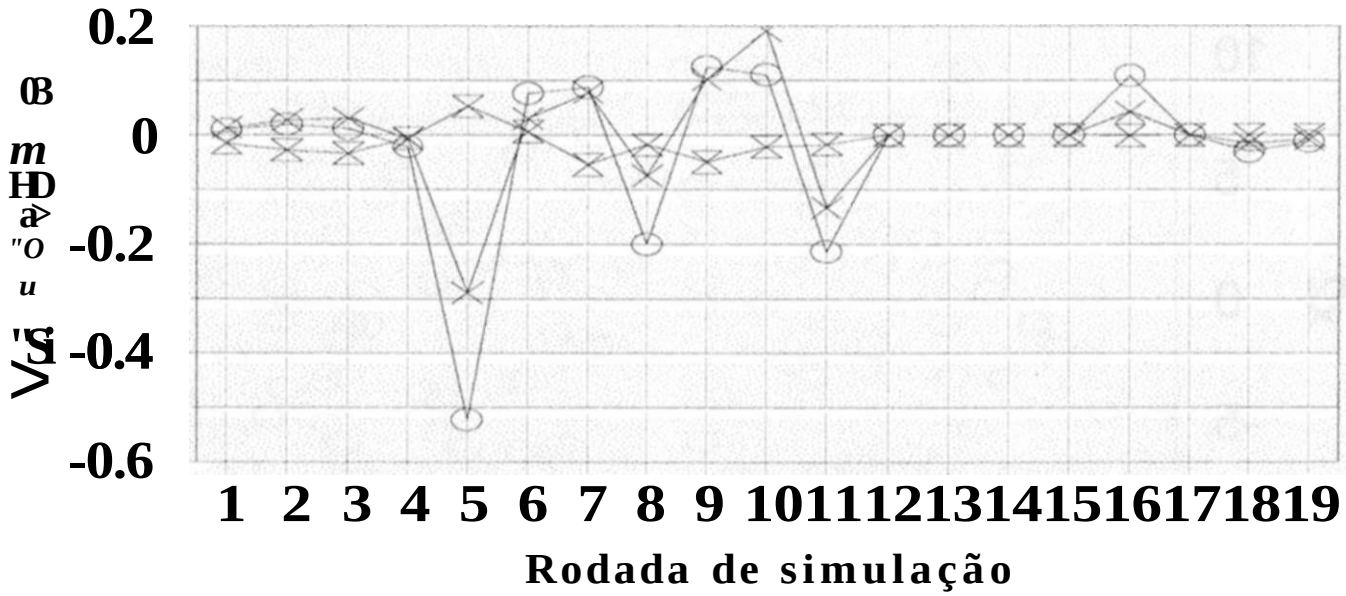
(Prob. 3.2.2.1.1) Minimize fi»

Sujeito a T(x) - O$_k$ < 0.

Agora, como no método do poliedro flexível, procura-se novos vértices para o poliedro, esses novos vértices devem ser viáveis ou quasc-viáveis, em cada iteração iremos minimizar T(x) até que T(x) < <D$_k$, que pode ser feito com qualquer método de minimização desvinculada sem derivadas.

Função de teste 16

Comportamento dos Deltas

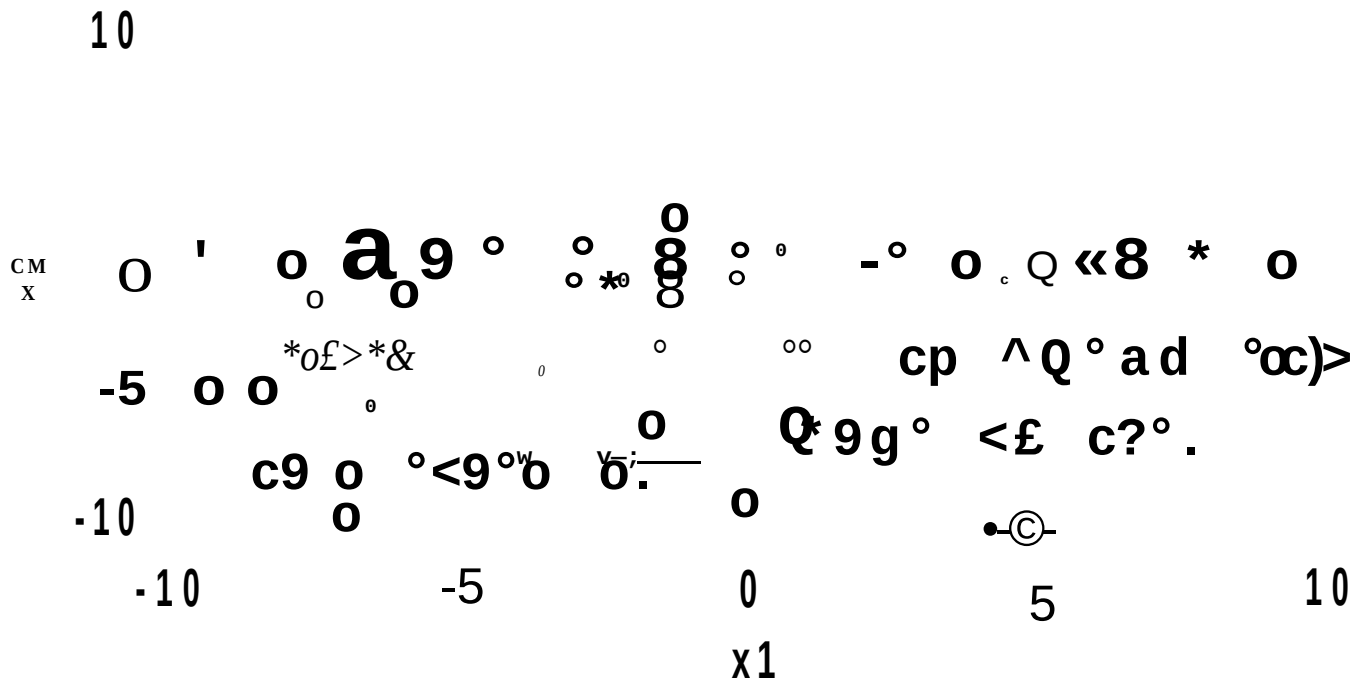


x Delta x1 Fk o Delta x2Fk

Delta f(x1 Fk,x2Fk)

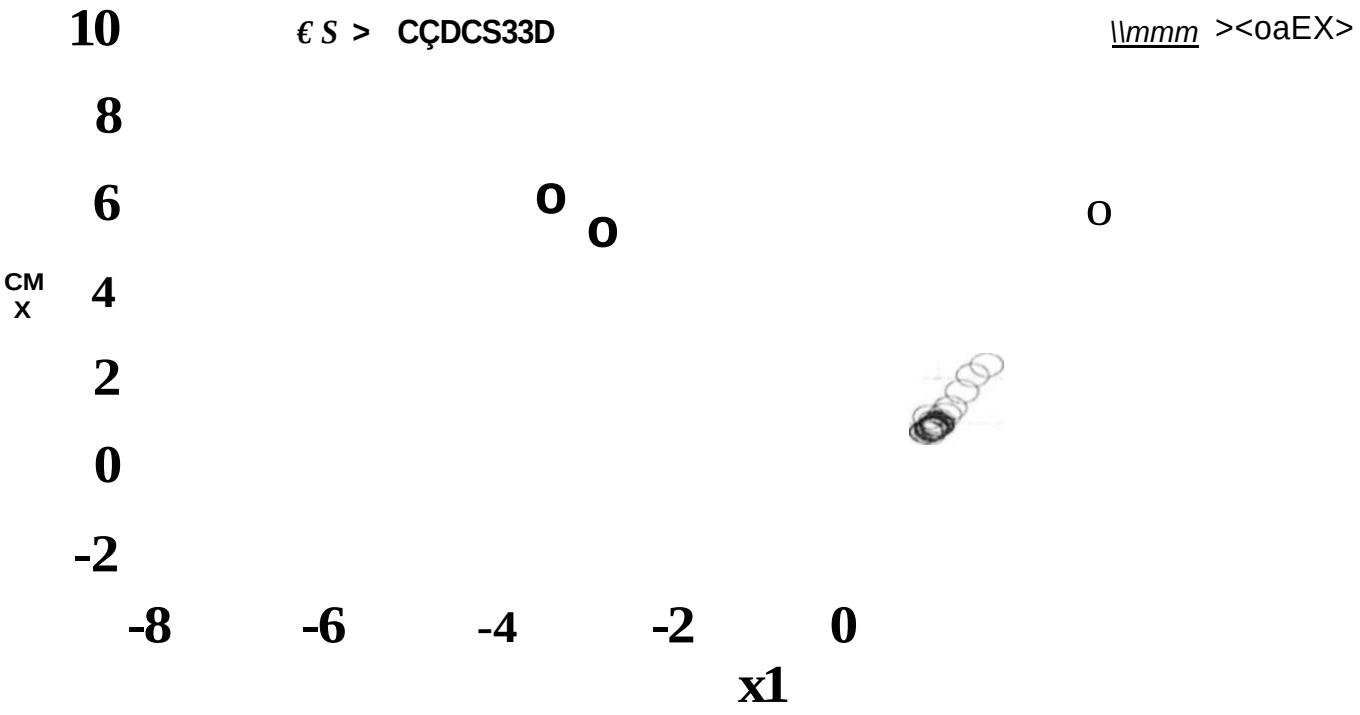
Nuvem de pontos

Primeira rodada de simulação

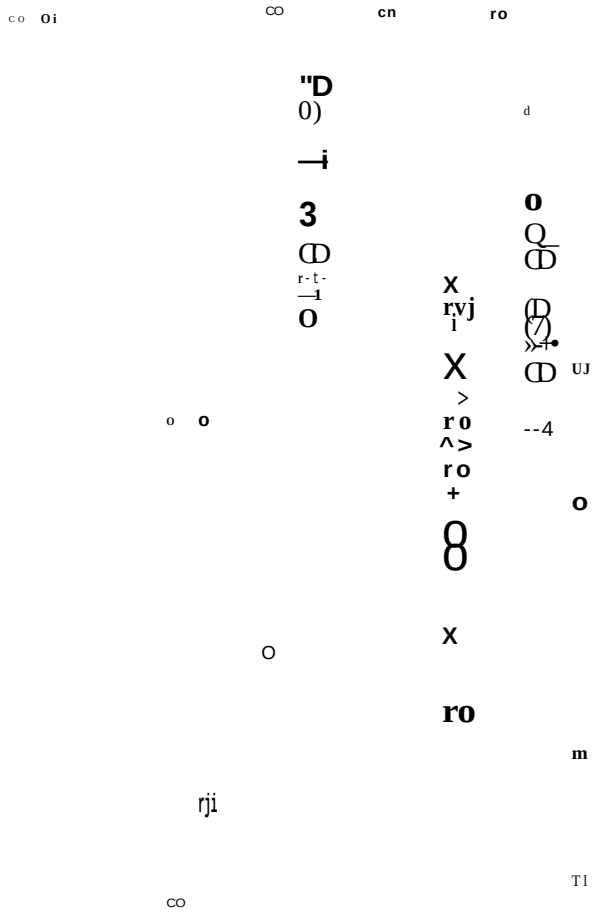


Nuvem de pontos

Vigésima rodada de simulação



Anexo S



A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	0.114397	-9.63675	-6.88223	-6.57088	0.140925	0.012244	5.718967	9.989067	Resultado da simulação		
24	x2	-4.72947	4.914086	7.344014	-3.1017	-0.8751	-0.47259	5.986325		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1 ,x2)	-100.921	-19049.7	-7814.65	-7873.49	-1093628				1.015274	2.216412	1.42905
26	(x1-x1fk)^3	-0.73113	-1208.64	-492.572	-436.581	1.5E-45				a1k	a2k	frnax
27	(x2-x2fk)^3	-335.106	19.63217	134.8164	-150.408	2.4E-44				1.5E-45	2.4E-44	-1.42905
28	Fk(x1 ,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					1.015274	2.216412	
30	Curare	1.023327	1.000515	1.001133	1.001126							
31	x1Fk-a1k-x1ik	0.900877	10.65202	7.897506	7.586154							
32	x2Fk-a2k-x2ik	6.945883	-2.69767	-5.1276	5.318108							
33	f(x1Fk-a1k,x2ik)	-33.2038	-15.1034	-39.8802	-17.1007							
34	f(x1ik,x2Fk-a2k)	-83.2839	-19531.6	-8251.37	-7409.55							
35	Sinal 1	1	1	1	1							
36	Sinal 2	1	-1	-1	1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	0.117503	-9.60084	-6.8556	-6.54529							
39	Cálculo 2	-4.70553	4.923179	7.361308	-3.08376							
40												
41	Iteração	2										
42	x1	0.117503	-9.60084	-6.8556	-6.54529	0.144296	0.015611	5.69967	9.958819	Resultado da simulação		
43	x2	-4.70553	4.923179	7.361308	-3.08376	-0.85582	-0.45589	5.976459		x1Fk	x2Fk	f(x1Fk,x2F
44	f(x1 ,x2)	-100.152	-18850.9	-7742.2	-7802.22	-1083434				1.015274	2.216412	1.42905
45	(x1-x1fk)^3	-0.7236	-1196.46	-487.605	-432.179	1.5E-45				a1k	a2k	frnax
46	(x2-x2fk)^3	-331.652	19.83137	136.1852	-148.891	2.4E-44				1.5E-45	2.4E-44	-1.42905
47	Fk(x1,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					1.015274	2.216412	
49	Curare	1.023378	1.00052	1.001143	1.001135							
50	x1Fk-a1k-x1ik	0.897771	10.61612	7.87087	7.560567							
51	x2Fk-a2k-x2ik	6.921937	-2.70677	-5.1449	5.300171							
52	f(x1Fk-a1k,x2ik)	-32.9285	-15.1741	-40.0989	-16.9528							
53	f(x1ik,x2Fk-a2k)	-82.7316	-19330.6	-8176.54	-7343.49							
54	Sinal 1	1	1	1	1							
55	Sinal 2	1	-1	-1	1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	0.155211	-9.16491	-6.53219	-6.23464							
58	Cálculo 2	-4.41479	5.034329	7.572709	-2.86598							

```
A:A22:      Iteração
A:B22:      1
A:A23:      x1
A:B23:       +$B$9+($B$10-$B$9)*@RAND
A:C23:       +$B$9+($B$10-$B$9)*@RAND
A:A24:      x2
A:B24:      +$B$11+($B$12-$B$11  )*@RAND
A:C24:       +$B$11+($B$12-$B$11)*@RAND
A:A25:      f(x1 ,x2)
A:B25:      -(B24-B23^2)^2-100*(1   -B23)^2
A:C25:      -(C24-C23^2)^2-100*(1  -C23)^2
A:A26:       '(x1-x1fk)^3
A:B26:      (B23-$GY25)^3
A:C26:      (C23-$GY25)^3
A:A27:       '(x2-x2fk)^3
A:B27:      (B24-$GZ25)^3
A:C27:      (C24-$GZ25)^3
A:A28:      Fk(x1,x2)
A:B28:      @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28:      @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29:     Fk/SumFk
A:B29:       +B28/($D$14+$GU28)
A:C29:       +C28/($D$14+$GU28)
A:A30:     Curare
A:B30:       ($HA27-B25)^(1/($F$10+($HA27-B25)))
A:C30:       ($HA27-C25)^(1/($F$10+($HA27-C25)))
A:A31:     x1Fk-a1k-x1ik
A:B31:      +$GY29-B23
A:C31:      +$GY29-C23
A:A32:     x2Fk-a2k-x2ik
A:B32:      +$GZ29-B24
A:C32:      +$GZ29-C24
A:A33:     f(x1Fk-a1k,x2ik)
A:B33:     -(B24-$GY29^2)^2-100*(1  -$GY29)^2
A:C33:     -(C24-$GY29^2)^2-100*(1  -$GY29)^2
A:A34:     f(x1ik,x2Fk-a2k)
A:B34:     -($GZ29-B23^2)^2-100*(1   -B23)^2
A:C34:     -($GZ29-C23^2)^2-100*(1  -C23)^2
A:A35:     Sinal 1
A:B35:      (B33-B25)/($D$14+@ABS(B33-B25))
A:C35:     (C33-C25)/($D$14+@ABS(C33-C25))
A:A36:     Sinal 2
A:B36:      (B34-B25)/($D$14+@ABS(B34-B25))
A:C36:     (C34-C25)/($D$14+@ABS(C34-C25))
A:A37:      Passo
A:B37:      +$D$10*@EXP(-($E$10/$B22))
A:C37:      +$D$10*@EXP(-($E$10/$B22))
A:A38:     Cálculo 1
A:B38:       +B23+B37*B30*B35*B31
A:C38:     +C23+C37*C30*C35*C31
A:A39:     Cálculo 2
A:B39:       +B24+B37*B30*B36*B32
A:C39:     +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23:  @AVG(B23..GT23)
A:GW23:  @STD(B23..GT23)
A:GX23:  r@MAX(B23..GT23)
```

A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24. .GT24.B29. .GT29)
A:HA25: (GZ25-GY25^2)^2+100*(1-GY25)^2
A:GU26: (@SUMPRODUCT(B26. .GT26.B28. .GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: frnax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

Demonstra-se que o problema 3.2.2.1.1 ó equivalente ao problema 2.1.1.1, 11 limmelblau]

2.3.2.2 dom derivadas

i) Métodos de penalidades

A idéia c reduzir a solução do problema 2.1.1.1 a uma seqüência dc problemas de minimização desvinculada da seguinte forma:

(Prob. 3.2.2.2.1) Min { f(x) t p,(x) : x c R" }, i - 1, 2,...

Onde p, : R^{n} > R, i 1,2,... são funções penalidades.

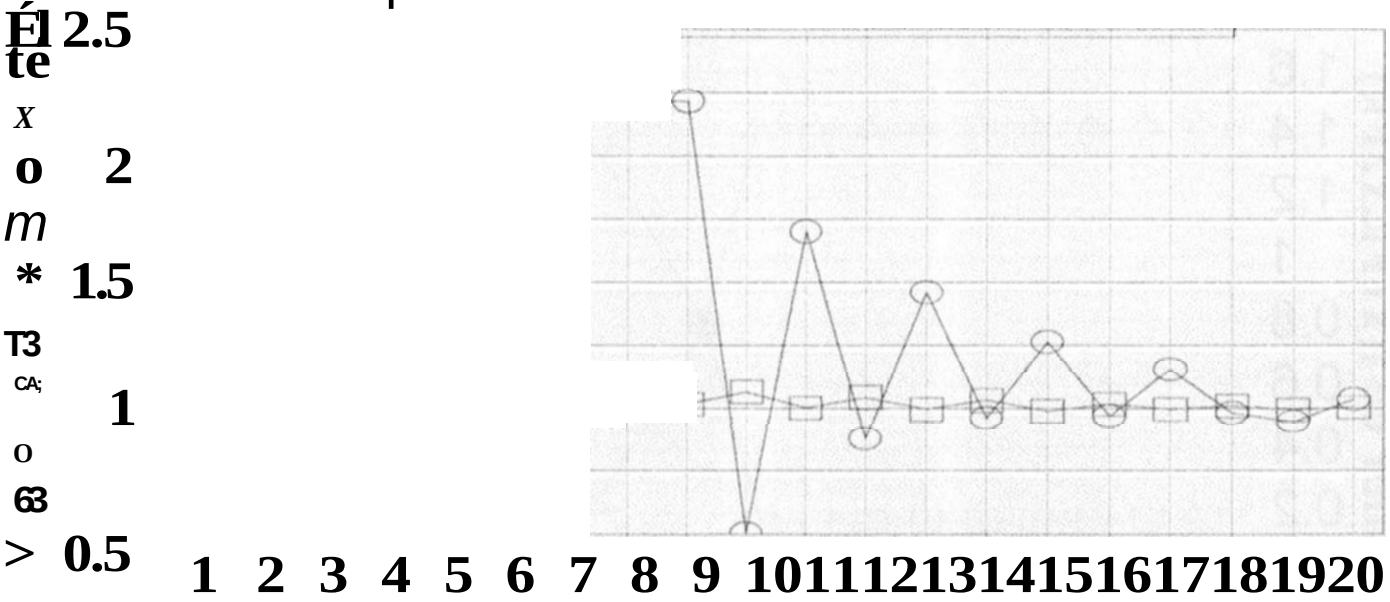
Esse método admite duas variantes básicas: penalidades exteriores c penalidades interiores, a primeira adiciona a f(x) um custo positivo sc x **t/"** V forçando as soluções dos problemas 3.2.2.2.1 a sc aproximar dc V c na segunda forçam-se essas soluções a permanecer no interior de V. O método dc penalidades admite ainda uma combinação das penalidades (interiores e exteriores) dc forma a gerar um método misto, podendo com isso usufruirmos das vantagens dc ambos.

Nas penalidades exteriores o problema geral de otimização 2.1.1.1 é transformado cm uma seqüência dc problemas de minimização não vinculada, define-se uma seqüência de funções contínuas atendendo a alguns requisitos (ver [Ribeiro], |1 Iimmclblau| c [AvricI|) c a compacticidadc do conjunto dc soluções viáveis, a partir daí dcmonstra-sc que se **Xj** é ótimo, a seqüência (Xj) é compacta e qualquer ponto dc acumulação dc (X,) é ótimo.

No caso dc penalidades interiores, da mesma forma, o problema geral 2.1 I. I é transformado cm uma seqüência dc problemas dc minimização desvinculada, no entanto, as penalidades agora são interiores, hipotetiza-sc que V é fechado, c que a aderência do interior de V é igual a V c não va/.ia, dclinc-sc uma seqüência dc funções contínuas, adiciona-se a hipótese dc compacticidadc dc V c então garante-sc que

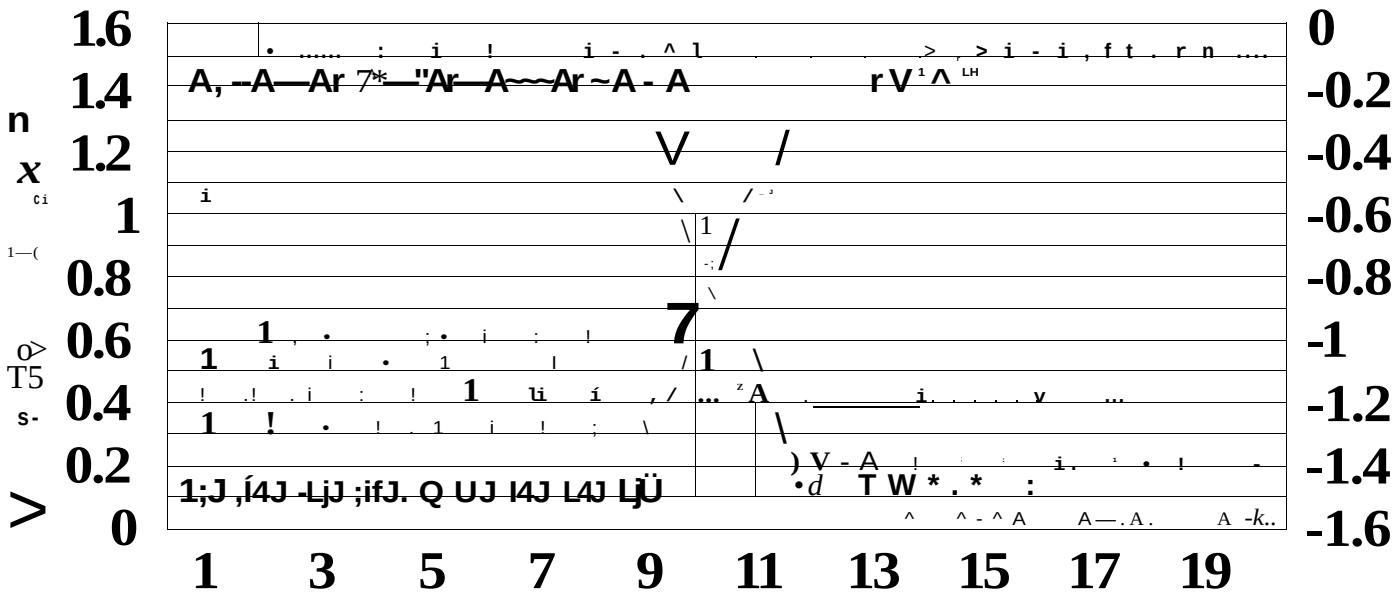
Função de teste 17

Comportamento de x1 Fk e x2Fk



Função de teste 17

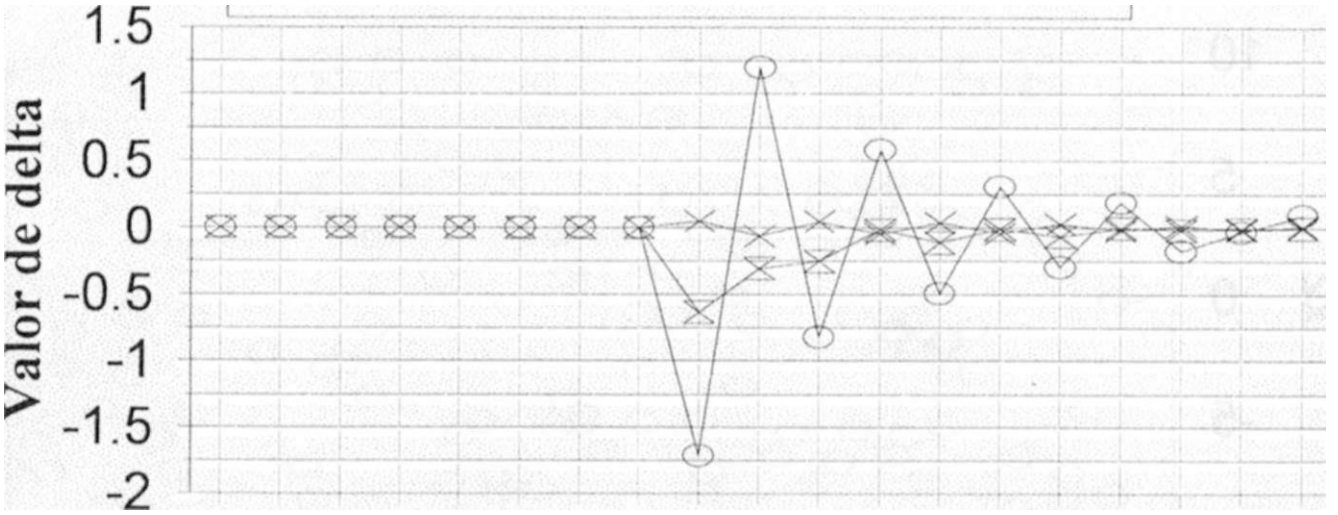
Comportamento de f(x1 Fk,x2Fk) e fmax



Valor de fmax **Valor de f(x1Fk,x2Fk)**

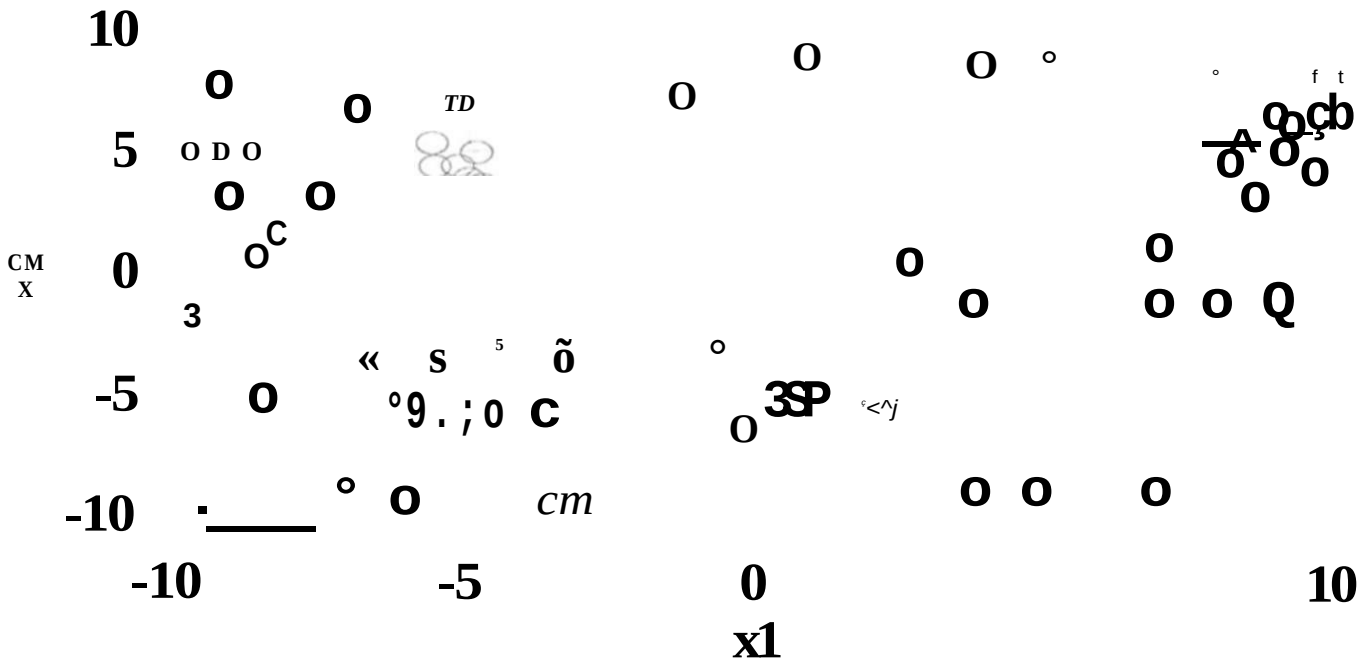
Função de teste **17**

Comportamento dos Deltas



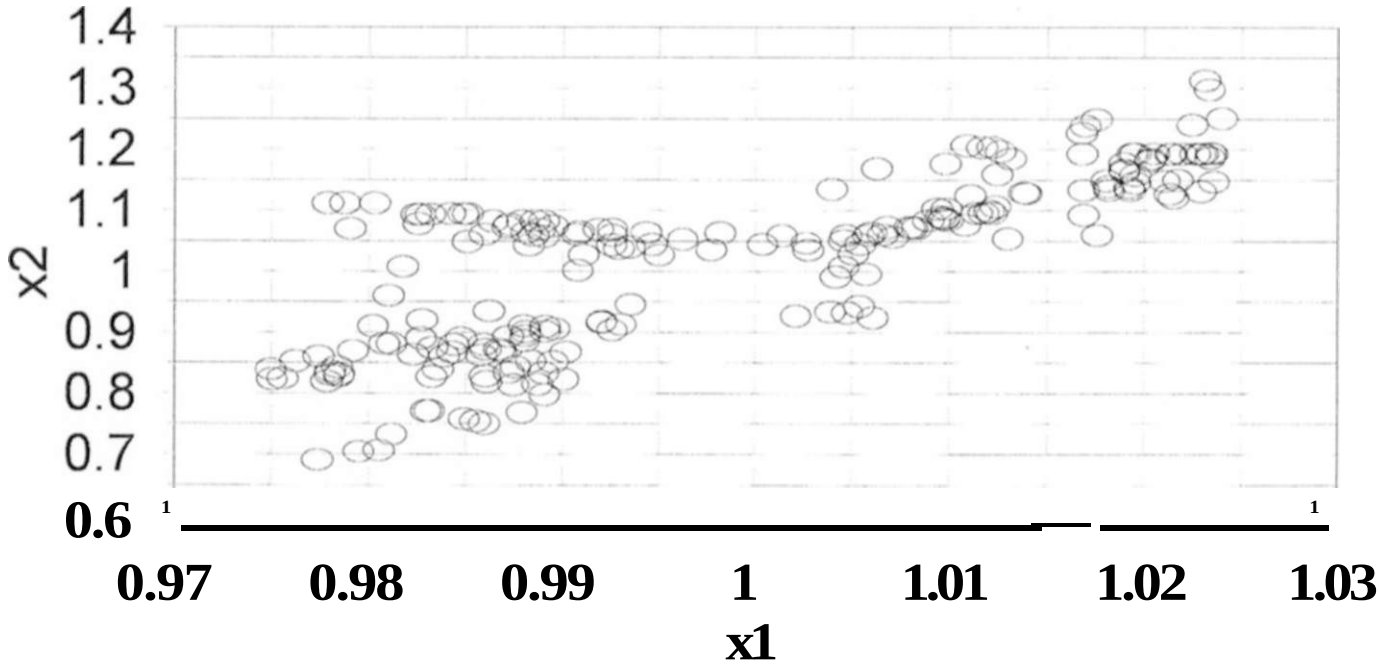
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação



Anexo T

Função de teste 18

$$f(x_1,x_2) = 0.1\left(x_1^2 + x_2^2 - 0.5\right)^2 + \left(x_1^2 + x_2^2\right)^2 + 0.5$$

Proposto em [Himmtelblau] pacf. 195tabela 5.2-1

Parâmetros

Da função	
	Gama 1 Gama 2 Gama 3
xlmax	Delta 1 Delta 2 Deita 3
	Erro
	1E-15 0.997577 0.99272
	6E-00

A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	9.185341	4.683929	8.673829	-3.17292	0.630119	-0.13009	5.534104	9.886037	Resultado da simulação		
24	x2	-5.36055	1.180598	5.763656	-4.92585	0.410919	0.545713	5.73315		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1,x2)	-6.1E+07	-1031883	-4.2E+07	-73011	-2.3E+09				-0.52892	-0.47523	13.04772
26	(x1-x1fk)^3	916.7046	141.6529	779.3865	-18.4835	-9.5E-47				a1k	a2k	fmax
27	(x2-x2fk)^3	-116.595	4.539916	242.8409	-88.1577	-6.2E-47				-9.5E-47	-6.2E-47	-13.0477
28	Fk(x1,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					-0.52892	-0.47523	
30	Curare	1	1.000013	1	1.000153							
31	x1Fk-a1k-x1ik	-9.71426	-5.21285	-9.20275	2.644001							
32	x2Fk-a2k-x2ik	4.885318	-1.65583	-6.23889	4.450616							
33	f(x1Fk-a1k,x2ik)	-2719.44	-178.847	-3497.07	-2285.15							
34	f(x1 ik,x2Fk-a2k)	-6E+07	-1065798	-4.3E+07	-99040.6							
35	Sinal 1	1	1	1	1							
36	Sinal 2	1	-1	-1	-1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	9.152614	4.666366	8.642825	-3.16401							
39	Cálculo 2	-5.34409	1.186177	5.784675	-4.94084							
40												
41	Iteração	2										
42	x1	9.152614	4.666366	8.642825	-3.16401	0.615382	-0.13108	5.515691	9.850949	Resultado da simulação		
43	x2	-5.34409	1.186177	5.784675	-4.94084	0.411782	0.545609	5.732752		x1Fk	x2Fk	f(x1Fk,x2F
44	f(x1,x2)	-6E+07	-1008508	-4.1E+07	-71488.1	-2.3E+09				-0.52892	-0.47523	13.04772
45	(x1-x1fk)^3	907.4707	140.226	771.5358	-18.2973	-9.5E-47				a1k	a2k	fmax
46	(x2-x2fk)^3	-115.42	4.585956	245.3035	-89.0518	-6.2E-47				-9.5E-47	-6.2E-47	-13.0477,
47	Fk(x1,x2)	0	0	0	0	1				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					-0.52892	-0.47523	
49	Curare	1	1.000014	1	1.000156							
50	x1Fk-a1k-x1ik	-9.68153	-5.19529	-9.17175	2.635092							
51	x2Fk-a2k-x2ik	4.868859	-1.66141	-6.25991	4.465612							
52	f(x1Fk-a1k,x2ik)	-2702.31	-180.332	-3521.96	-2299.5							
53	f(x1 ik,x2Fk-a2k)	-5.9E+07	-1042153	-4.2E+07	-97359							
54	Sinal 1	1	1	1	1							
55	Sinal 2	1	-1	-1	-1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	8.755259	4.453136	8.266394	-3.05585							
58	Cálculo 2	-5.14426	1.254366	6.041597	-5.12415							

```
A:A22:      Iteração
A:B22:      1
A:A23:      x1
A:B23:       +$B$9+($B$10-$B$9)*@RAND
A:C23:       +$B$9+($B$10-$B$9)*@RAND
A:A24:      x2
A:B24:       +$B$11+($B$12-$B$11)*@RAND
A:C24:       +$B$11+($B$12-$B$11)*@RAND
A:A25:      f(x1 ,x2)
A:B25:      -100*(B24-B23^3)^2-(1   -B23)^2
A:C25:      -100*(C24-C23^3)^2-(1  -C23)^2
A:A26:       '(x1-x1fk)^3
A:B26:      (B23-$GY25)^3
A:C26:      (C23-$GY25)^3
A:A27:       ''(x2-x2fk)^3
A:B27:      (B24-$GZ25)^3
A:C27:      (C24-$GZ25)^3
A:A28:      Fk(x1,x2)
A:B28:      @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28:      @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29:     Fk/SumFk
A:B29:       +B28/($D$14+$GU28)
A:C29:       +C28/($D$14+$GU28)
A:A30:     Curare
A:B30:      ($HA27-B25)^(1   /($F$10+($HA27-B25)))
A:C30:      ($HA27-C25)^(1  /($F$10+($HA27-C25)))
A:A31:      x1Fk-a1k-x1ik
A:B31:      +$GY29-B23
A:C31:      +$GY29-C23
A:A32:      x2Fk-a2k-x2ik
A:B32:      +$GZ29-B24
A:C32:      +$GZ29-C24
A:A33:      f(x1Fk-a1k,x2ik)
A:B33:      -100*(B24-$GY29^3)^2-(1  -$GY29)^2
A:C33:      -100*(C24-$G Y29^3)^2-(1 -$G Y29)^2
A:A34:      f(x1ik,x2Fk-a2k)
A:B34:      -100*($GZ29-B23^3)^2-(1   -B23)^2
A:C34:      -100*($GZ29-C23^3)^2-(1  -C23)^2
A:A35:      Sinal 1
A:B35:       (B33-B25)/($D$14+@ABS(B33-B25))
A:C35:      (C33-C25)/($D$14+@ABS(C33-C25))
A:A36:      Sinal 2
A:B36:       (B34-B25)/($D$14+@ABS(B34-B25))
A:C36:      (C34-C25)/($D$14+@ABS(C34-C25))
A:A37:      Passo
A:B37:      +$D$10*@EXP(-($E$10/$B22))
A:C37:      +$D$10*@EXP(-($E$10/$B22))
A:A38:      Cálculo 1
A:B38:       +B23+B37*B30*B35*B31
A:C38:      +C23+C37*C30*C35*C31
A:A39:      Cálculo 2
A:B39:       +B24+B37*B30*B36*B32
A:C39:      +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23:  @AVG(B23..GT23)
A:GW23:  @STD(B23..GT23)
A:GX23:  @MAX(B23..GT23)
```

A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: +100*(GZ25-GY25^3)^2+(1-GY25)^2
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

existe solução no interior de V. A partir de então demonstra-se que qualquer ponto da seqüência de soluções (Xj) dos problemas de penalidades interiores resolve o problema 2.1.1.1. A parada c feita a partir do gradiente da função objetivo dos problemas desvinculados, aumentando-sc a precisão das buscas a medida que crescem as penalidades, é exigido que todas as funções envolvidas sejam continuamente diferenciáveis.

|I limmclblau| apresenta o seguinte exemplo: suponha que queremos achar o mínimo dc

$f(x,,x,)$ $(x,-3)^2$ i $(x_2-2)^2$

Sujeito à $h(xi,x_2)$ **X| I** x_2 - 4 - 0, podemos formar uma nova função objetivo incorporando a restrição como penalidade como segue

$l^{)}(xi,x_2)$ - $(xi-3)^{7'}$ + $(x_2-2)^2$ i $(xi + x_2 - 4)^2$. Durante a minimização de $P(xi,x_2)$ o vetor solução c forçado pela penalidade a satisfazer a restrição cm algum grau.

ii) Método dc direções viáveis

Esse método **se** destina a resolver o problema

(Prob. 3.2.2.2.2) Minimizar b»

x t V - { x: **gj**(x) < 0, i - 1, 2, m } c as funções f(x) e gj(x), i 1,2, m são continuamente diferenciáveis.

Dado um ponto **Xj** e V, determina-se uma semi-reta {x:x = x; + À,Sj } passando pelo interior dc V, onde Sj e R^n e nessa semi-reta escolhe **Xj**,i= **Xj** + À,jSj tal que f(xj,i) < **ft^Xj),** assim esse método pode ser usado para resolver o problema 3.2.2.2.2 apenas se o interior de V for não vazio, desse modo somente scrao admitidos vínculos do tipo igualdade se estes torem lineares.

Esse método é uma extensão do processo de Cauchy, com a diferença que aqui xi+l calculado a partir dc x, deve está dentro da região viável.

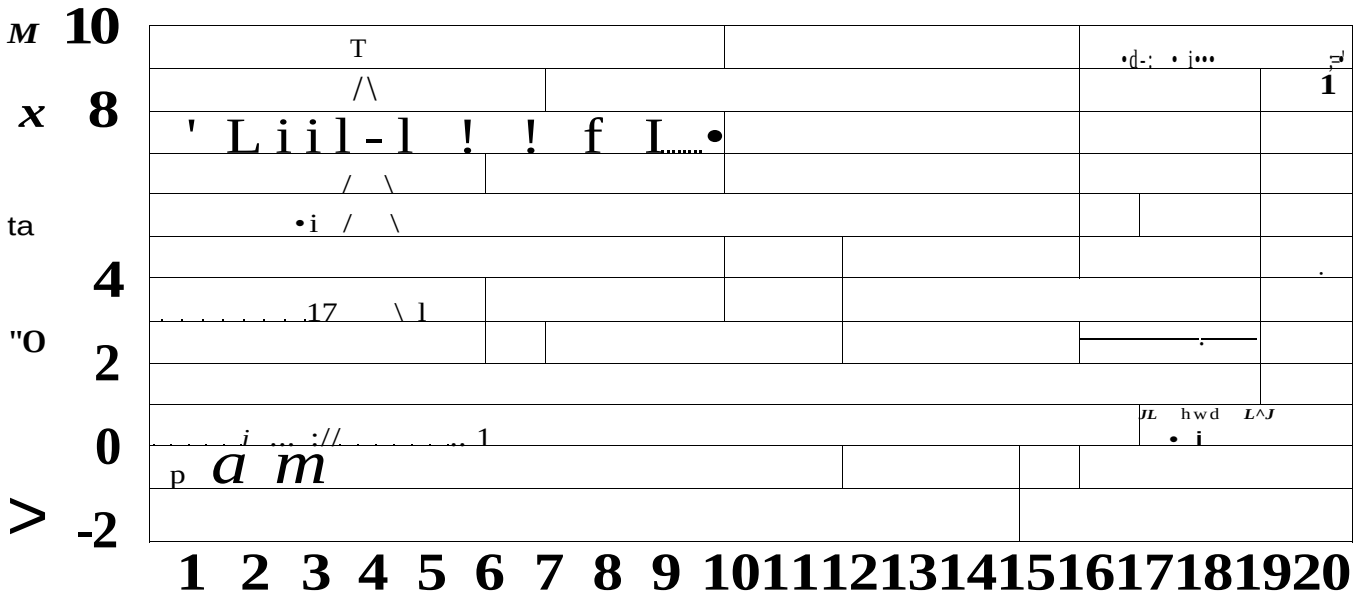
Rodada -->															
	-0.52892	-0.52892	-0.52892	2.170157	0.952325	0.952325	0.952325	0.952325	0.952325	0.952325	0.952325	0.952325	0.952325	0.952325	0.952325
	-0.47523	-0.47523	-0.47523		0.841753	0.841753	0.841753	0.841753	0.841753	0.841753	0.841753	0.841753	0.841753	0.841753	0.841753
fmax	-13.0477	-13.0477	-13.0477		-0.05038!	-0.05038	-0.05038	-0.05038	-0.05038	-0.05038	-0.05038	-0.05038	-0.05038	-0.05038	-0.05038
	13.04772	13.04772	13.04772	6.232908	0.050381	0.050381	0.050381	0.050381	0.050381	0.050381	0.050381	0.050381	0.050381	0.050381	0.050381
	0	0	2.699078	-1.21783	0	0	0	0	0	0	0	0	0	0	0
Delta x2Fk	0	0	10.47523	-9.15825	0	0	0	0	0	0	0	0	0	0	0
Delta f	0	0	-6.81481	-6.18253	0	0	0	0	0	0	0	0	0	0	0

^	V	x	Y	z	AA	AB	AC	AD	^m	AF	AG
8	ro	co	cn	o)		00	to	ro	ro	ro	ro
	0.952325	0.952325	1.060501	o	0.999108	0.999108	0.988643	0.994924	1.00033	1.004711	0.997661
cn	0.841753	0.841753	1.196406	1.196406	0.999446	0.999446	0.967489	0.985644	1.001097	1.013987	0.992897
cn	-0.05038	-0.05038	-0.00503	-0.00503	-0.00045	-0.00045	-0.00045	-0.00027	-8.9E-05	-3.2E-05	-6.6E-06
-4	0.050381	0.050381	0.00503	0.00503	0.00045	0.00045	0.00045	0.000267	8.9E-05	1.3E-06	6.5E-06
oo	o	0.108175	o	-0.06139	o	o	-0.01047	0.006281	0.005406	0.004381	-0.00705	-8.1E-07
to	o	0.354653	o	-0.19696	o	o	-0.03196	0.018155	0.015453	0.012889	-0.02109	-3.9E-05
o	o	-0.04535	o	-0.00458	o	o	-0.00018	-0.00018	-8.8E-05	2.5E-05	-2E-05	8.8E-07

^	AH	^	AJ	AK	AL	AM	AN
4*:	**0.99766**	**0.997657**	**0.997586**	**0.997618**	**0.9976**	**0.997578**	**0.997577**
0.992858	**0.992862**	**0.992677**	**0.992763**	**0.992762**	**0.992721**	**0.99272**	
-6.6E-06	**-6.6E-06**	**-6.6E-06**	**-6.6E-06**	**-6E-06**	**-6E-06**	**-6E-06**	
	7.1 E-06	**6.8E-06**	**6.8E-06**	**6.1 E-06**	**6E-06**	**6E-06**	
-7E-05	**-0.00018**	**-1.1E-05**	**-1.8E-05**	**-2.2E-05**	**-2.9E-07**	**-0.99758**	
-1E-06	**-4.1E-05**	**-5.4E-07**	**-0.99272**				
-2.5E-07	**-7.6E-07**	**-1E-07**	**-5.1E-10**	**-6E-06**			

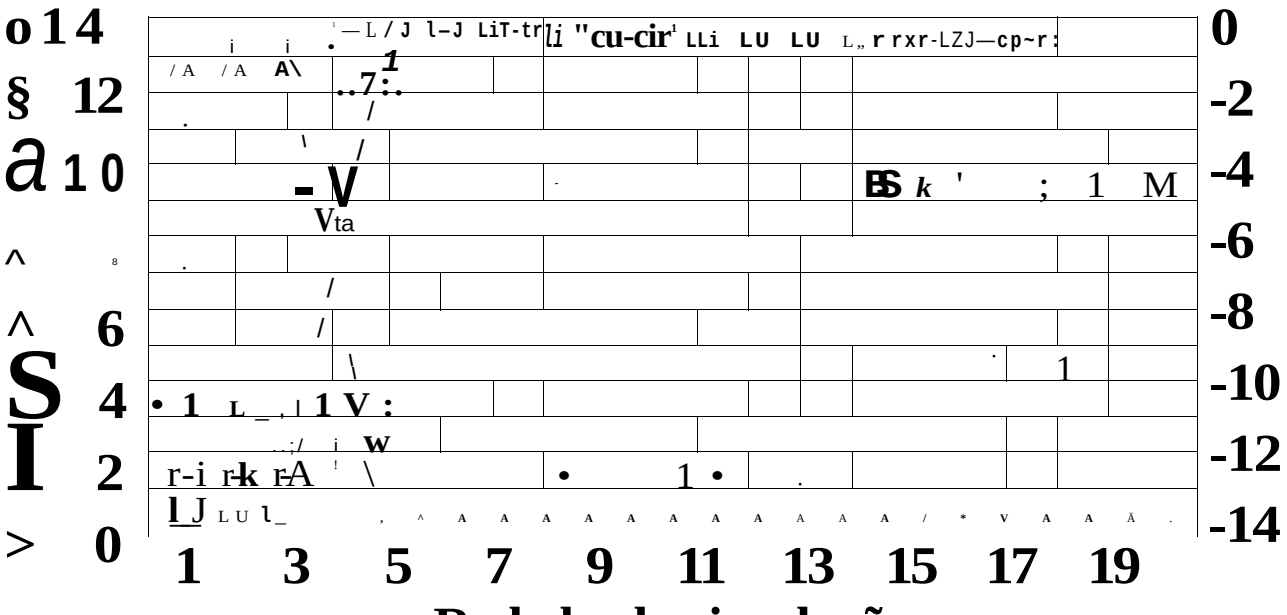
Função de teste 18

Comportamento de x1 Fk e x2Fk



Função de teste 18

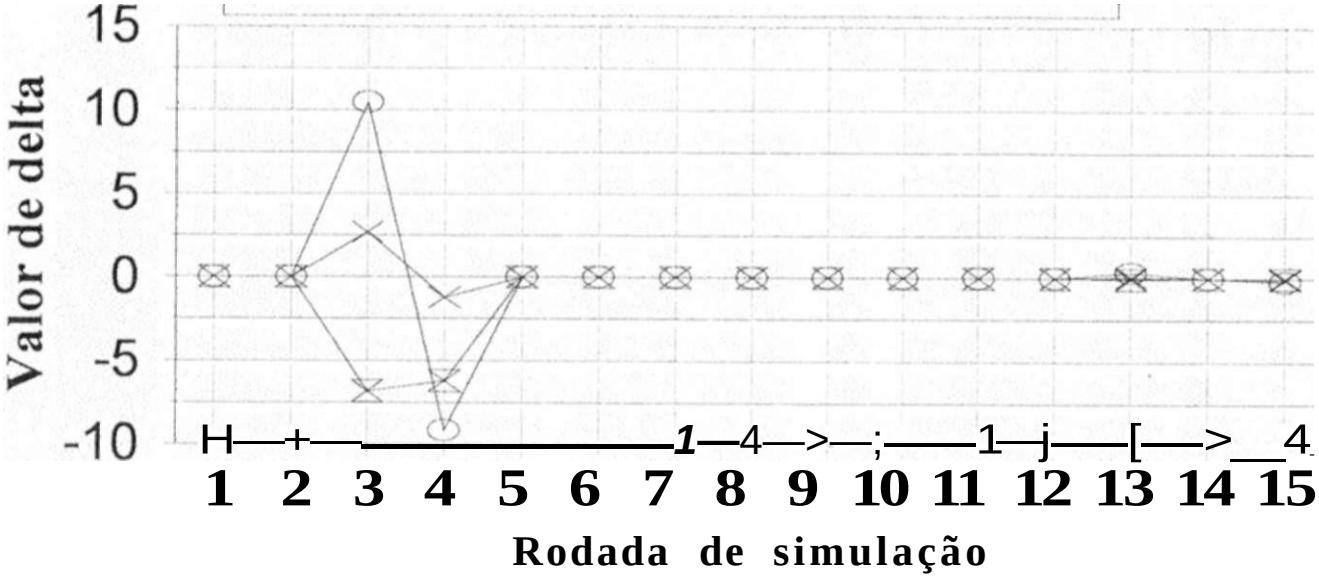
Comportamento de f(x1Fk,x2Fk) e fmax



ii **Valor de fmax** **Valor de f(x1Fk,x2Fk)**

Função de teste 18

Comportamento dos Deltas

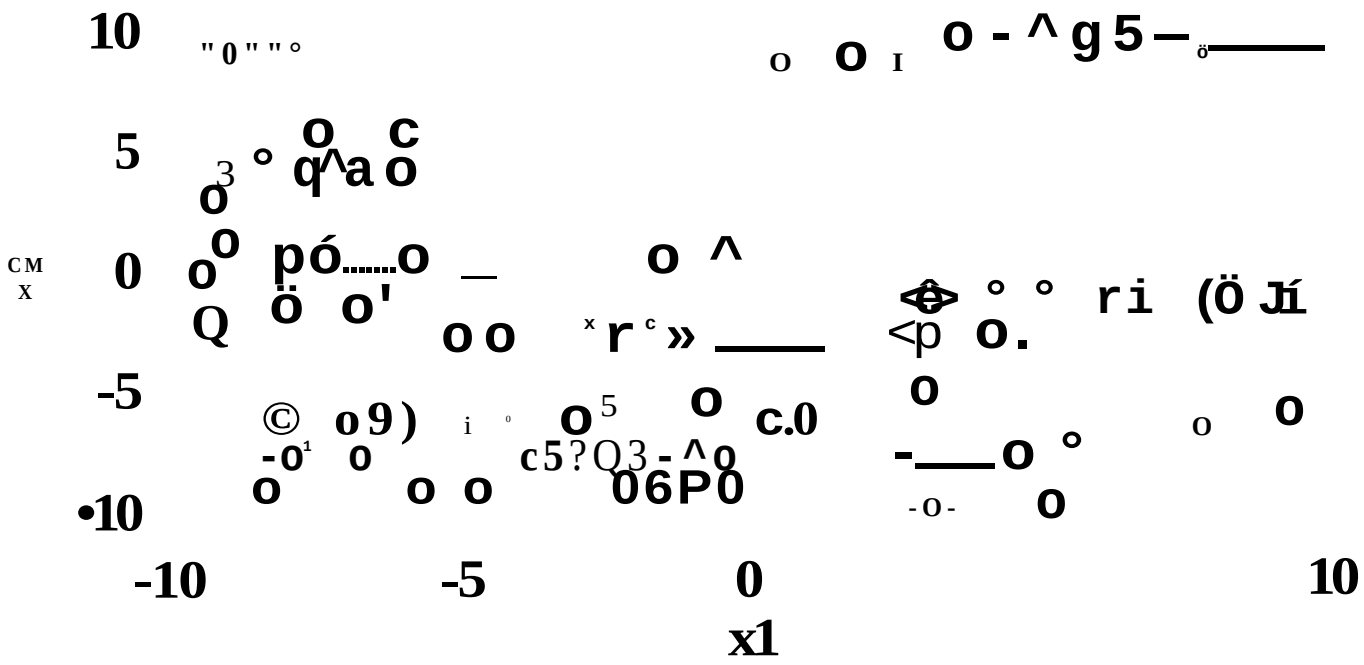


x **Delta x1Fk** e **Delta x2Fk**

Delta f(x1Fk,x2Fk)

Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação

Anexo U

Função de teste 19

Proposto em [Himmelblau] pac

Parâmetros

Do algoritmo

0.001082

Como vínculos quase violados provocam ineficiência na busca c pode acarretar zig-zag, procura-se uma direção viável que, ao mesmo tempo, reduza f{x) c penetre no interior dc V.

Vimos, então, vários métodos para resolução do problema clássico dc programação não linear, evidentemente que, aqui, não esgotamos todos os métodos existentes porém é nossa idéia apenas ilustrar as diversas abordagens para a solução desse problema para o qual iremos propor um novo algoritmo a partir do capítulo 7.

2.4 Conclusões

Iodos os métodos aqui apresentados, normalmente, se aplicam a uma classe dc funções "bem comportadas". Em geral exige-se características específicas para essas funções tais como: convexidade, difercnciabilidadc da função objetivo, c, às vezes das restrições, ou compacticidade do conjunto de pontos viáveis ou, o que é pior, uma combinação dessas hipóteses. Ocorre que, na medida cm que tais exigências são feitas, restringe-se consideravelmente, a classe dc funções cm que podemos aplica-los.

Um problema crítico nos métodos que usam derivadas é que, em torno do ponto de ótimo existirá, inerentemente ao processo ('^'^ o), uma instabilidade numérica muito acentuada, quando o método *dx* utiliza a matriz llcssiana então essa questão ganha complicadores maiores pois passamos a lidar com matrizes mal condicionadas que deverão ser invertidas e isso nem sempre será possível.

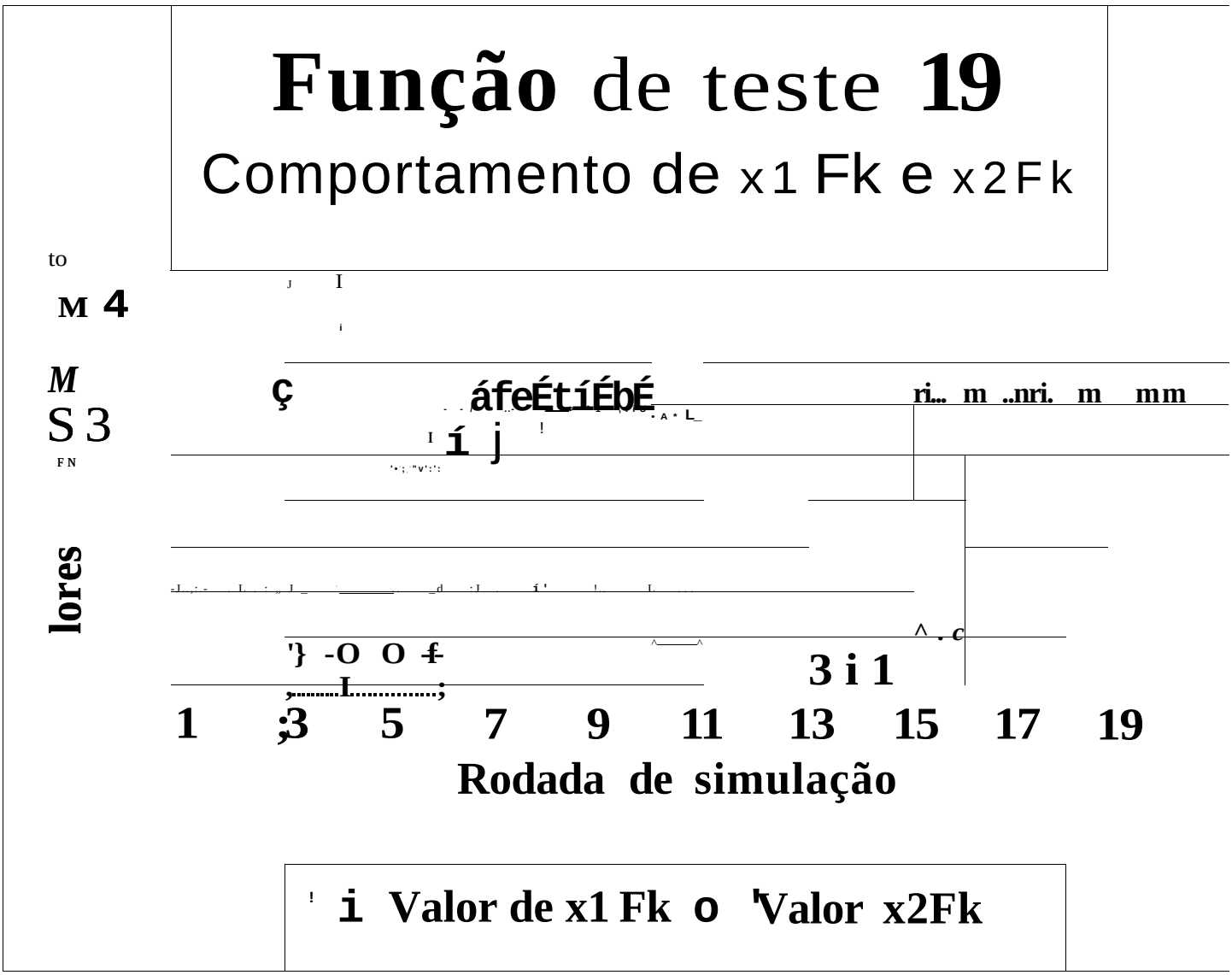
Devemos ressaltar inicialmente que a grande maioria dos algoritmos determinísticos existentes são dc natureza seqüencial, isto é, a determinação do valor do passo atual (iteração **d)** depende do passo anterior (iteração k-1); a utilização dc operações simultâneas dc cálculo nesses algoritmos são raramente aplicáveis (ausência dc paralelismo dc operações).

Cálculo 2 **1.324897 0.961326 4.513638 3.477033**

```
A:A22:      Iteração
A:B22:      1
A:A23:      x1
A:B23:       +$B$9+($B$10-$B$9)*@RAND
A:C23:       +$B$9+($B$10-$B$9)*@RAND
A:A24:      x2
A:B24:      +$B$11 +($B$12-$B$11)*@RAND
A:C24:       +$B$11+($B$12-$B$11)*@RAND
A:A25:      f(x1 ,x2)
A:B25:          -(1.5-B23*(1-B24))^2-(2.25-B23*(1-B24^2))^2-(2.625-B23*(1-B24^3))^2
A:C25:      -(1.5-C23*(1  -C24))^2-(2.25-C23*(1  -C24^2))^2-(2.625-C23*(1  -C24^3))^2
A:A26:       '(x1-x1fk)^3
A:B26:      (B23-$GY25)^3
A:C26:      (C23-$GY25)^3
A:A27:       '(x2-x2fk)^3
A:B27:      (B24-$GZ25)^3
A:C27:      (C24-$GZ25)^3
A:A28:      Fk(x1,x2)
A:B28:      @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28:      @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29:     Fk/SumFk
A:B29:       +B28/($D$14+$GU28)
A:C29:       +C28/($D$14+$GU28)
A.A30:     Curare
A:B30:      ($HA27-B25)^(1/($F$10+($HA27-B25)))
A:C30:      ($HA27-C25)^(1/($F$10+($HA27-C25)))
A:A31:     x1Fk-a1k-x1ik
A:B31:     +$GY29-B23
A:C31:     +$GY29-C23
A:A32:     x2Fk-a2k-x2ik
A:B32:      +$GZ29-B24
A:C32:     +$GZ29-C24
A:A33:     f(x1Fk-a1k,x2ik)
A:B33:     -(1.5-$GY29*(1  -B24))^2-(2.25-$GY29*(1  -B24^2))^2-(2.625-$GY29*(1  -B24^3))^2
A:C33:     -(1.5-$GY29*(1 -C24))^2-(2.25-$GY29*(1 -C24^2))^2-(2.625-$GY29*(1 -C24^3))^2
A:A34:     f(x1ik,x2Fk-a2k)
A:B34:     -(1.5-B23*(1  -$GZ29))^2-(2.25-B23*(1  -$GZ29^2))^2-(2.625-B23*(1  -$GZ29^3))^2
A:C34:     -(1.5-C23*(1  -$GZ29))^2-(2.25-C23*(1  -$GZ29^2))^2-(2.625-C23*(1  -$GZ29^3))^2
A:A35:     Sinal 1
A:B35:     (B33-B25)/($D$14+@ABS(B33-B25))
A:C35:     (C33-C25)/($D$14+@ABS(C33-C25))
A:A36:     Sinal 2
A:B36:      (B34-B25)/($D$14+@ABS(B34-B25))
A:C36:     (C34-C25)/($D$14+@ABS(C34-C25))
A:A37:     Passo
A:B37:      +$D$10*@EXP(-($E$10/$B22))
A:C37:    +$D$10*@EXP(-($E$10/$B22))
A:A38:     Cálculo 1
A:B38:       +B23+B37*B30*B35*B31
A:C38:     +C23+C37*C30*C35*C31
A:A39:     Cálculo 2
A:B39:       +B24+B37*B30*B36*B32
A:C39:     +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23:  @AVG(B23..GT23)
A:GW23:  @STD(B23..GT23)
A:GX23:  @MAX(B23..GT23)
```

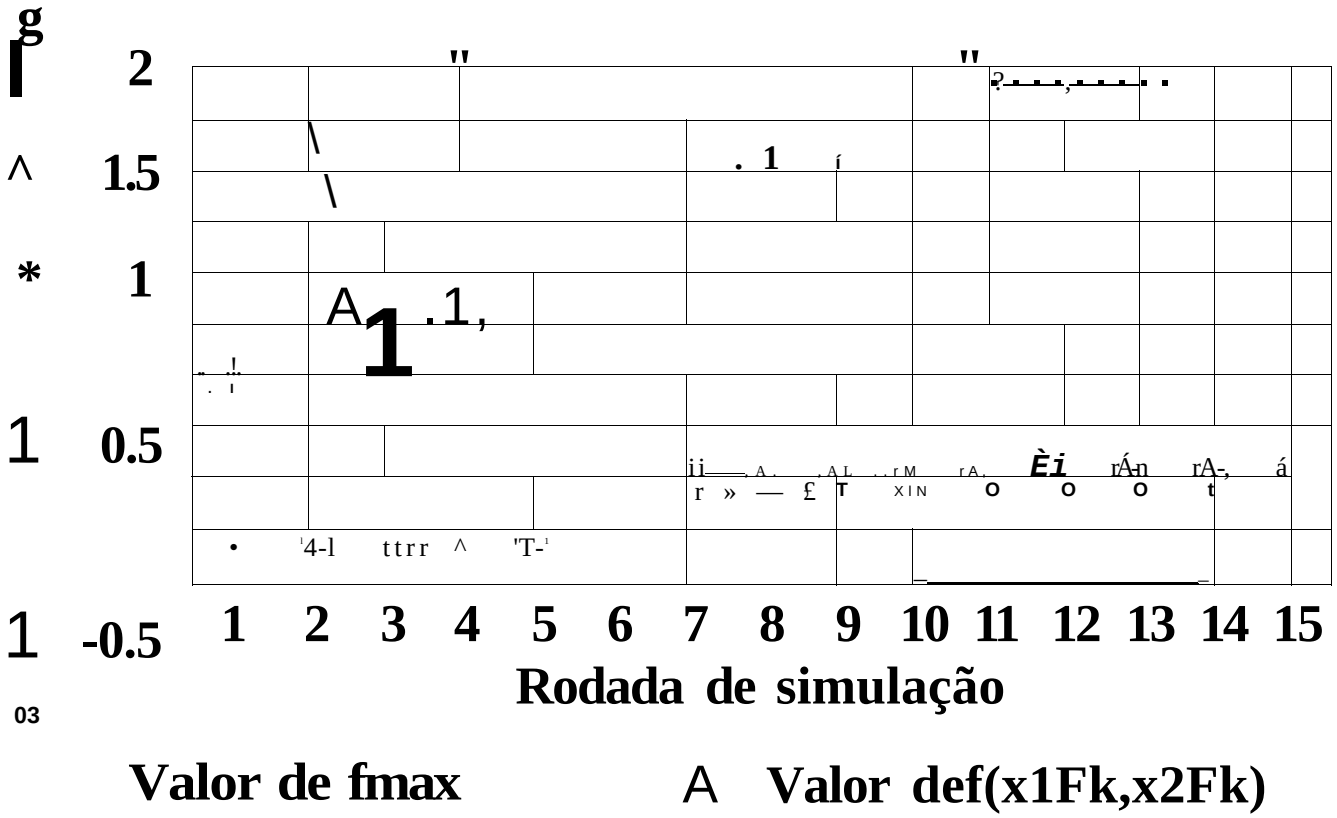
A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: (1.5-GY25*(1-GZ25)^2+(2.25-GY25*(1-GZ25^2))^2+(2.625-GY25*(1-GZ25^3))^2
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A.HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A:GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28. x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

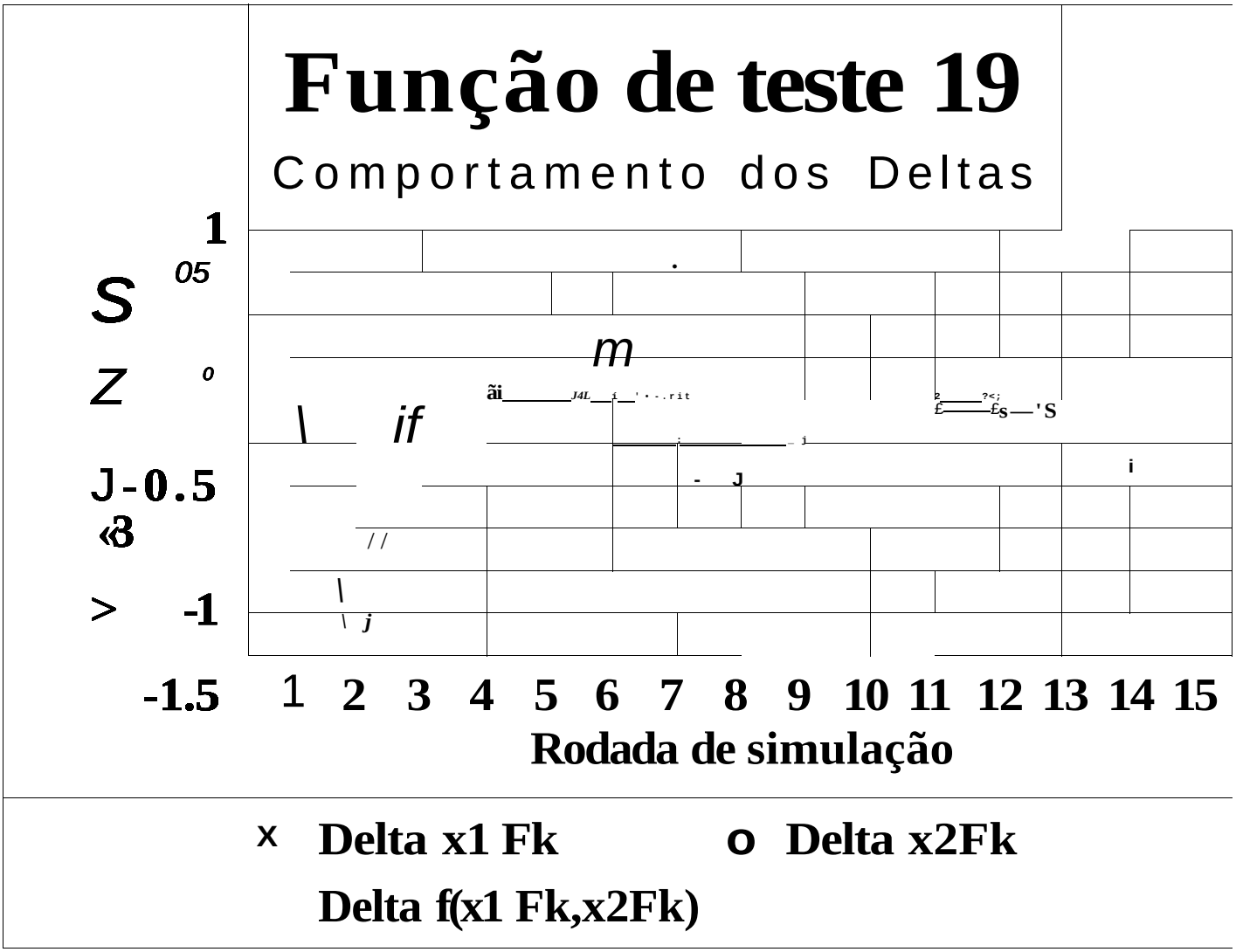
^	X	V	N	AA	AB	AC	AD	^m
co	ro	Œ	Œ	oi	¯*-o.	00	Œ	ºo
	3.093466	3.093466	3.093466	3.093466	3.0862	3.090303	3.083353	3.081986	3.086183
cn	0.514376	0.514376	0.514376	0.514376	0.521065	0.517288	0.523685	0.521324	0.520542
cn	-0.00288	-0.00288	-0.00288	-0.00288	· ooo¤8 ¤	-0.00166	-0.00144	-0.00133	-0.00113
-1	0.002882	0.002882	0.002882	0.002882	0.001087	o ·oo ðÐ7	0.001374	0.001056	0.001082
co	-5.2E-12	o	o	-0.00727	0.004103	-0.00695	-0.00137	0.004197	-3.08618
cn	1Ñ ro	o	o	0 006689	-0.00378	0.006398	-0.00236	-0.00078	-0.52054
¯o	-2E-13	o	o	-0.00179	o oooÐ¯.	-0.00028	-0.00032	2.6E-05	-0.00108


Função de teste 19
Comportamento de x1 Fk e x2Fk
lores
1 3 5 7 9 11 13 15 17 19
Rodada de simulação
i Valor de x1 Fk o Valor x2Fk

Função de teste **19**

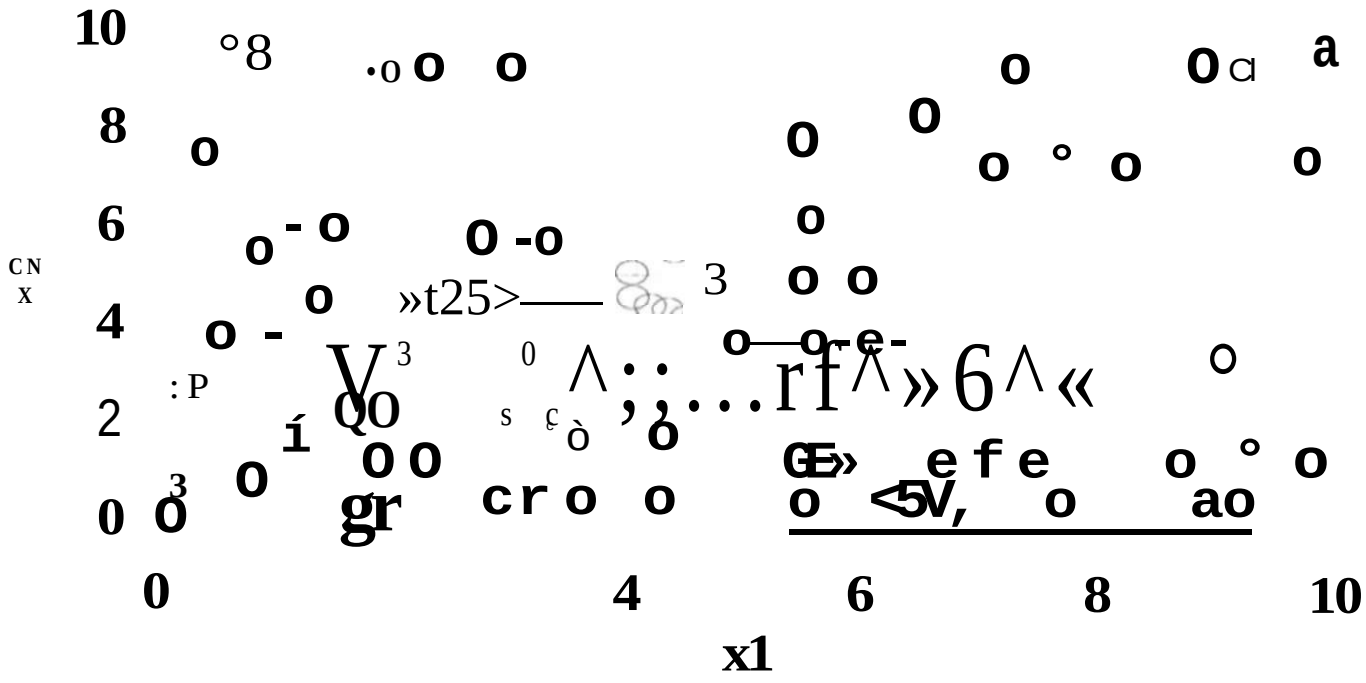
Comportamento de f(x1 Fk,x2Fk) e fmax




Função de teste 19
Comportamento dos Deltas
1
05
0
-0.5
-1
-1.5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Rodada de simulação
x Delta x1 Fk
o Delta x2Fk
Delta f(x1 Fk,x2Fk)

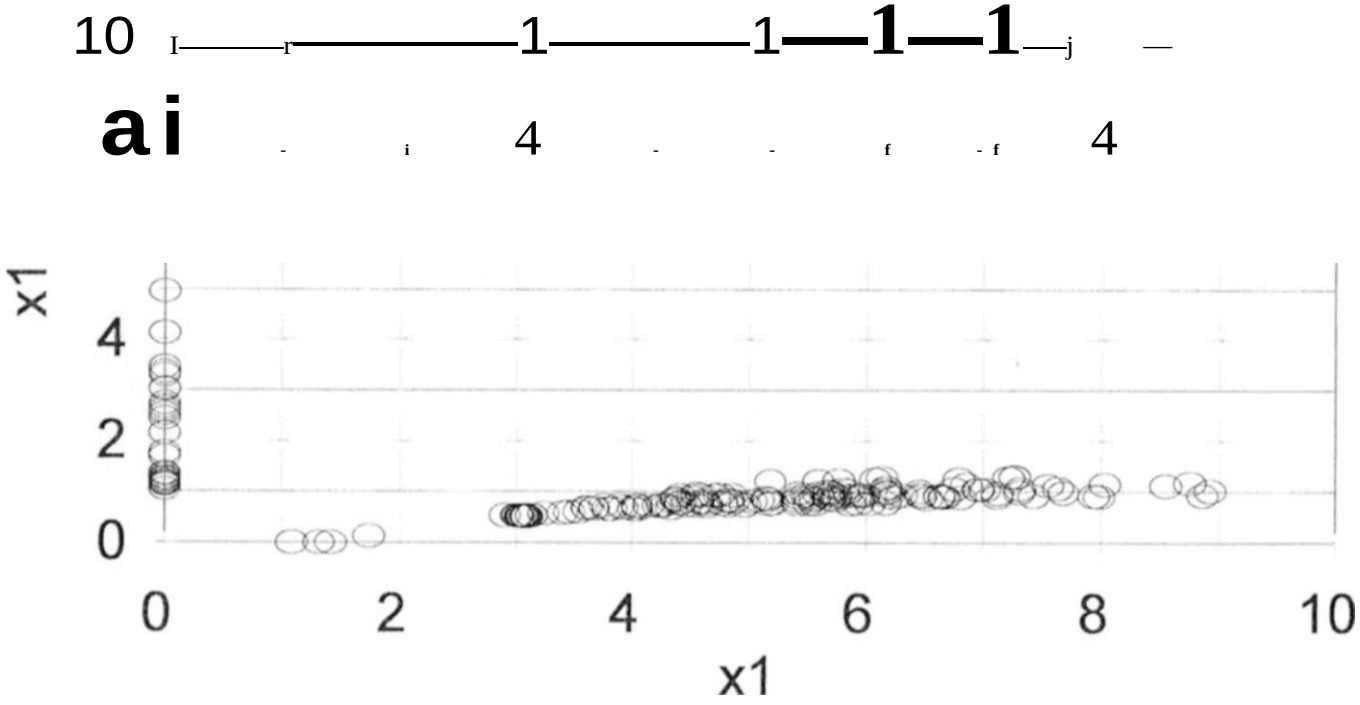
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação



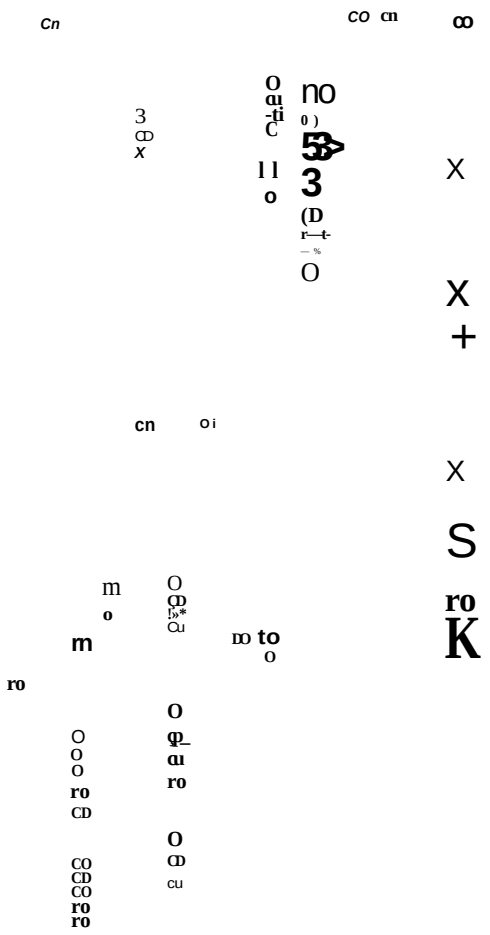
Alguns dos algoritmos apresentados anteriormente tem o inconveniente de zigzaguear em tono do ponto de ótimo, isso leva a uma lentidão de convergência indesejada. Quando tentamos corrigir esse problema fazemos crescer sua complexidade computacional. O aumento na dimensão do espaço produz um conseqüente crescimento do esforço computacional c conseqüente complicação na implementação do algoritmo.

Devemos salientar que, quando o algoritmo usa as informações de gradiente e matriz Hessiana introduz cálculos adicionais necessários a geração destes bem como de multiplicação c inversão de matrizes. O cálculo de derivadas introduz erros por aproximações numéricas que se propagam ao longo de todo o processo influenciando nas futuras iterações. Quando trabalhamos com métodos numéricos a inversão de matrizes torna-se critica pois matrizes mal condicionadas podem levar o algoritmo a divergir (erros de ***oveiflow),*** essa instabilidade numérica é um inconveniente que deve ser considerado quando da escolha do método a ser utilizado para implementação. Por exemplo, no método de Newton a equação 2.3.1.2.6 requer inversão da matriz Hessiana e aí devemos utilizar por precaução técnicas que garantam uma inversa positiva definida, nem sempre os códigos já implementados em algumas linguagens, compiladores ou pacotes aplicativos realizam essa tarefa satisfatoriamente.

O maior de todos os problemas no entanto nos parece ser o fato de todos esses algoritmos encontrarem apenas extremos locais (máximos ou mínimos). Para garantir extremos globais assentam-se sobre hipóteses altamente restritivas (por exemplo, convexidade da função objetivo c concavidade das restrições), essas hipóteses normalmente não são obedecidas na maioria dos casos práticos, tenta-se compensar essa deficiência rodando o algoritmo a partir de pontos iniciais distintos c comparar os resultados. Nesse ponto as seguintes perguntas devem ser respondidas: quantas vezes teremos que rodar o algoritmo? Isso garante que não percamos algum ponto candidato ao extremo procurado[9]

Por todos os motivos acima enumerados procuraremos, como já dito anteriormente, gerar um algoritmo que não apresente tantas desvantagens.

Anexo V



A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	4.664921	0.269681	0.99247	1.939454	4.461469	2.513715	1.499434	4.961094	Resultado da simulação		
24	x2	0.012116	0.991833	3.175218	2.958888	2.331325	2.558605	1.431377		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1 ,x2)	-86.6751	8.746068	6.439822	8.395166	-2238.77				1.197752	1.970439	-11.8167
26	(x1-x1fk)^3	41.67975	-0.79936	-0.00865	0.408025	0.10102				a1k	a2k	fmax
27	(x2-x2fk)^3	-7.51022	-0.93718	1.748728	0.965745	0.14432				0.10102	0.14432	11.8229
28	Fk(x1 ,x2)	0	6.7E-12	6.2E-35	1.3E-14	3.051907				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	2.2E-12	2E-35	4.4E-15					1.096732	1.826119	
30	Curare	1.023393	1.010963	1.016101	1.011982							
31	x1Fk-a1k-x1ik	-3.56819	0.827051	0.104262	-0.84272							
32	x2Fk-a2k-x2ik	1.814003	0.834286	-1.3491	-1.13277							
33	f(x1Fk-a1k,x2ik)	-4.61332	7.506881	6.892743	8.655727							
34	f(x1ik,x2Fk-a2k)	-44.3966	10.25355	11.88407	7.695358							
35	Sinal 1	1	-1	1	1							
36	Sinal 2	1	1	1	-1							
37	Passo	0.006738	0.006738	0.006738	0.006738							
38	Cálculo 1	4.640317	0.264048	0.993184	1.933707							
39	Cálculo 2	0.024625	0.997516	3.165982	2.966612							
40												
41	Iteração	2										
42	x1	4.640317	0.264048	0.993184	1.933707	4.466916	2.504945	1.489662	4.934366	Resultado da simulação		
43	x2	0.024625	0.997516	3.165982	2.966612	2.340346	2.561441	1.428337		x1Fk	x2Fk	f(x1Fk.x2F
44	f(x1 ,x2)	-85.38	8.764721	6.529969	8.385538	-2181.7				1.174117	1.965646	-11.8501
45	(x1-x1fk)^3	41.6448	-0.75374	-0.00592	0.438267	0.070606				a1k	a2k	fmax
46	(x2-x2fk)^3	-7.31292	-0.90741	1.72945	1.0029	0.131019				0.070606	0.131019	11.8229
47	Fk(x1 ,x2)	0	9.7E-31	1.3E-90	1.2E-38	2.567814				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	3.8E-31	4.9E-91	4.7E-39					1.103511	1.834627	
49	Curare	1.02348	1.010906	1.015952	1.012008							
50	x1Fk-a1k-x1ik	-3.53681	0.839464	0.110328	-0.8302							
51	x2Fk-a2k-x2ik	1.810002	0.837111	-1.33135	-1.13199							
52	f(x1Fk-a1k,x2ik)	-4.46918	7.522171	7.001856	8.620007							
53	f(x1ik,x2Fk-a2k)	-43.5251	10.21093	11.89493	7.785728							
54	Sinal 1	1	-1	1	1							
55	Sinal 2	1	1	1	-1							
56	Passo	0.082085	0.082085	0.082085	0.082085							
57	Cálculo 1	4.343181	0.194389	1.002384	1.864742							
58	Cálculo 2	0.176687	1.06698	3.054954	3.060647							

A:A22: Iteração
A:B22: 1
A:A23: x1
A:B23: +$B$9+($B$10-$B$9)*@RAND
A:C23: +$B$9+($B$10-$B$9)*@RAND
A:A24: x2
A:B24: +$B$11 +($B$12-$B$11)*@RAND
A:C24: +$B$11+($B$12-$B$11)*@RAND
A:A25: f(x1 ,x2)
A:B25: -(4*B23^2+4*B24^2-4*B23*B24-12*B24)
A:C25: -(4*C23^2+4*C24^2-4*C23*C24-12*C24)
A:A26: '(x1-x1fk)^3
A:B26: (B23-$GY25)^3
A:C26: (C23-$GY25)^3
A:A27: '(x2-x2fk)^3
A:B27: (B24-$GZ25)^3
A:C27: (C24-$GZ25)^3
A:A28: Fk(x1 ,x2)
A:B28: @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28: @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29: Fk/SumFk
A:B29: +B28/($D$14+$GU28)
A:C29: +C28/($D$14+$GU28)
A:A30: Curare
A:B30: ($HA27-B25)^(1 /($F$10+($HA27-B25)))
A:C30: ($HA27-C25)^(1/($F$10+($HA27-C25)))
A:A31: x1Fk-a1k-x1ik
A:B31: +$GY29-B23
A:C31: +$GY29-C23
A:A32: x2Fk-a2k-x2ik
A:B32. +$GZ29-B24
A:C32: +$GZ29-C24
A:A33: f(x1Fk-a1k,x2ik)
A:B33: -(4*$GY29^2+4*B24^2-4*$GY29*B24-12*B24)
A:C33: -(4*$G Y29^2+4*C24^2-4*$GY29*C24-12*C24)
A:A34: f(x1ik,x2Fk-a2k)
A:B34: -(4*B23^2+4*$GZ29^2-4*B23*$GZ29-12*$GZ29)
A:C34: -(4*C23^2+4*$GZ29^2-4*C23*$GZ29-12*$GZ29)
A:A35: Sinal 1
A:B35: (B33-B25)/($D$14+@ABS(B33-B25))
A:C35: (C33-C25)/($D$14+@ABS(C33-C25))
A:A36: Sinal 2
A:B36: (B34-B25)/($D$14+@ABS(B34-B25))
A:C36: (C34-C25)/($D$14+@ABS(C34-C25))
A:A37: Passo
A:B37: +$D$10*@EXP(-($E$10/$B22))
A:C37: +$D$10*@EXP(-($E$10/$B22))
A:A38: Cálculo 1
A:B38: +B23+B37*B30*B35*B31
A:C38: +C23+C37*C30*C35*C31
A:A39: Cálculo 2
A:B39: +B24+B37*B30*B36*B32
A:C39: +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23: @AVG(B23..GT23)
A:GW23: @STD(B23..GT23)
A:GX23: @MAX(B23..GT23)

A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A.GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: +4*GY25^2+4*GZ25^2-4*GY25*GZ25-12*GZ25
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A.HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27 .GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A.GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

Rodada -->	**1.197752**	**1.174117**	**1.164385**	**1.143564**	**0.946608**	**0.893384**	**0.980203**	**0.956812**	**1.005601**	**0.999524**	**1.000405**
fmax	**1.970439**	**1.965646**	**1.998402**	**2.009494**	**1.965856**	**1.991203**	**1.965661**	**1.975133**	**2.005121**	**1.98891**	**1.98607**
Delta x1Fk	**11.8229**	**11.8229**	**11.8229**	**11.8229**	**11.8229**	**11.96907**	**11.97816**	**11.99888**	**11.99888**	**11.99976**	**11.99995**
Delta x2Fk	**-11.8167**	**-11.8501**	**-11.8908**	**-11.9226**	**-11.9912**	**-11.958**	**-11.9964**	**-11.9944**	**-11.9999**	**-11.9995**	**-11.9992**
Delta f	**-0.02364**	**-0.00973**	**-0.02082**	**-0.19696**	**-0.05322**	**0.086819**	**-0.02339**	**0.048789**	**-0.00608**	**0.000887**	**0.00197**
	-0.00479	**0.032756**	**0.011091**	**-0.04364**	**0.025347**	**0.009472**	**0.029988**	**-0.01621**	**-0.00284**	**0.009468**	
	-0.03339	**-0.04076**	**-0.0318**	**-0.06858**	**0.033251**	**-0.03846**	**0.002073**	**-0.00552**	**0.000356**	**0.000328**	**-0.00065**

	1.002374	1.006622	1.002028	1.001958	1.001608	1.000473	0.998654	0.999835	1.000129
	1.995537	1.99935	1.998576	1.999197	1.999531	1.99825	1.996778	1.998188	1.999221
	11.99995								
	-11.9999	-11.9998							
	0.004248	-0.00459		-0.00035	-0.00113	-0.00182	0.001181	0.000294	-1.00013
	0.003812	-0.00077	0.000621	0.000333	-0.00128	-0.00147	0.00141	0.001034	-1.99922
	5E-05	-0.00016	-1.2E-05		2.2E-06		-1.9E-05		

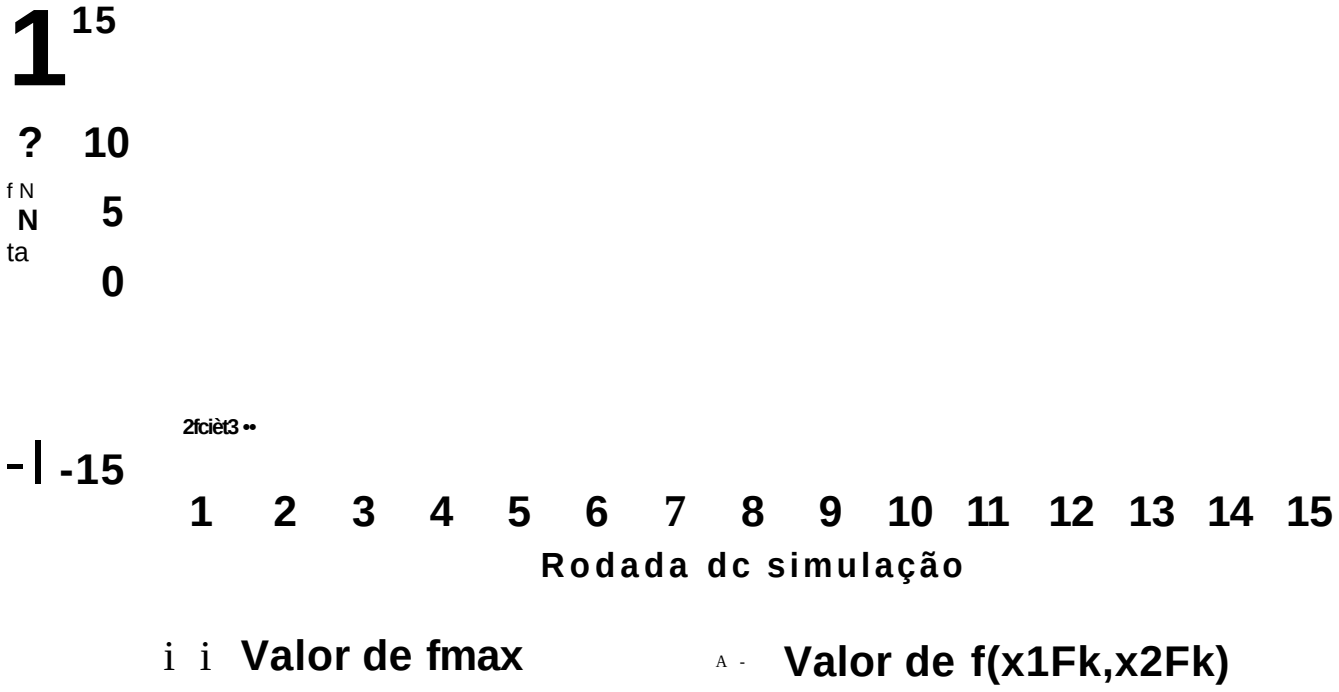
Função de teste 20

Comportamento de x1 Fk e x2Fk

• i **Valor de x1Fk** **Valor x2Fk**

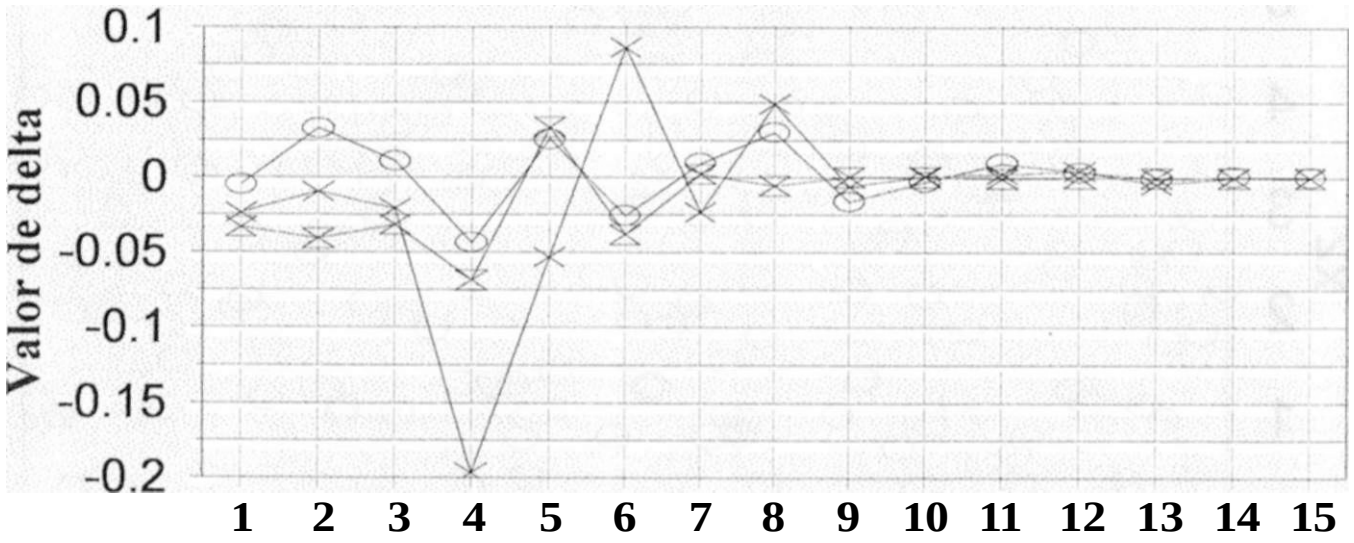
Função de teste **20**

Comportamento de f(x1F,x2F) e fmax



Função de teste **20**

Comportamento dos Deltas



x **Delta x1 r i ?** ª ª ª ª ᵠᵉ ^ **f » X 2 F k**

> **Delta f(x1 Fk,x2Fk)**

Capítulo 3

Processamento paralelo

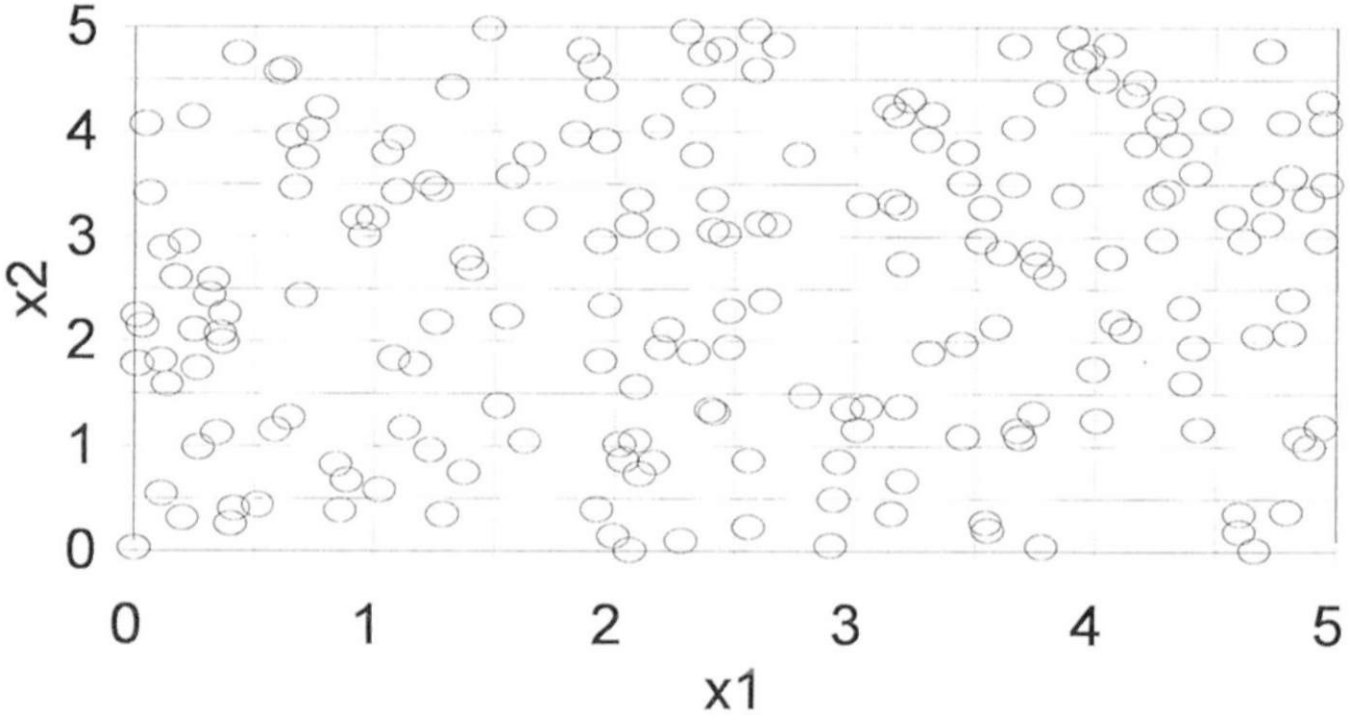
The human brain has about 10" neurons, each capable of switching no more than a thousand limes a second, tie nee the brain can perform about 10^{14} switches per second. In contrast, a digital computer may have as many as Itf transistors, each capable of switching as often as Iff limes per second. The total number of switches per second is as high as l(r . If the number of switches is proportional to computational power, the modern computer should be 10,000 limes more powerful than the brain. In reality, the computer is far behind the brain.

The classical computer is a single-processor system, while the brain has a very large number of processing units that operate at the same time. Modern semiconductor technology has reached the level at which building a computer with thousands of processors is possible. These systems are called massively parallel computers."

Neural and massively parallel computers
Hranko Soucek & Marina Soucek

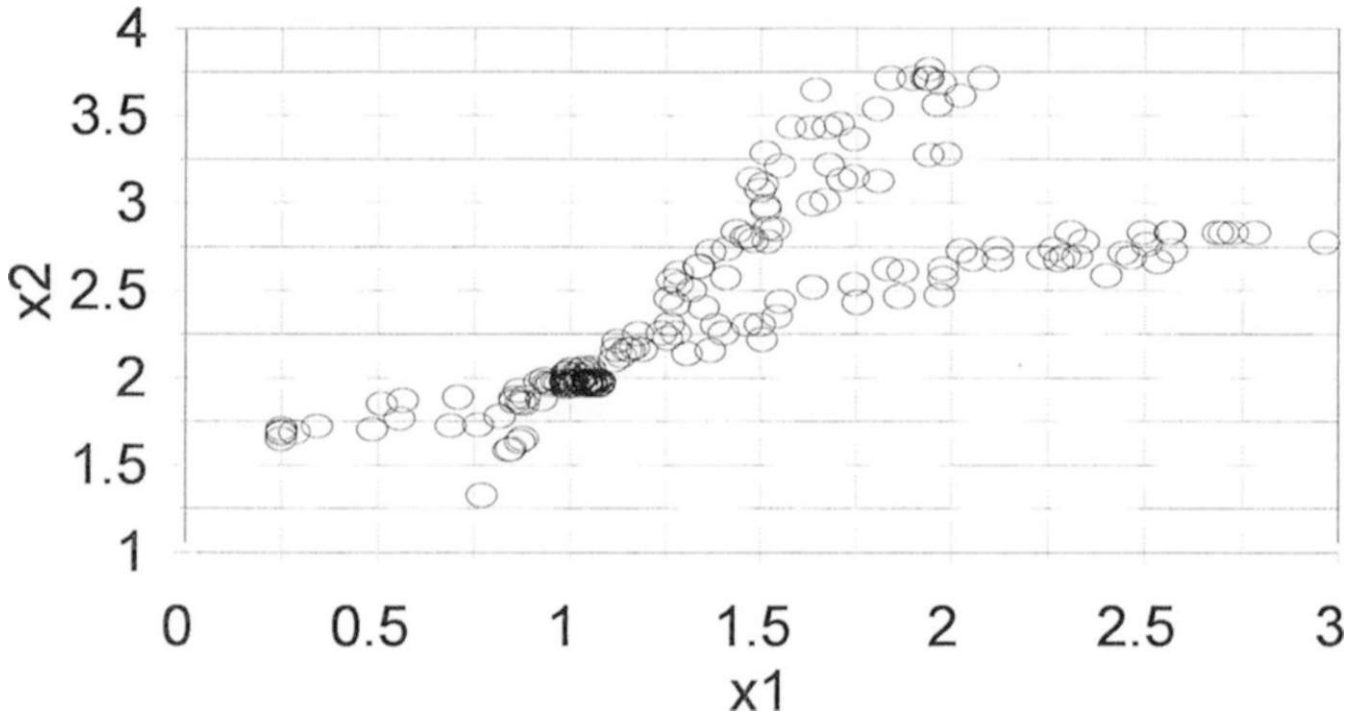
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação



Anexo X

Problema de programação linear
Parâmetros
Da função
Do algoritmo
xlmax
3
Gama 1 Gama 2 Gama 3
Delta 1 Delta 2 Delta 3
Erro
1E-15 4.004596 4.995187
32.79898

A	A	B	C	D	GT	GU	GV	GW	GX	GY	GZ	HA
22	Iteração	1										
23	x1	0.300659	6.997128	0.598466	0.524719	9.804845	5.611257	2.88C398	9.955559	Resultado da simulação		
24	x2	0.056273	6.958449	9.682016	9.254261	9.652791	4.762617	2.828551		x1Fk	x2Fk	f(x1Fk,x2F
25	f(x1 ,x2)	0.882683	-1.1E+46	-3.6E+65	-6.5E+57	-1E+105				4.250966	4.823552	32.61969
26	(x1-x1fk)^3	-61.6442	20.70992	-48.7271	-51.7386	-0.39821				a1k	a2k	fmax
27	(x2-x2fk)^3	-108.346	9.7304	114.6825	86.98007	-0.09304				-0.39821	-0.09304	32.44614
28	Fk(x1,x2)	0	0	0	0	1.062592				x1Fk-a1k	x2Fk-a2k	
29	Fk/SumFk	0	0	0	0					4.649175	4.916594	
30	Curare	1.026586	1	1	1							
31	x1Fk-a1k-x1ik	4.348516	-2.34795	4.05071	4.124456							
32	x2Fk-a2k-x2ik	4.860321	-2.04185	-4.76542	-4.33767							
33	f(x1Fk-a1k,x2ik)	9.579715	-6.9E+35	-1.4E+83	-5.3E+75							
34	f(x1ik.x2Fk-a2k)	25.18429	-2.1E+20	25.7799	25.63241							
35	Sinal 1	1	1	-1	-1							
36	Sinal 2	1	1	1	1							
37	Passo	0.003369	0.003369	0.003369	0.003369							
38	Cálculo 1	0.315699	6.989218	0.584819	0.510824							
39	Cálculo 2	0.073083	6.95157	9.665961	9.239648							
40												
41	Iteração	2										
42	x1	0.315699	6.989218	0.584819	0.510824	9.787489	5.60637	2.87405	9.937682	Resultado da simulação		
43	x2	0.073083	6.95157	9.665961	9.239648	9.636799	4.758275	2.82428		x1Fk	x2Fk	f(x1Fk.x2F
44	f(x1 ,x2)	0.99681	-7.7E+45	-1.7E+65	-3.1E+57	-6E+104				4.248808	4.837316	32.6842
45	(x1-x1fk)^3	-60.8426	20.58006	-49.1884	-52.2291	-0.00593				a1k	a2k	fmax
46	(x2-x2fk)^3	-108.138	9.450856	112.5838	85.31947	-0.02424				-0.00593	-0.02424	32.44614
47	Fk(x1 ,x2)	0	0	0	0	1.00035				x1Fk-a1k	x2Fk-a2k	
48	Fk/SumFk	0	0	0	0					4.254734	4.861559	
49	Curare	1.026581	1	1	1							
50	x1Fk-a1k-x1ik	3.939035	-2.73448	3.669915	3.74391							
51	x2Fk-a2k-x2ik	4.788476	-2.09001	-4.8044	-4.37809							
52	f(x1Fk-a1k,x2ik)	8.874881	-1E+34	-1.5E+81	-5.7E+73							
53	f(x1 ik,x2Fk-a2k)	24.93919	-9.4E+19	25.47743	25.32944							
54	Sinal 1	1	1	-1	-1							
55	Sinal 2	1	1	1	1							
56	Passo	0.041042	0.041042	0.041042	0.041042							
57	Cálculo 1	0.481664	6.876988	0.434196	0.357165							
58	Cálculo 2	0.274838	6.865791	9.468777	9.05996							

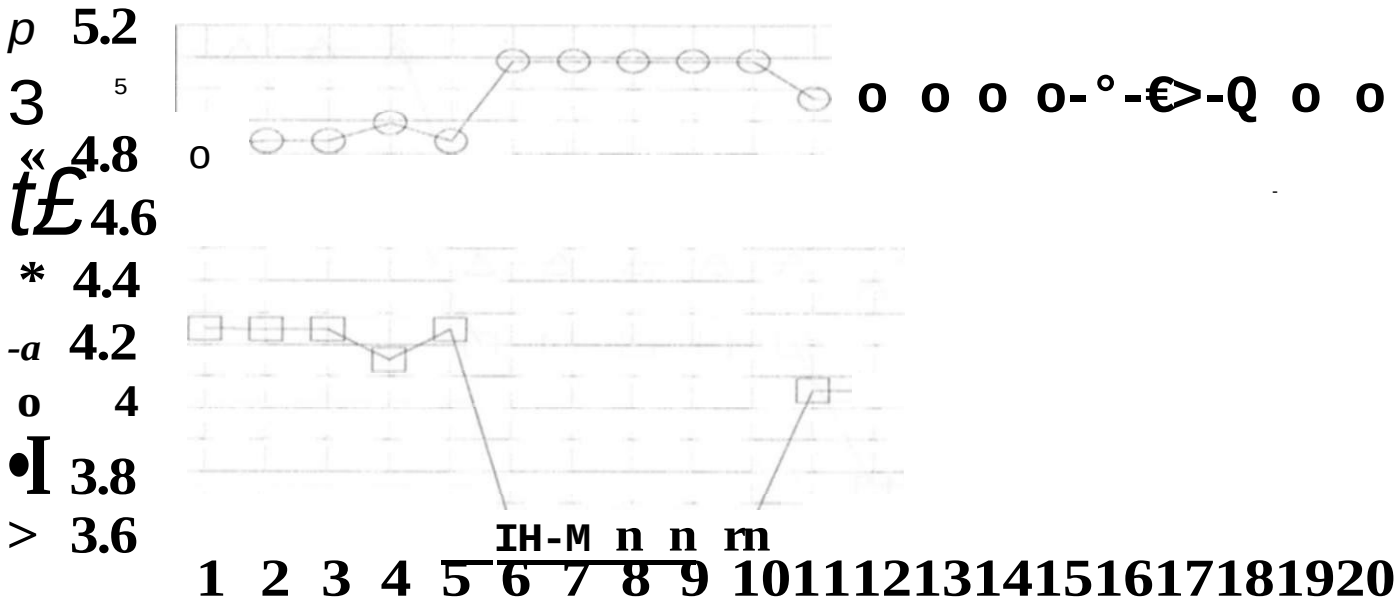
A:A22: Iteração
A:B22: 1
A:A23: x1
A:B23: +$B$9+($B$10-$B$9)*@RAND
A:C23: +$B$9+($B$10-$B$9)*@RAND
A:A24: x2
A:B24: +$B$11+($B$12-$B$11)*@RAND
A:C24: +$B$11 +($B$12-$B$11)*@RAND
A:A25: f(x1,x2)
A:B25: +2*B23+5*B24-(1 /$E$16)*(@EXP($E$16*(B23+4*B24-24))+@EXP($E$16*(3*B23+B24-21))+@EXP($E$16*(B23+B24-9)))
A:C25: +2*C23+5*C24-(1 /$E$16)*(@EXP($E$16*(C23+4*C24-24))+@EXP($E$16*(3*C23+C24-21))+@EXP($E$16*(C23+C24-9)))
A:A26: '(x1-x1fk)^3
A:B26: (B23-$GY25)^3
A:C26: (C23-$GY25)^3
A:A27: '(*2-x2fk)^3
A:B27: (B24-$GZ25)^3
A:C27: (C24-$GZ25)^3
A:A28: Fk(x1,x2)
A:B28: @EXP(-$D$12*@EXP($F$12*$B22)*(B25-$HA27)^(2*$E$12))
A:C28: @EXP(-$D$12*@EXP($F$12*$B22)*(C25-$HA27)^(2*$E$12))
A:A29: Fk/SumFk
A:B29: +B28/($D$14+$GU28)
A:C29: +C28/($D$14+$GU28)
A:A30: Curare
A:B30: ($HA27-B25)^(1 /($F$10+($HA27-B25)))
A:C30: ($HA27-C25)^(1 /($F$10+($HA27-C25)))
A:A31: x1Fk-a1k-x1ik
A:B31: +$GY29-B23
A:C31: +$GY29-C23
A:A32: x2Fk-a2k-x2ik
A:B32: +$GZ29-B24
A:C32: +$GZ29-C24
A:A33: f(x1Fk-a1k,x2ik)
A:B33: +2*$GY29+5*B24-(1/$E$16)*(@EXP($ES16*($GY29+4*B24-24))+@EXP($E$16*(3*$GY29+B24-21))+@EXP($E$16*($GY29+B24-9)))
A:C33: +2*$GY29+5*C24-(1/$E$16)*(@EXP($E$16*($GY29+4*C24-24))+@EXP($E$16*(3*$GY29+C24-21))+@EXP($E$16*($GY29+C24-9)))
A:A34: f(x1ik,x2Fk-a2k)
A:B34: +2*B23+5*$GZ29-(1 /$E$16)*(@EXP($E$16*(B23+4*$GZ29-24))+@EXP($E$16*(3*B23+$GZ29-21))+@EXP($E$16*(B23+$GZ29-9)))
A:C34: +2*C23+5*$GZ29-(1/$E$16)*(@EXP($E$16*(C23+4*$GZ29-24))+@EXP($E$16*(3*C23+$GZ29-21))+@EXP($E$16*(C23+$GZ29-9)))
A:A35: Sinal 1
A:B35: (B33-B25)/($D$14+@ABS(B33-B25))
A:C35: (C33-C25)/($D$14+@ABS(C33-C25))
A:A36: Sinal 2
A:B36: (B34-B25)/($D$14+@ABS(B34-B25))
A:C36: (C34-C25)/($D$14+@ABS(C34-C25))
A:A37: Passo
A:B37: +$D$10*@EXP(-($E$10/$B22))
A:C37: +$D$10*@EXP(-($E$10/$B22))
A.A38: Cálculo 1
A:B38: +B23+B37*B30*B35*B31
A.C38: +C23+C37*C30*C35*C31
A:A39: Cálculo 2

A:B39: +B24+B37*B30*B36*B32
A:C39: +C24+C37*C30*C36*C32
A:GU23: @SUMPRODUCT(B23..GT23,B25..GT25)/$GU25
A:GV23: @AVG(B23..GT23)
A:GW23: @STD(B23..GT23)
A:GX23: @MAX(B23..GT23)
A:GY23: Resultado da simulação
A:GU24: @SUMPRODUCT(B24..GT24,B25..GT25)/$GU25
A:GV24: @AVG(B24..GT24)
A:GW24: @STD(B24..GT24)
A:GY24: x1Fk
A:GZ24: x2Fk
A:HA24: f(x1Fk,x2Fk)
A:GU25: @SUM(B25..GT25)
A:GY25: @SUMPRODUCT(B23..GT23,B29..GT29)
A:GZ25: @SUMPRODUCT(B24..GT24,B29..GT29)
A:HA25: +2*GY25+5*GZ25
A:GU26: (@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B26..GT26, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B26..GT26,B28..GT28)/$GU28)^(1/3)
A:GY26: a1k
A:GZ26: a2k
A:HA26: fmax
A:GU27: (@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)/($D$14+@ABS(@SUMPRODUCT(B27..GT27, B28..GT28)/$GU28))*@ABS(@SUMPRODUCT(B27..GT27,B28..GT28)/$GU28)^(1/3)
A:GY27: +GU26
A:GZ27: +GU27
A:HA27: @MAX(B25..GT25)
A.GU28: @SUM(B28..GT28)
A:GY28: x1Fk-a1k
A:GZ28: x2Fk-a2k
A:GY29: +GY25-GY27
A:GZ29: +GZ25-GZ27

V	X	Y	Z	AA	AB	AC	AD
4.056819	**4.056819**	**4.056819**	**4.056819**	**3.98072**	**4.035241**	**4.009033**	**4.009033**	**4.004596**
4.970704	**4.970704**	**4.970704**	**4.970704**	**5.015993**	**4.986491**	**5.001035**	**5.001035**	**4.995187**
32.78079	**32.78079**	**32.78079**	**32.78079**	**32.78829**	**32.7958**	**32.79857**	**32.79857**	**32.79898**
32.96716	**32.96716**	**32.96716**	**32.96716**	**33.0414**	**33.00294**	**33.02324**	**33.02324**	**32.98513**
	0	**-0.0761**	**0.05452**	**-0.02621**		**-0.00444**	**-4.0046**
	0	**0.045288**	**-0.0295**	**0.014544**	**0**	**-0.00585**	**-4.99519**
0	**0**	**0.074245**	**-0.03847**	**0.020302**	**0**	**-0.03811**	**-32.9851**

Problema de prog. linear

Comportamento de x1Fk e x2Fk



CAPÍTULO 3

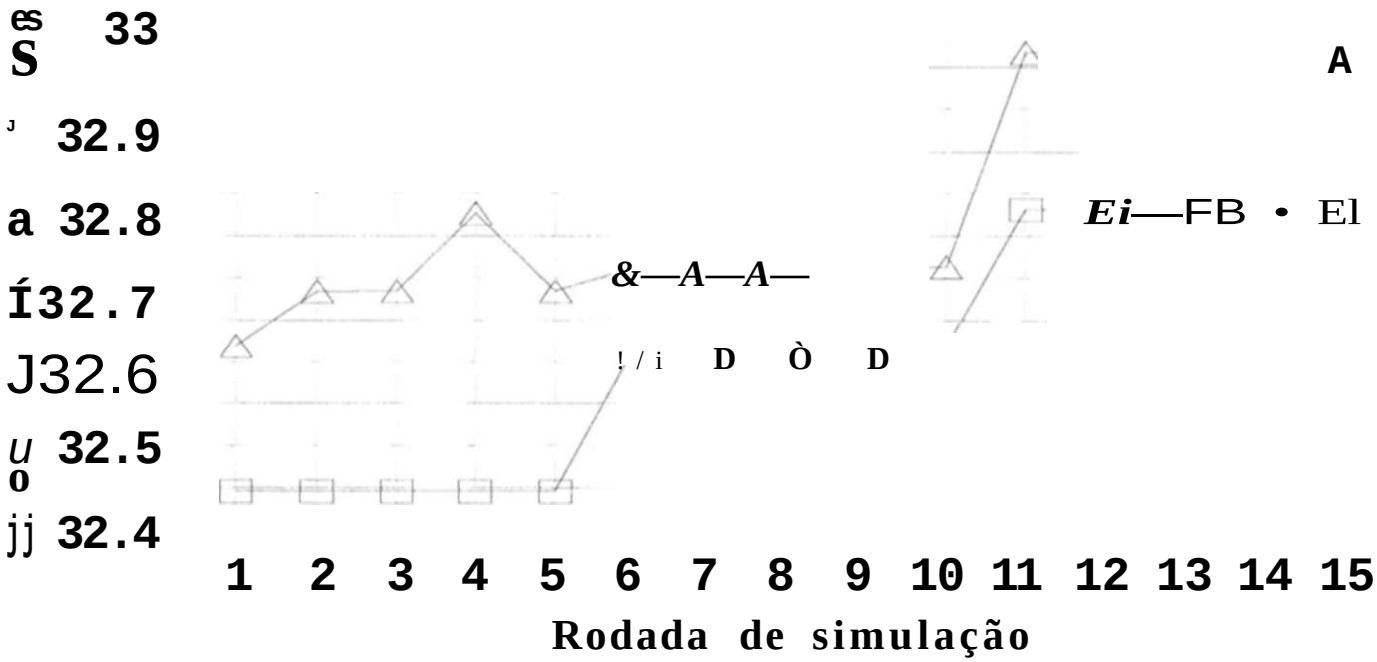
Processamento paralelo

3.1 Introdução

As máquinas de calcular existem desde que o homem começou a perseguir a idéia de contar. No inicio essas máquinas eram muito simples c compostas de pequenos seixos ou ossos de animais ou aves aonde se fazia uma correspondência um-a-um entre o que se queria contar e essas pedras ou ossos. Os romanos 2000 anos atrás já utilizavam o ábaco, onde os números eram representados por pedrinhas de calcário denominadas ***ca/cu/i*** (daí a palavra cálculo) representando as quantidades de forma mais abstrata c sofisticada. Km 1614 a invenção dos logaritmos por Népcr tornou possível a construção das réguas de cálculo (baseada no fato de que os logaritmos permitem transformar as multiplicações e divisões em somas c subtrações). Em 1642, o francês Blaise Pascal, desenvolveu uma máquina capaz de somar e subtrair números de 8 algarismos, utilizando engrenagens (constituída por rodas montadas sobre um eixo, em cada uma das quais era possível escrever uma cifra: 0, 1, 2,..., 9 provocando uma rotação cm torno do eixo proporcional à cifra em questão, essa máquina tinha um mecanismo que registrava, através do avanço de um dente da roda colocada a sua esquerda, toda vez que a roda voltava a passar pelo zero relativo). O matemático alemão Gottifried von Leibniz construiu uma máquina semelhante a de Pascal mas adicionou as operações de multiplicar e dividir (Leibniz idealizou um dispositivo que permitia lazer girar simultaneamente todas as rodas e armazenar o resultado em um totalizador, isso permite efetuar a multiplicação mecanicamente). Em 1801, o tecelão francês Joseph-Marie Jacquard constrói um tear comandado por um conjunto de cartões perfurados em seqüência, esse fato foi de enorme importância para o desenvolvimento das máquinas calculadoras. Entretanto, o maior precursor dos modernos computadores foi o matemático inglês Charles Babbage. Babbage foi o primeiro a propor a construção de uma calculadora automática seqüencial (que denominou máquina analítica), isto é, uma máquina cap.iz de

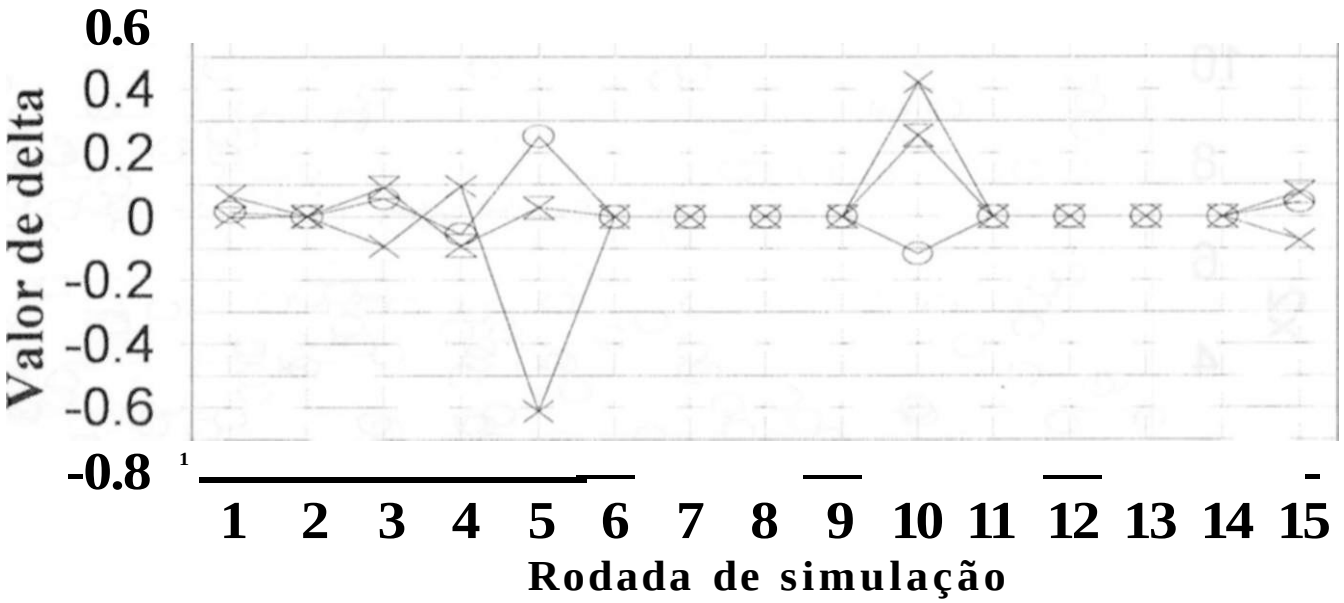
Problema de prog. linear

Comportamento de G(x1,x2) e Gmax



Problema de prog. linear

Comportamento dos Deltas

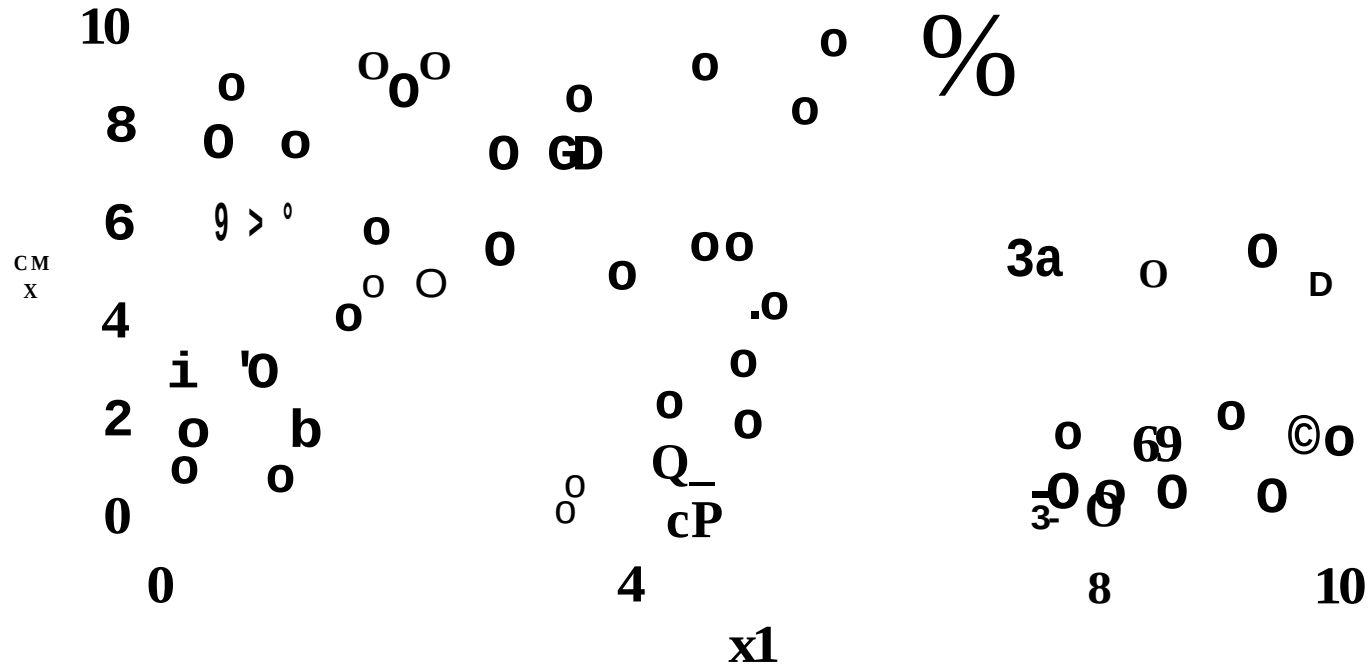


Delta x1 Fk - e - Delta x2Fk

x Delta f(x1 Fk,x2Fk)

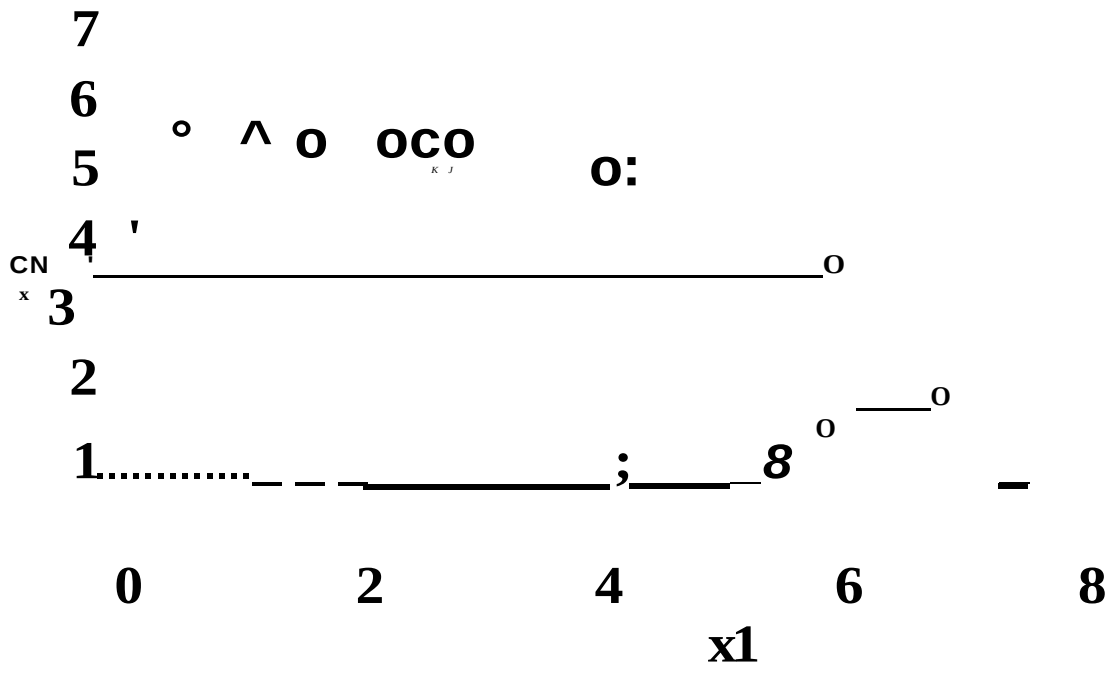
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação



Anexo Z

Função de teste 9 (Rosenbrock)

f(x1,x2) = 100(x2-x'^2)^2+x

Proposto em [Himmelblau] pacf. 394 como problem 2

Parâmetros

Da função

xlmin

Do algoritmo

Gama 1 Gama 2 Gama 3

Delta 1 Delta 2 Delta 3

Erro

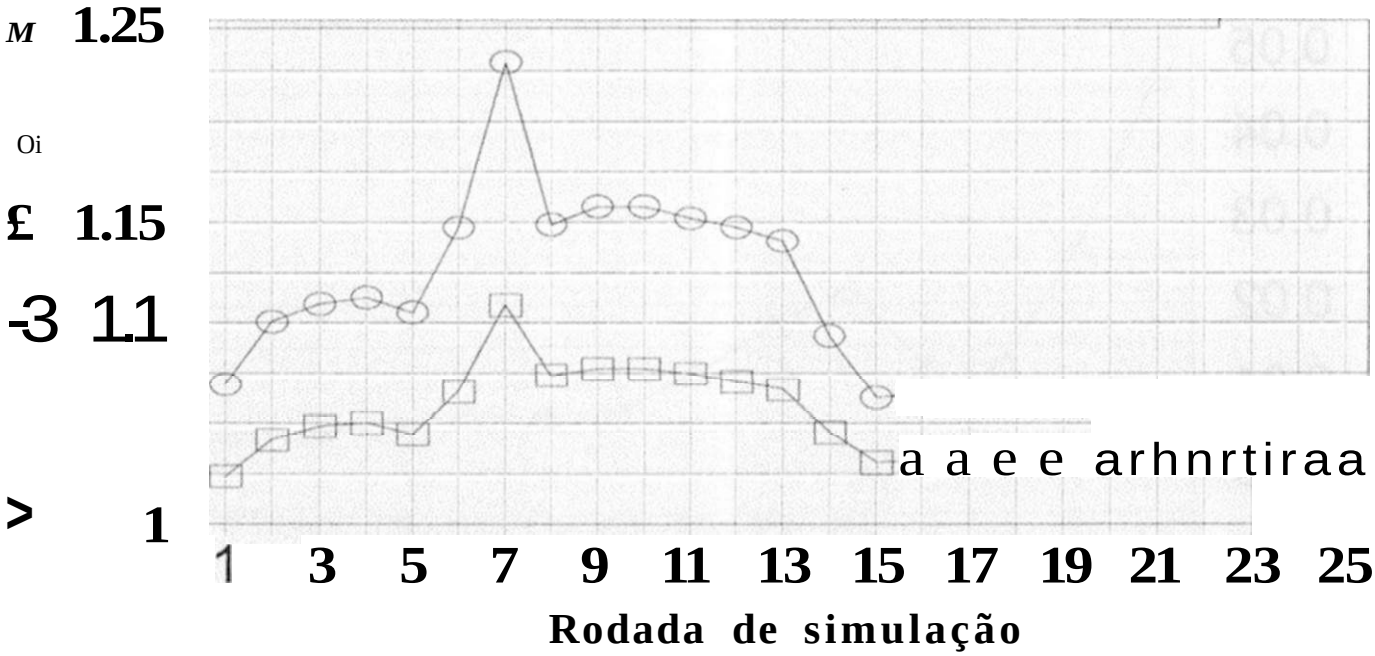
1E-15 1.03041 1.061854

0.000926

A	J	K	L	M	N	0	P	Q	R	S	T	u
3	Rodada ->	1	2	3	4	5	6	7	8	9	10	11
4	x1Fk	1.023657	1.041543	1.048455	1.050133	1.044658	1.065876	1.10894	1.073808	1.07738	1.077283	1.07459
5	x2Fk	1.069254	1.100664	1.109572	1.112665	1.105405	1.147546	1.229268	1.148922	1.157801	1.157648	1.152015
6	fmax	-0.00875	-0.00875	-0.00875	-0.00875	-0.00686	-0.00686	-0.00686	-0.00686	-0.00686	-0.00686	-0.00686
7	f(x1 Fk,x2Fk:	0.046272	0.026853	0.012984	0.012287	0.021859	0.017458	0.011891	0.007163	0.006856	0.006809	0.006308
8	Delta x1Fk	0.017886	0.006912	0.001678	-0.00548	0.021219	0.043064	-0.03513	0.003572	-9.7E-05	-0.0C269	-0.00396
9	Delta x2Fk	0.031409	0.008908	0.003094	-0.00726	0.042142	0.081722	-0.08035	0.008879	-0.00015	-0.00563	-0.00435
10	Delta f	-0.01942	-0.01387	-0.0007	0.009572	-0.0044	-0.00557	-0.00473	-0.00031	-4.7E-05	-0.0005	-0.00112

>	V	X	V	N	AA	AB	^O	AD	AE	AF	^O
CO	CO	‾k ‘	en	CO	- J	CO	‾k CO	ro o	ro	ro ro	ro CO
	1.070628	1.06689	1.045398	1.030338'	1.031492	1.030028	1.030028	1.030028	1.030555	1.030327	1.030356
cn	1.147662	1.140699	1.093539	eD≅Ĉ*4	1.064796	1.062035	1.062035	1.062035	1.061785	1.061894	1.06188
CD	-0.00686	-0.00553	-0.00211	-0.00102	-0.00102	-0.00102	-0.00102	-0.00102	-0.00099	-0.00093	-0.00093
	0.005189	0.005072	0.002108	0.001036	0.001059	0.001018	0.001018	0.001018	0.00094	0.00093	0.000928
∝>	-0.00374	-0.02149	-0.01506	0.001153	-0.00146	-3.4E-08	o	1.1E-13	0.000527	-0.00023	3E-05	4.1E-05
CD	-0.00696	-0.04716	-0.03087	0.002125	-0.00276	-6.4E-08	o	-5E-14	-0.00025	0.000108	-1.4E-05	CO m sg
‾k o	-0.00012	-0.00296	-0.00107	2.3E-05	-4.1E-05	-7.7E-10	o	cn m 4ŝ	-7.8E-05	-1E-05	-2.5E-06	-1.5E-06!

Função de teste 9



ii **Valor de x1Fk © Valor x2Fk**

executar automaticamente uma sequencia de operações predeterminadas. Embora Babbage não tenha conseguido construir sua máquina, cm virtude de problemas técnicos c financeiros, sua concepção introduziu os princípios gerais que nortearam a construção dos modernos computadores. Os princípios gerais traçados para essa máquina eram:

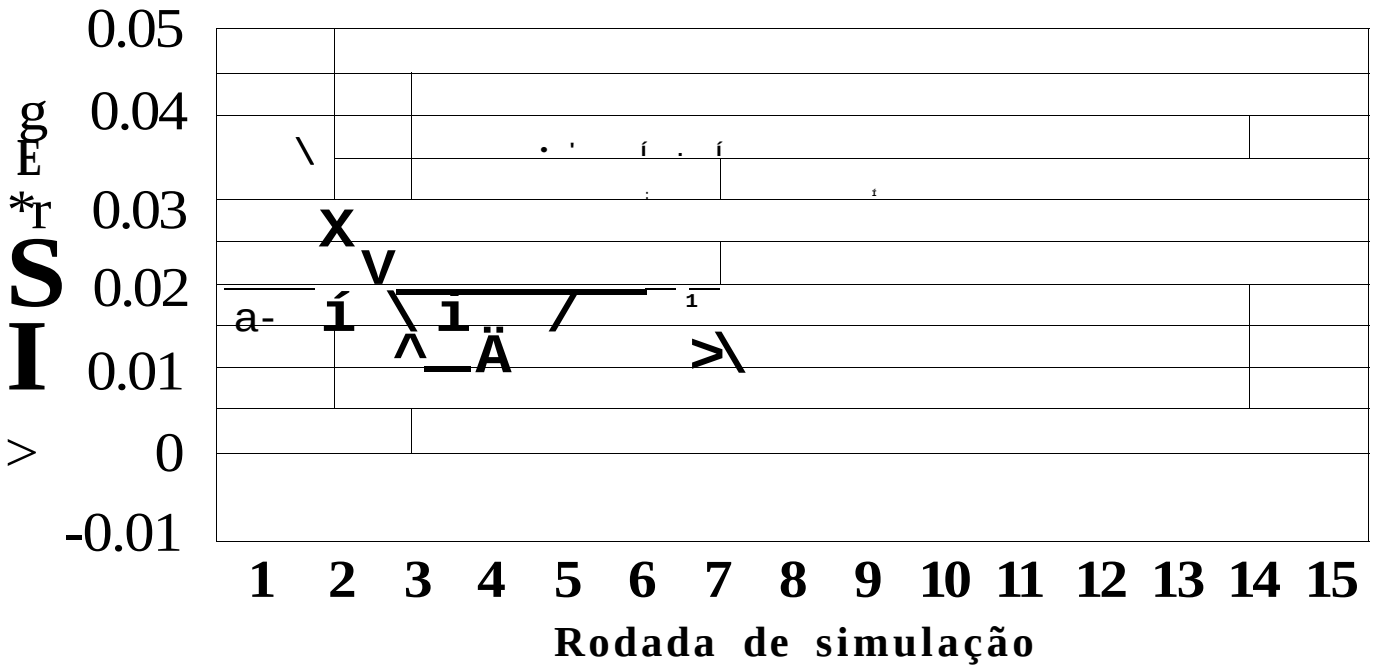
- Um programa deveria comandar a máquina;
- Uma memória deveria armazenar os resultados intermediários;
- Os cálculos deveriam ser realizados cm ciclos;
- Os resultados deveriam ler uma forma tal que sc adequassem as características da maquina.

Uma outra contribuição significativa foi dada pelo engenheiro americano Herman Hollerith, funcionário do Departamento Federal dc Estatística dos listados Unidos da America, sendo esse órgão o responsável pelo censo demográfico daquele país, realizado a cada dez anos. Chamou a atenção c assustou Hollerith o fato de que o censo de 1880 ter consumido o trabalho de 500 pessoas durante 7 anos, acusando uma população dc 55 milhões dc habitantes. Preocupado com o censo seguinte (1890), I Iollcrith procurou uma forma dc mecanizar os trabalhos dc apuração. Aproveitando o uso dos cartões perfurados idealizados por Jacquard, I Iollcrith projetou uma máquina que lia c processava os cartões; luram então construídas as máquinas a partir dc 1885 c o recenseamento dc 1890 ocupou 43 pessoas durante um ano. liste sucesso extraordinário propiciou a difusão dc máquinas a cartão no comercio, na indústria, etc.

Apesar do sucesso das máquinas a cartão dc I Iollcrith (particularmente apropriadas para trabalhos dc estatística c contabilidade - muitos números c poucas operações), o fato dc serem mecânicas tornava-as muito lentas. I louve então um grandioso incremento no empenho para o desenvolvimento dc máquinas mais rápidas. Mm 1944, lloward Aikcn da Universidade dc Haward, montou a primeira máquina automática decimal por programa, usando reles, o MARK l. Em 1946, John P. Ecker c John W. Mauchly, construíram o primeiro computador eletrônico na Universidade da Pensylvania, o ENIAC (Eletronic Numcric Intcgrator and Calculator), com 18.000 válvulas, essa máquina utilizava o sistema binário. O

Função de teste 9

Comportamento de f e fmax

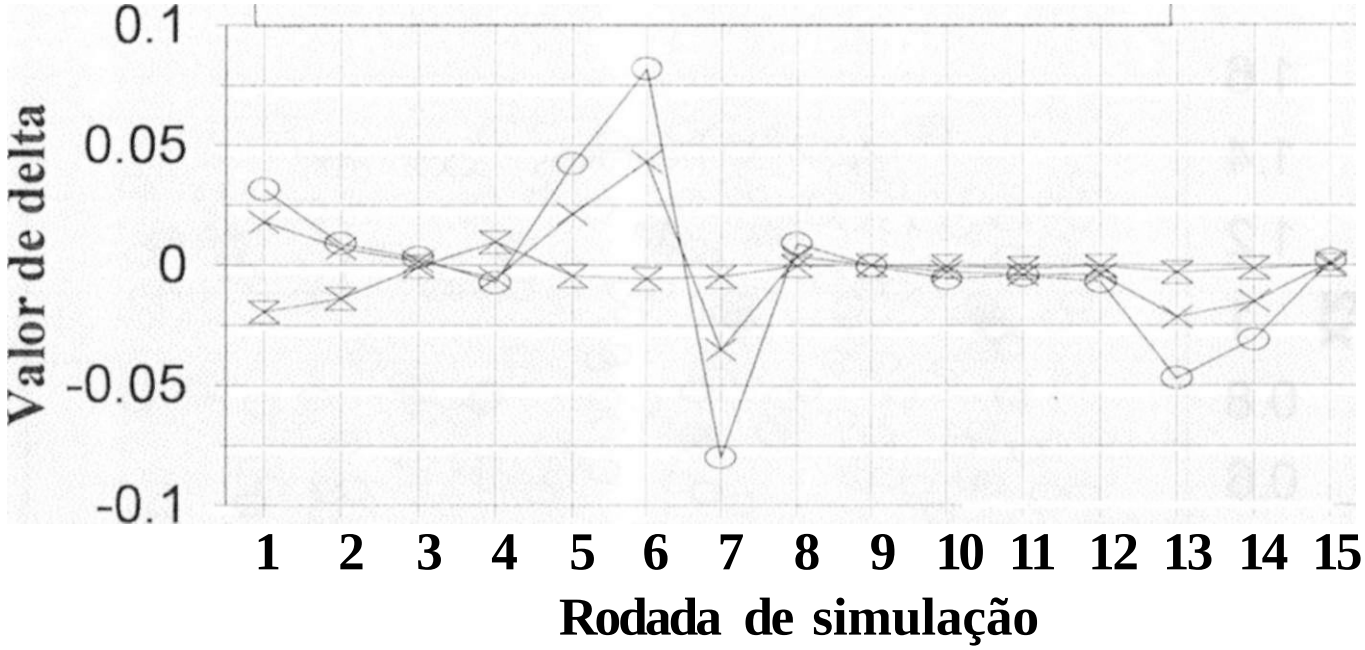


ii Valor de fmax

A Valor de f(x1Fk,x2Fk)

Função de teste 9

Comportamento dos Deltas

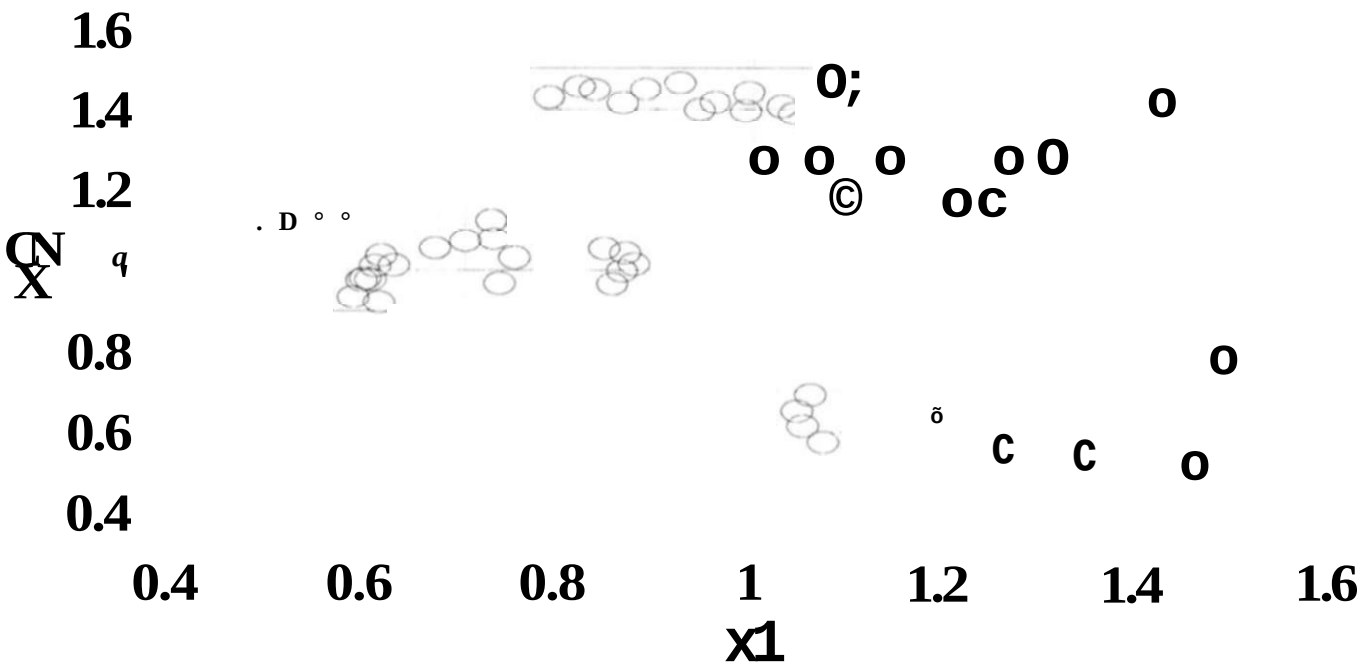


x **Delta x1Fk** o **Delta x2Fk**

\- **Delta f(x1Fk,x2Fk)**

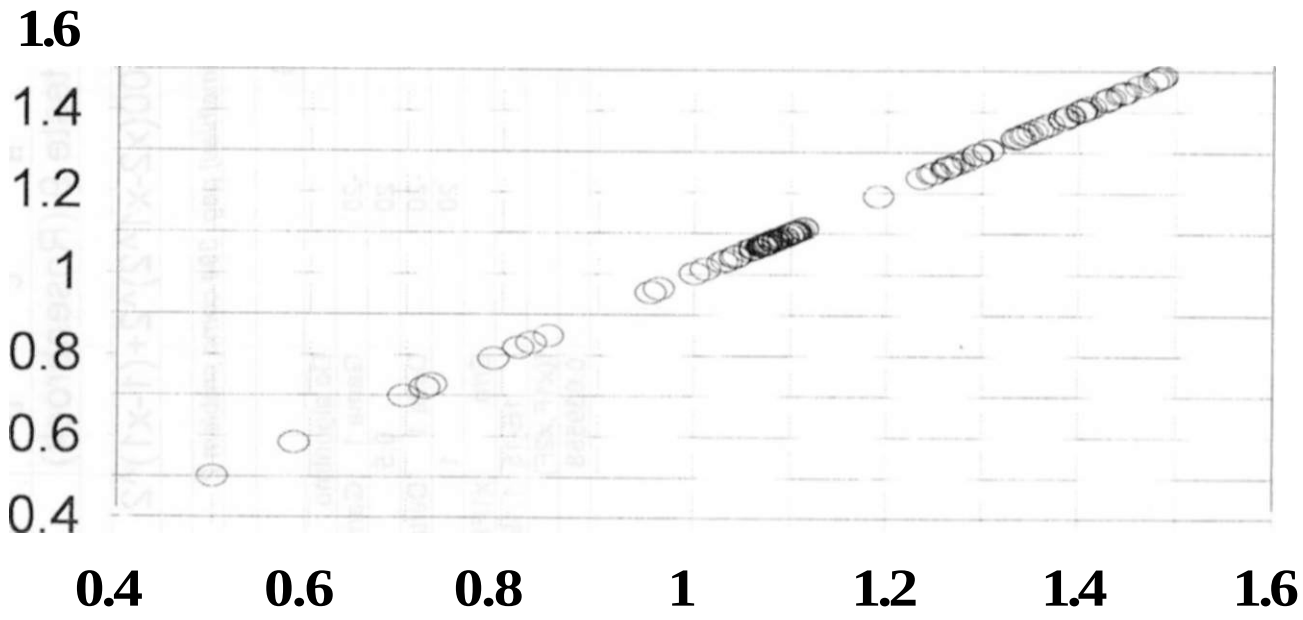
Nuvem de pontos

Primeira rodada de simulação



Nuvem de pontos

Vigésima rodada de simulação



Função de teste 9 (Rosenbrock)

Proposto em [Himmelblau] paç

Parâmetros

Da função	Do algoritmo	
xlmin	Gama 1	1E-15
	Gama 2	1.193077
	Delta 1	1.428206
	Delta 2	
	Delta 3	
	m	
	11	
		0.039558

Rodada -->												
fmax	2.865321	2.865321	2.865321	17445	17445	17445	-0.17445	1.215534	1.215534	1.215534	1.215534	1.215534
	7.943419	7.943419	7.943419		07533	.07533	-0.07533	1.544517	1.544517	1.544517	1.544517	1.544517
	-10.5893	-10.5893	-10.5893	49792	.49792	.49792	-2.49792	-0.49527	-0.49527	-0.49527	-0.49527	-0.49527
	10.58931	10.58931	10.58931	.49792	49792	.49792	2.49792	0.495269	0.495269	0.495269	0.495269	0.495269
	0	0	-3.03978	0	0	0	1.389989	0	0	0	0	0
	0	0	-8.01875	0	0	0	1.619846	0	0	0	0	0
Delta f	0	0	-8.09139	0	0	0	-2.00265	0	0	0	0	0

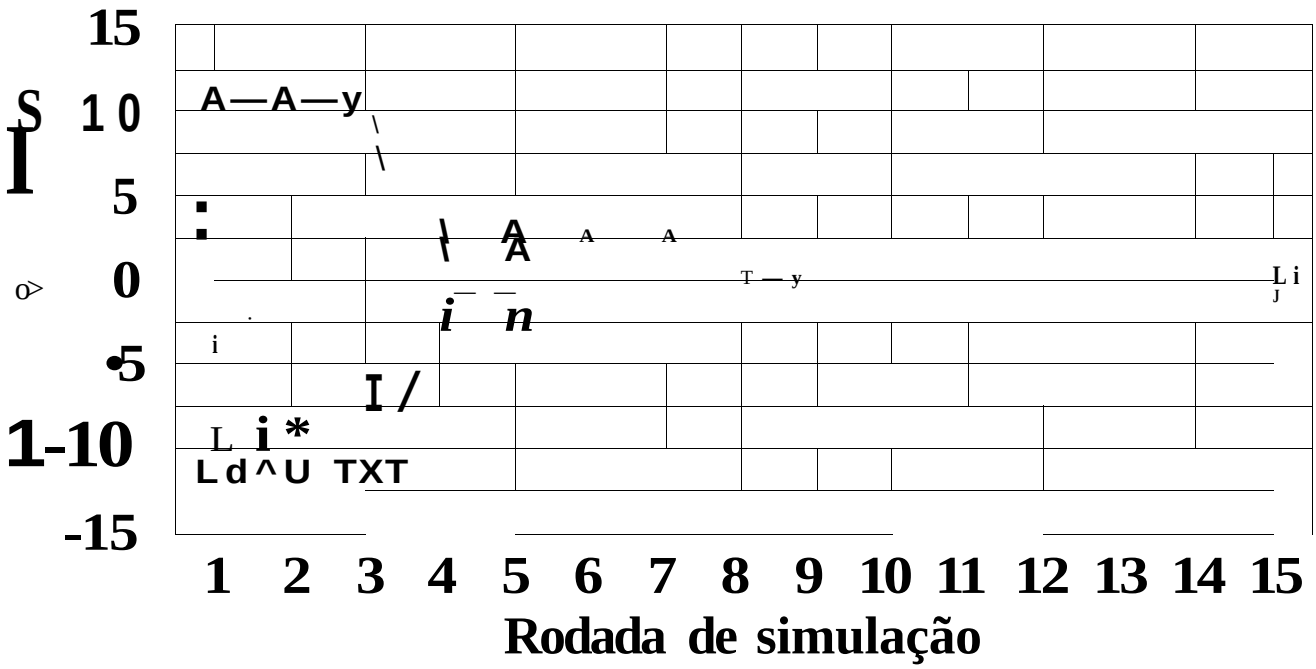
Função de teste 9

Comportamento de x1 Fk e x2Fk

Valor de x1 Fk **Valor x2Fk**

Função de teste

Comportamento de f e fmax



• B - Valor de fmax A Valor de f(x1Fk,x2Fk)

RNIAC representou um grande salto tecnológico em termos de velocidade dc processamento (o MARK I gastava 6 segundos para multiplicar dois números de 10 algarismos enquanto o RNIAC realizava a mesma tarefa em 3 ms). Imediatamente após o surgimento do KNIAC, surgiu um outro computador eletrônico, o EDVAC, onde, pela primeira vez, foi utilizada a idéia dc programação interna proposta por John von Neumann da Universidade de Princeton. Essa idéia sc difundiu de tal forma que hoje em dia é adotada pela quase totalidade dos computadores. Iodas essas máquinas a que nos referimos até agora eram máquinas não comerciais, só em 1951 foi anunciado o UNIVAC l, o primeiro computador comercial e logo após o IBM 701, iniciando a primeira geração de computadores. Bm meados da década de 50 aparecem os primeiros computadores completamente transistorizados dando início a segunda geração (IBM 1401). Bm 1960 é criada a primeira linguagem universal de programação de computadores, o COBOL (*Common Business Orieníed Ixmguage*). Na metade da década de 60 surgem os circuitos integrados e tem início a terceira geração de computadores; essa geração entretanto não é marcada unicamente pelo uso destes dispositivos, mas também pelo surgimento dos sistemas operacionais, um complexo conjunto dc rotinas de controle, que viabilizou a multiprogramação em larga escala (IBM /360).

A partir da segunda metade da década de 70 surge uma demanda por comunicação entre essas máquinas e muitos problemas aparecem, tais como, uma rede de telecomunicações que atendia muito precariamente essas necessidades, escassez de softwares para comunicações, incompatibilidade entre sistemas proprietários, etc. Surgem então as primeiras e tímidas soluções para esses problemas, tais como, redes especializadas para transmissão de dados, pacotes de comunicação implementando os mais diversos protocolos de transmissão dc arquivos, etc. A padronização internacional das interfaces (para comunicações entre máquinas, entre redes, etc.) foi um passo mais longo c mais efetivo na direção do incremento de comunicação entre as máquinas.

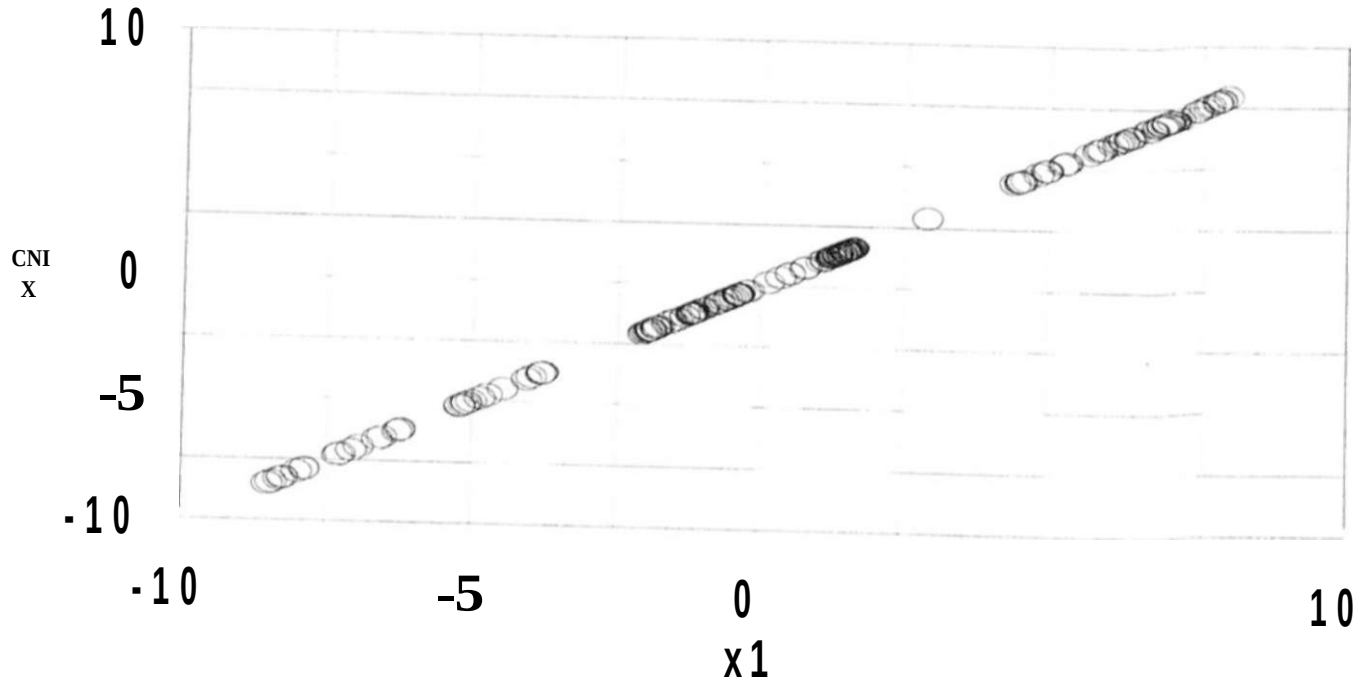
Foi também a partir da segunda metade da década de 70 que os computadores domésticos começaram a difundir-se de uma forma muito rápida devido ao seu baixo custo aliada a uma oferta de

Delta x1Fk e Delta x2Fk

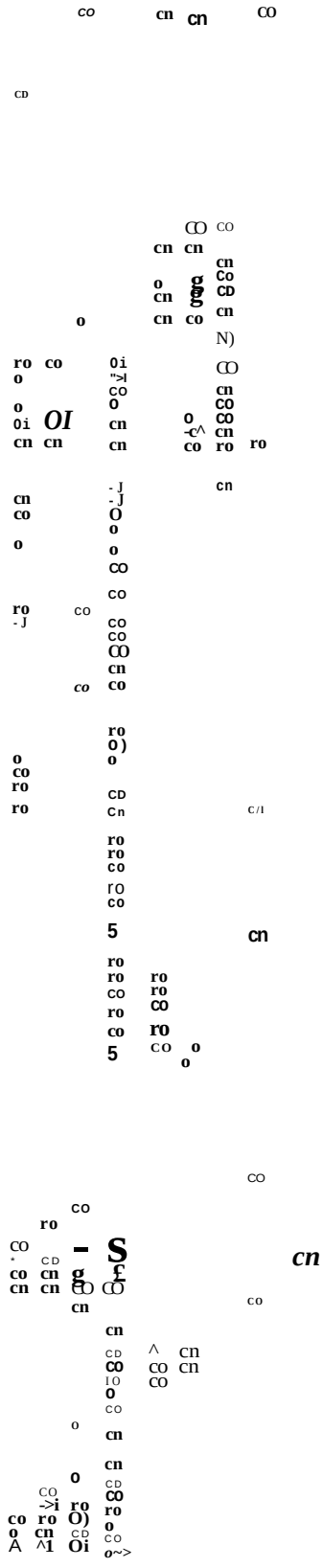
Delta f(x1Fk,x2Fk)

Nuvem de pontos

Vigésima rodada de simulação



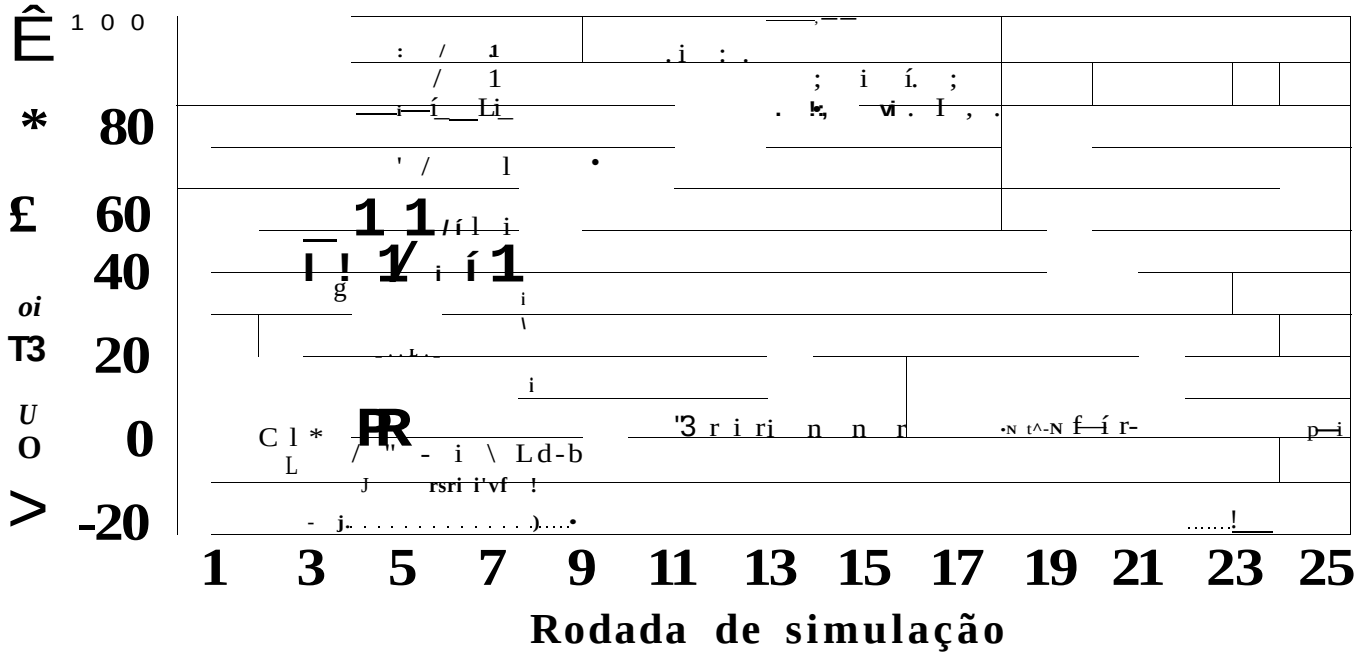
A	A	B	C	D	E	F	G	H
1	**Função de teste 9 (Rosenbrock)**							
2								
3	**f(x1,x2) = 100(x2-x1^2)^2+(1-x1)^2**							
4								
5	***Proposto em [Himmelblau] paçf. 394 como problem 2***							
6								
7	**Parâmetros**							
8	**Da função**			**Do algoritmo**				
9	**xlmin**	**-100**		**Gama 1**	**Gama 2**	**Gama 3**		
10	**xlmax**	**100**		**0.5**	**5**	**100**		
11	**x2min**	**-100**		**Delta 1**	**Delta 2**	**Delta 3**		
12	**x2max**	**100**		**1**	**1**	**1**		
13				**Erro**	**x1Fk**	**x2Fk**		
14				**1E-15**	**1.168214**	**1.364402**		
15				**f(x1F,x2F)**				
16				**0.028306**				



^	V	X	Y	N	AA	AB	AC	AD	^m	AF
8	[illegible]	8	[illegible]	8		8	8	[illegible]	[illegible]	[illegible]
	1.188766	[illegible]	1.160103	1.172338	1.164463	1.169545	1.169545	1.167494	1.168788	1.167964	1.167964
[illegible]	1.185057	1.43518	1.43518	'328419	1.397136	1.352786	1.352786	1.370684	1.359392	1.366582	1.366582
8	-5.23859	-0.82382	-0.82382	-0.2409	-0.19648	-0.0514	-0.0514	-0.03389	-0.03294	[illegible]	-0.02881
	5.238992	0.82382	0.82382	0.240901	0.196483	0.051397	0.051397	0.033894	0.032944	0.028808	0.028808
[illegible]	-0.02866	[illegible]	0.012234	-0.00787	0.005082	[illegible]	[illegible]	-0.00205	0.001294	-0.00082	[illegible]	0.000325
[illegible]	0.250124	[illegible]	-0.10676	0.068717	-0.04435	[illegible]	[illegible]	0.017898	-0.011291	[illegible]	[illegible]	-0.00284
[illegible]	-4.41517	[illegible]	-0.58292	-0.04442	-0.14509	[illegible]	[illegible]	-0.0175	-0.00095	-0.00414	[illegible]	-0.0CC35

Função de teste 9

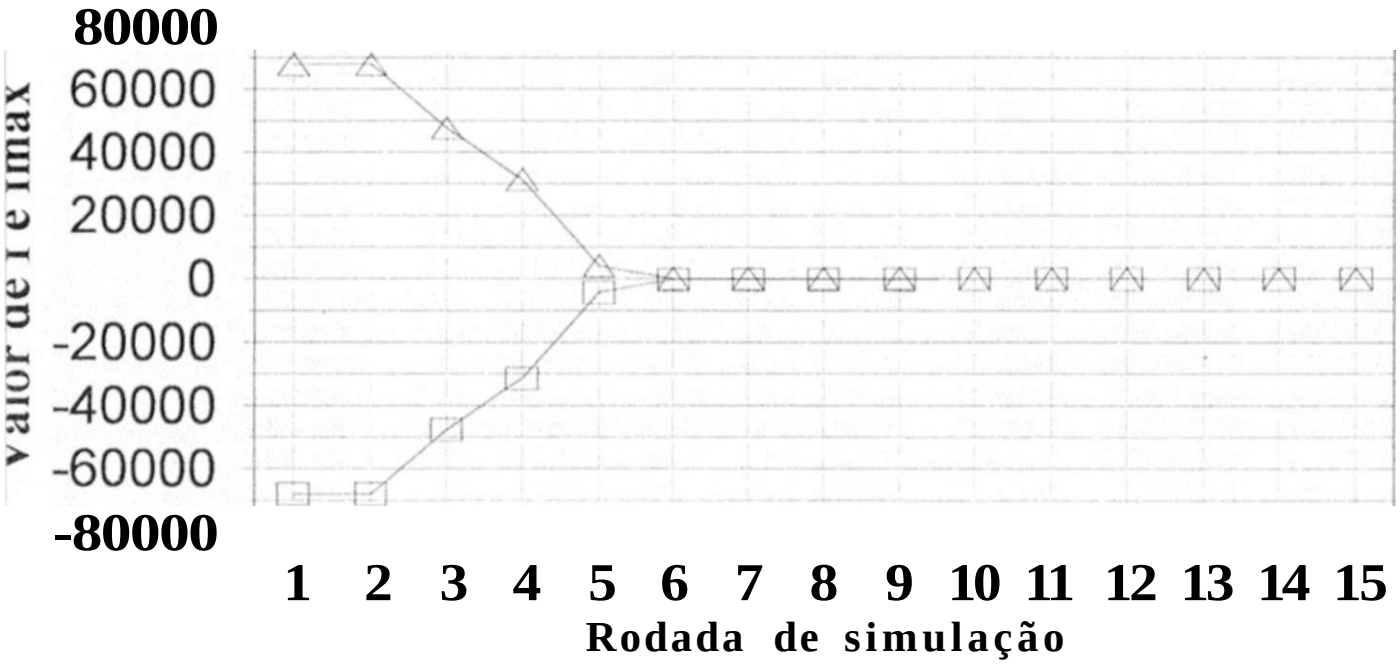
Comportamento de x1 Fk e x2Fk



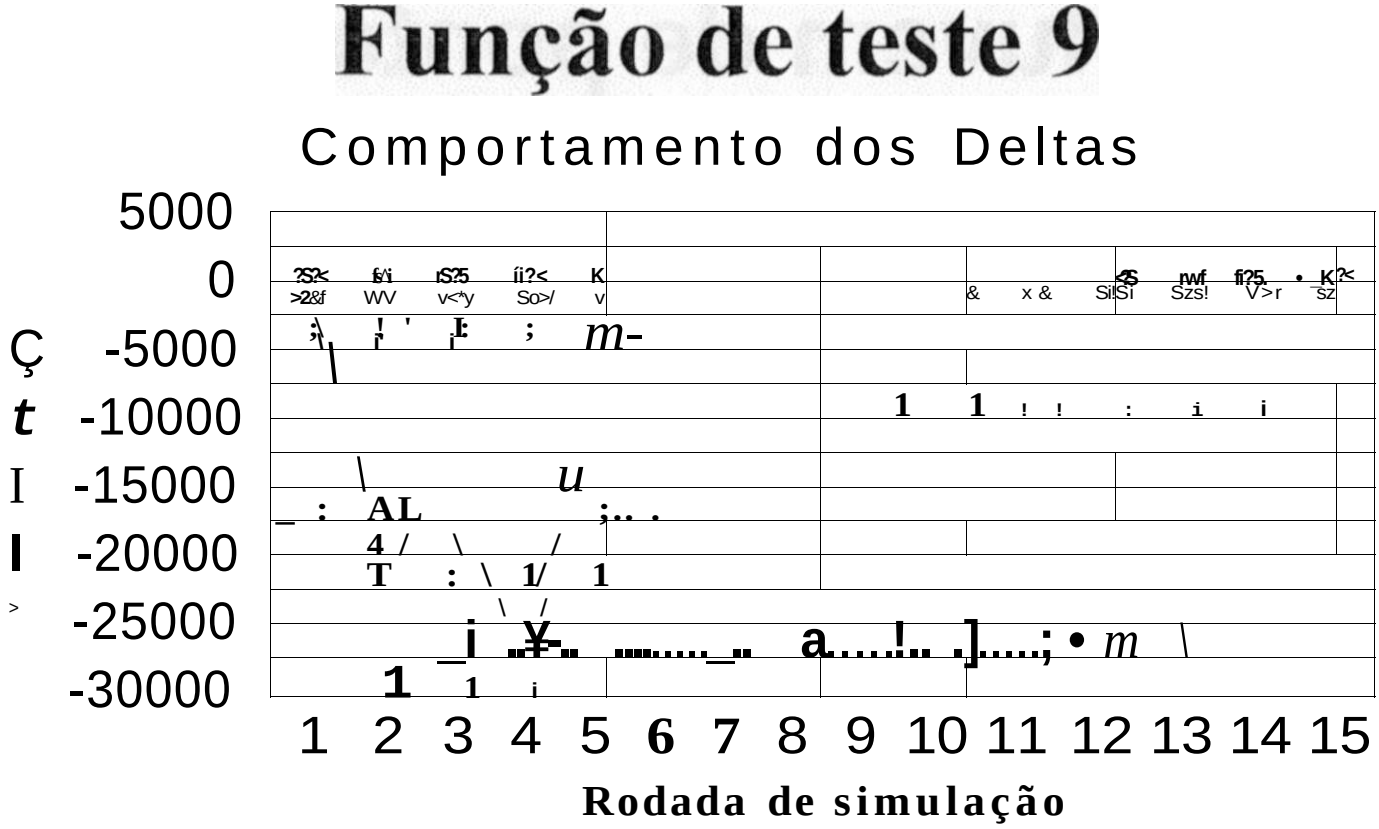
i i **Valor de x1Fk e Valor x2Fk**

Função de teste

Comportamento de f e fmax

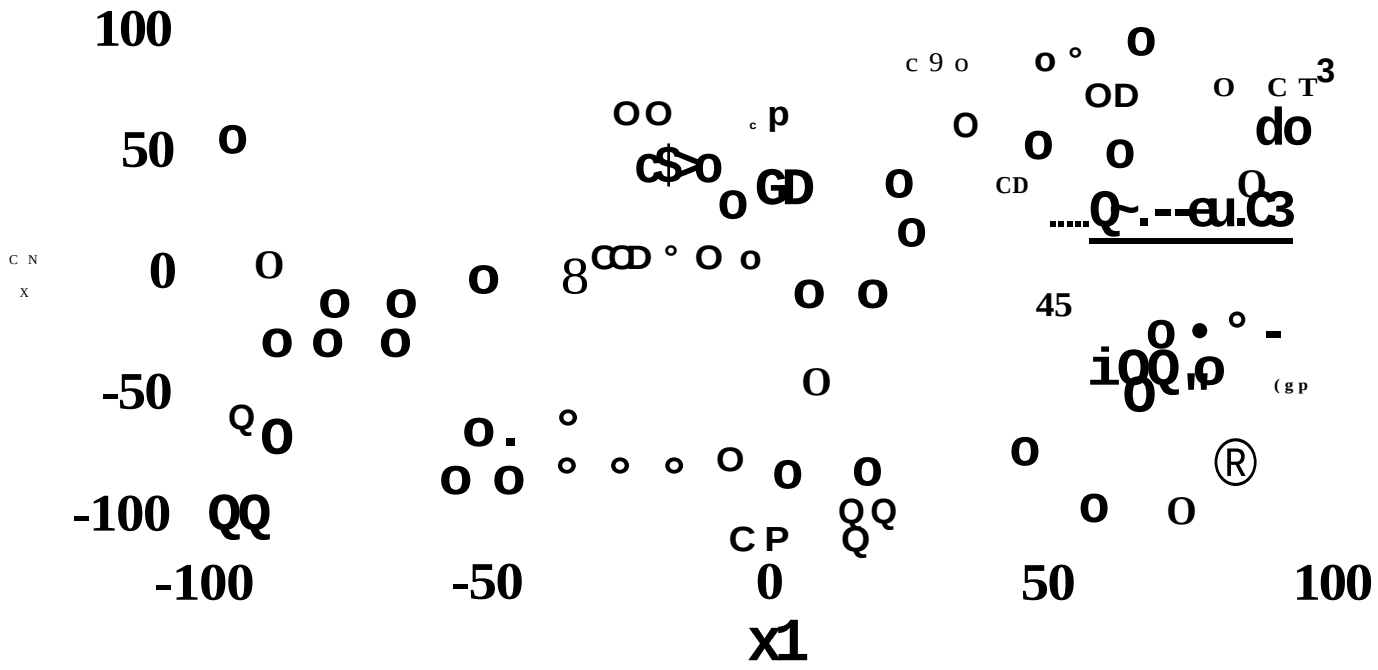


Função de teste 9



Nuvem de pontos

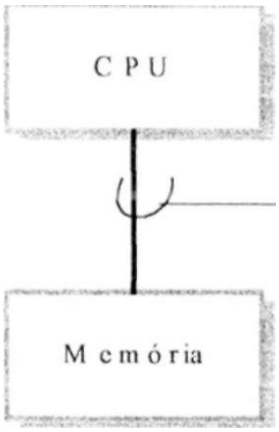
Primeira rodada de simulação



serviços crescente relativa à oferta de softwares padrões e de uso geral (planilhas, gerenciadores de bancos de dados, processadores de texto, etc), de softwares específicos, mas de baixo custo (pacotes estatísticos, contas a pagar, contas a receber, pacotes matemáticos, etc.) e principalmente a propagação de bancos de dados ligados em redes com acesso público repletos de informações as mais diversas e para os mais diferentes perfis de usuários.

Todos esses fatores convergiam para uma necessidade crescente dos usuários em relação a suas maquinas: velocidade. Velocidade aqui entendida da forma mais ampla, isto é, velocidade de *clock* para a cpu, velocidade de acesso a memória, diminuição do tempo de *seek* dos *hard disks,* velocidade de comunicação, velocidade nos algoritmos c em suas implementações, etc.

Todo esse empenho tem um objetivo comum que é a diminuição da influência na performance da máquina (ou mesmo a sua eliminação) do efeito provocado pelo chamado gargalo de von Neumann, ilustrado na figura 3.1.1 abaixo.



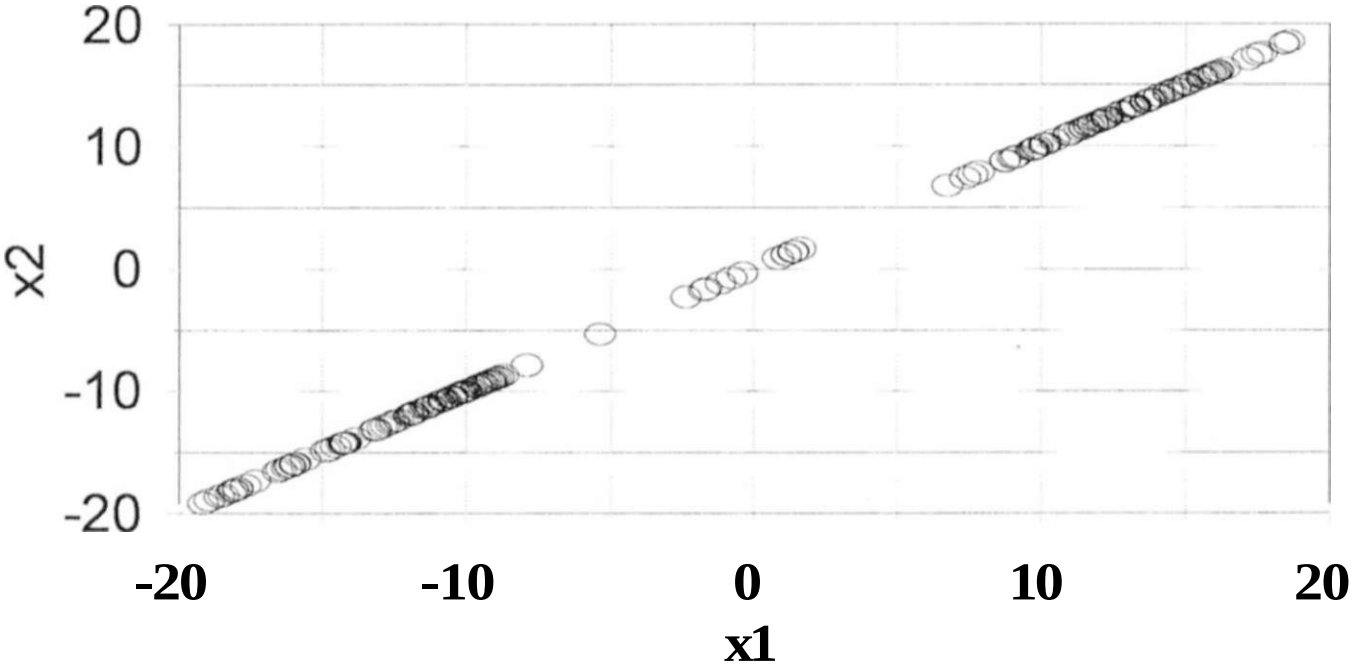
gu rga lo

Figura 3.1.1 **- G a r g a l o d c von Neumann.**

Para tornar esse objetivo possível tem havido um empenho crescente de todos os interessados (fabricantes, universidades, empresas cm geral, governo, etc). Todavia, a maior parte desse esforço tem sido utilizada

Nuvem de pontos

Vigésima rodada de simulação



Bibliografia

[1] ACKLEY, David H., HINTON, G. E., SEJNOWSKI, T. J. -'A learning algorithm for Boltzmann machines'- Cognitive Science No 9, 1985.

[2] AHUJA, Sanjeev B., REGGIA, James A., BERNDT, Rita S. -'Automated classication of phonological errors in aphasic language'- Computer methods and programs in biomedicine, 21(1985).

[3] ALURU, S., PRABHU, G. M. & GUSTAFSON, J. -'A random number generator for parallel computers' - Parallel Computing, 18 (1992).

[4] ANDERSON,Kennet W.,HALL, Dick Wick - 'Sets, sequences, and mappings: the basic concepts of analysis'- John Wiley & Sons, lnc - 1963.

[5] ARNOLD, V. 1. -'Teoria da catástrofe'- Editora da Unicamp - 1989.

[6] AVR1EL, Mordecai - 'Nomlinear programing: analysis and methods'- Prentice-Hall, New Jersey - 1976.

[7] BERTALANITY, Ludwig von - 'Teoria geral dos sistemas'- Editora Vozes - 1973.

[8] BHAT, B. R. - 'Modern probability theory - an introductory text book'- second ediction - A Halsted press book - 1985.

[9] BOGNI, Carlos, MARRONE, Luis - ' Arquitecturas no convencionalcs' - I Escola Brasileiro-Argcntina de informática - Editorial Kapelusz S. A. - Buenos Airies - 1987.

[10] CAMPELLO DE SOUZA, Fernando M - 'Decisões racionais em situações de incerteza' - Tese para concurso público de professor titular do DES/UFPE - 1993.

[11] CET1N, B. C , BARHEN, J , BURDICK, J. W. -'Terminal repeller unconstrained subenergy tunneling (trust) for fast global optimization'- Journal of Optimization Theory and Applications: Vol. 77, No. 1, april 1993.

[12] CHURCHMAN, C. West - 'Introdução a teoria dos sistemas'- Editora Vozes - segunda edição - 1972

[13] CONTE, S. D. -'Elementos de análise numérica'- Editora Globo, terceira edição, Porto Alegre - 1977.

[14] COSTA, Luis Sergio Salles, CAULL1RAUX, Heitor M. -'Manufatura integrada por computador - sistemas integrados de produção: estatégia, organização, tecnologia e recursos humanos'- Editora Campus, 1995.

[15] D'AMBRÓSIO, Ubiratan -'Métodos da topologia - Introdução e aplicações'- LTC editora SA - São Paulo - 1977.

[16J DEM1DOVICH, B. P., MARON, 1 A. -'Computational mathematics'- MIR Publishers Moscow - 1976.

[17] DERTHICK, Mark -'Variations on the Boltzmann machine learning algorithm'- Departament of computer science - Carnegie Mellon University- CMU-CS-84-120, august 1984.

[18] D1ANESE, A. - 'Computação e simulação analógica e híbrida' - São Paulo, McGraw-hill - 1977.

[19] DUAN, Q Y , GUPTA, V. K , SOROOSHIAN, S. -'Shuffled complex evolution approach for effective and efficient global minimization'- Journal of Optimization Theory and Applications: Vol. 76, No. 3, march 1993.

[20] ENSLOW Jr, Philip H., -'Multiprocessors and parallel processing'- John Wiley & Sons, Inc - 1974.

[21] FAHLMAN, Scott E.-'Parallel processing in artificial intelligence'- Parallel Computing 2(1985) 283-286, North-Holland.

[22] FELLER, William - 'An introduction to probability theory and its applications' - volume one - John wiley & Sons, Inc.

[23] FERNANDEZ, P. J. -'Introdução a teoria das probabilidades'- LTC Editora SA - Rio de Janeiro - 1973.

[24] FLOUDAS, Christodoulos A., PARDALOS, Panos M.-Recent advances in global optimization' - Princeton University Press.

[25] GIOZZA, W. F., ARAÚJO, J. F. M., MOURA, J. A. B., SAUVÉ, J. P. -'Redes de computadores - Tecnologia e aplicações'- McGraw-Hill do Brasil, 1986.

[26] GLORIA, A D., FARABOSCHI, P. & R1DELLA, S -'A dedicated massively parallel architeture for the Boltzman machine'- Parallel Computing, 18 (1992).

[27] GRAYBEAL, W , POOCH, U -'Simulatiomprinciples and methods'- Cambridge, Winthrop - 1980.

[28] HAMMERSLEY, J. M., HANDSCOMB, D. C. -'Monte Carlo methods'- Fletcher & Son Ltd, Norwich, Inglaterra, 1964.

[29] HAMMERSLEY, J. M. , MORTON, K W. -'A new Monte Carlo technique: antithetic variates', Proc. Cambridge Phil. Soc, vol. 52, pt. 3, pp. 449-457, 1956.

[30] HASSOUN, Mohamad -'Fundamentals of artificial neural networks'- MIT Press 1995.

[31] HANXLEDEN, R v., SCOTT, L R -'Correctness and determinism of parallel Monte Carlo Processes'-Parallel Computing, 18 (1992).

[32] HIMMELBLAU, David M - 'Applied nonlinear programming'- McGRAW-Hill Book company, 1972

[33] HINDIN, Harvey J. -'Parallel processing promises faster program execution'- Computer Design pag 57-66, august 15, 1985.

[34] HINTON, G. E , SEJNOWSKI, T. J -'Apprentissage dans les machines de Boltzmann'- Cognitiva 85, 4-7 juin 1985, Paris.

[35] HIRIART-URRUTY, J. B. -'Conditions for global otimality'- Kluwer Academic Publishers - 1995.

[36] HOLDEN, A. D. C. -'Trends in artificial intelligence'- IEEE Transactions on Computers, Vol. C-25, No. 4, april 1976.

[37] KLIR, George J. -'Generalized information theory'- Fuzzy sets and systems, 40, pg. 127-142, 1991.

[38] KOBAYASHI, Hisashi -' Modeling and analysis - an introduction to system performance evaluation methodology' - I B M corporation - Addison-Wesley publishing company - 1978.

[39] KOPCHENOVA, N V., MARON, I A - 'Computational mathematics - worked examples and problems with elements of theory'- MIR Publishers - Moscow - 1975.

[40] LACKEY, Stan, VERES, Jim, ZIEGLER, Mike -'Supercomputer expands parallel processing options'- Computer Design pag 76-81, august 15, 1985.

[41] LASDON, Leon s,-'Optimization theory for large systems'- Macmillan Publishing Co., Inc., 1970.

[42] LEE, Alec M. - 'Applied queueing theory'- London, Macmillan - 1966.

[43] LEE, Y. W. - 'Statistical theory of communication ' - John Wiley & Sons, Inc. - 1960

[44] LEWIS, T., SMITH, B. -'Computer principles of modeling and simulation'- Boston, Houghton Mifflin - 1979.

[45] LILEGDON, W., MARTIN, D., PRITSKER, A -'FACTOR/AIM: A manufacturing simulation system' - Simulation 62(6) 367-372, June 1994.

[46] LOEVE, Michael - 'Probability theory' - D. Van Nostrand Company, Inc - 1955.

[47] LOOTSMA, F. A. - 'Numerical methods for non-linear optimization'- Academic Press - 1972.

[48] LU, S. & ZHANG, G. -'A combined inductive learning and experimental design approach to manufacturing operation planning'- Journal of Manufacturing Systems, 9(2): 103-115, 1990.

[49] MAHEY, Philippe - 'Programação não linear - introdução a teoria e aos métodos '

Editora Campus - 1987.

[50] MARTIN, J. -'Design of real-time computer systems'- Englewood Cliffs, Prentice-Hall - 1967.

[51] MATTEIS, A. De, PAGNUTTI, S -'Controlling correlations in parallel Monte Carlo'- Parallel Computing, 21 (1995).

[52] MENASCE, Daniel A , BARROSO, Luiz A -'A methodology for performance evaluation of parallel applications on multiprocessors'- IBM - Rio de Janeiro 1984.

[53] MORLEY, Richard E.-'Networking multiple processors increases computing speed'- I&CS September, 1990.

[54] MATTEIS, A De, PAGNUTTI, S. -'Controlling correlations in parallel Monte Carlo'- Parallel Computing, 21, pg 73-84, 1995.

[55] NAKAMURA, Makoto, SASASE, Iwao, MORI Shinsaku -'Two parallel queues with dynamic routing under a threshold-type scheduling'- The transactions of the IE1CE, Vol. E 73, No. 3 march 1990.

[56] NAYLOR, Thomas H. -'Experimentos de simulación en computadoras con modelos de sistemas económicos'- Editorial Limusa, Mexico, 1977.

[57] NAYLOR, T. H., BALINTFY, J. L., BURDIK, D. S., CHU, K -'Técnicas de simulação em computadores'- Editora Vozes Ltda em colaboração com a Editora da Universidade de São Paulo, 1971.

[58] OREN, T. -'Quality assurence paradigms for artificial intelligence in modelling and simulation'- Simulation,48(4):49-51 ,apr, 1987.

[59] OREN, T. & ZEIGLER, B. -'Artificial intelligence in modelling and simulation: directions to explore' - Simulation, 48(4): 131-4,apr, 1987.

[57] OURMAEV, A. -'Elements de simulation sur calculateurs analogiques'- Editions MIR, Moscou 1978.

[58] RIETMAN, Edward - 'Exploring parallel processing '- Windcrest Books

[59] RIBEIRO, Clóvis Augusto - 'Implementação de um sistema de algoritmos de programação não linear '- Tese de mestrado - COPPE - 1973.

[60] RUDIN, Walter - 'Principles of mathematical analysis'- McGraw-Hill Book Company, Inc - New York - 1953.

[61] SHANG, Y., WAH, B. W. -'Global optimization for neural network training'- Computer, march 1996.

[62] SHIM1ZU, Tamio -' Simulação em computadores digitais' - Editora da Universidade de São Paulo - 1975.

[63] SCHOEMAKER, S. - 'Computer networks and simulation' - Amsterdam, North-Holland - 1978.

[64] SHARK, L. K., TERRELL, T. J., SIMPSON, R. J. -'New high-speed adaptive frame syncronisers incorporating postdetection processing techniques'- IEE Proceedings-1 Vol. 138, No. 4, august 1991.

[65] SOUCEK, B., SOUCEK, M. -'Neural and massively parallel computers - the sixth generation'- A Wiley-lnterscience Publication, John Wiley & Sons, 1988.

[66] SVIOKLA, John J. -'Expert systems and their impact on the firm: the effects of planpower use on the information processing capacity of the financial collaborative'- Journal of Manegement Information Systems, Vol. 6, No. 3, winter 1989-1990.

[67] SPELT, P., LYNESS, E., DESAUSSURE, G. -'Development and training of a learning expert system in an autonomous mobile robot via simulation'- Simulation,53(5):223-228,nov.,1989.

[68] SPRINGER, C. H., HERLIHY, R. E., MALL, R. T., BEGGS, R. I -'Modelos probabilisticos- série de matemáticas para Ia direción de negócios'- Union Tipográfica Editorial Hispano Americana, México, 1972.

[69] STRONGIN, R G., & SERGEYEV, Y. D. - 'Global multidimensional optimization on parallel computer' - Parallel Computer, 18 (1992).

[70] STUCKMAN, B. E., EASOM, E. E. -'A comparison of Bayesian/sampling global optimization techniques'- IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 5, september/october 1992.

[71**1** TAKAHASHl, Y , HASHIDA, O -'Delay analysis of discrete-time priority queue with structured inputs'- Queueing systems theory and applications, 8, pg. 149-164, 1991, Switzerzand.

[72] TANK, David W., HOPFIELD, John J. -'Collective computation in ncuronlikc circuits'-

[73] TORN, Aimo, VIITANEN, Sami -'Topographicalglobaloptimization'——————

[74] USHERWOOD, Peter N. R.-'Sistemas nervosos'- Editora da Universidade de São Paulo, São Paulo, 1977.

[75] VUOLO, J. Henrique -'Fundamentos da teoria de erros'- Editora Edgard Blucher Ltda, São Paulo - 1992.

[76] WAGNER, Harvey M. - **'Pesquisa operacional**'- Prentice / Hall do Brasil - 1986 - 2ª edição.

[77] WESTE, Neil, BURR, David J., ACKLAND, Bryan D **-'Dynamic time warp pattern matching using an integrated multiprocessing array'-** IEEE Transactions on computers, vol. c-32 no. 8, august 1983.

[78] YAKOWITZ, Sidney J. - **'Computational probability and simulation** ' - Addison-Wesley Publishing Company - 1977.

para melhorar a performance das máquinas comerciais existentes, dc tal forma, a conseguir melhorar a qualidade c diminuir o preço, esse fenômeno acentuou-se a partir da consolidação dos computadores pessoais c hoje é a tendência geral da indústria dc computadores cm todo o mundo.

Em [Enslow] são apontadas quatro áreas distintas, mas interrelacionadas, que tem sido desenvolvidas para que sc possa evoluir mais ainda nessa direção, são cias:

- circuitos e dispositivos
- arquiteturas
- organização
- ***software***

Em relação aos circuitos e dispositivos tivemos uma grande evolução no que diz respeito ao aumento dc velocidade de comutação das portas lógicas, incremento considerável na capacidade dc processamento das cpu's, diminuição do tempo dc acesso a memória, melhoria na performances dos ***modems*** c surgimento dc novos dispositivos.

Em termos dc arquiteturas tem-se conseguido evoluções bastante razoáveis a ponto de diminuirmos cm boa parte o tempo necessário para execução dc funções básicas, tanto aritméticas quanto lógicas, alem, evidentemente, dc pacotes de uso geral (planilhas por exemplo) implementarem boa parte do fcrramcntal estatístico, matemático c financeiro necessário a grande maioria dos usuários.

Relativamente a organização temos tido um avanço considerável no que diz respeito a **topologia** dc conexão e interconexão dos diversos circuitos e dispositivos constituintes do computador, utilização dc memoria cache ou não, etc.

Acompanhando c, muitas vezes, forçando essa evolução estão os ***softwares*** cada dia mais exigentes em termos de recursos da máquina tais como memória, velocidade de relógio, recursos dc interrupção,'etc.

Em [Bogni] são apontados três fatores que influenciaram o desenvolvimento do processamento eletrônico dc dados na busca dc máquinas mais rápidas c de melhor performance, estes fatores estão colocados na tabela 3.1.1 abaixo.

	Área dc influência			
rator dc influência	Estrutura	Organização	Implementação	Performance
Tecnologia	*Pequena*	*Média*	*Grande*	*Grande*
Usuário	*Média*	*Grande*	*Média*	*Grande*
Comunicações	*Grande*	*Média*	*Pequena*	*Pequena*

Tabela 3.1.1 - Fatores que influenciaram o desenvolvimento dc máquinas mais rápidas segundo [Bogni].

Verifica-se facilmente que, todo esse empenho, na tentativa dc conseguir melhorar a performance das máquinas sem alterar substancialmente o seu conceito, isto c, máquinas digitais dc estados finitos, baseadas na concepção dc von Neumann, tende a um limite.

Neste sentido a demanda por máquinas mais e mais velozes, com alta capacidade dc memória (cache, principal c dc massa), dispositivos dc I/O mais sofisticados c, nesse momento, facilidade dc comunicação nunca vistas, permitindo a interconexão de computadores isolados formando redes, c conectando redes dc tal modo que se possa comunicar com qualquer máquina individual dentro dessa estrutura complexa c sofisticada, tende a aumentar.

O grande problema que procura-se resolver c diminuir ou mesmo eliminar o procedimento seqüencial para a resolução dc tarefas, isto é, a tarefa k I I só será iniciada quando a tarefa k tiver sido concluída; isto mesmo que essas duas tarefas não tenham nenhuma dependência funcional ou lógica. Um exemplo disso está acontecendo nos microcomputadores pessoais, processadores especializados, sistemas operacionais

multitarefas, etc, com o intuito de melhorar o desempenho realizando tarefas de forma concorrente ou mesmo paralelas.

A figura 3.1.1 abaixo ilustra os níveis de processamento paralelo, a medida em que o nível diminui os objetos que são manipulados também diminuem e aumenta o paralelismo c crescem os problemas de sincronização e escalonamento. De acordo com [Almeida & Árabe] considera-se como processamento distribuído quando um sistema opera no nível 5, nos níveis 3 e 4 processamento paralelo c nível 2 processamento vetorial.

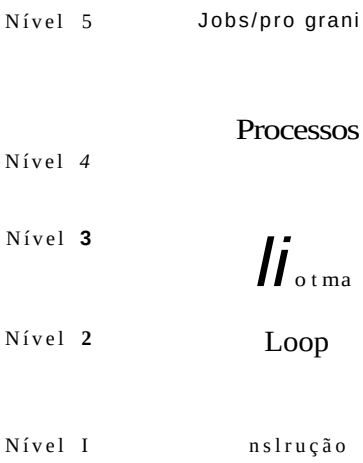


Figura 3.1.1 - Níveis de paralelismo segundo [Almeida &

A questão relativa à quebra do paradgma de von Neumann até algum tempo atrás era considerado apenas em círculos restritos da comunidade usuária de computadores. A demanda cada vez maior por processamento de sinais de voz e sinais de imagem em tempo real c outras aplicações que não exigem respostas cm tempo real, contudo envolvem modelos matemáticos que demandam muitos cálculos, tais como: processamento de cálculos científicos, modelagem c simulação de sistemas sociais e econômicos, sistemas que fazem previsões meteorológicas, etc, tem levado um número cada vez maior de pesquisadores, empresas e universidades a pesquisar novas arquiteturas, softwares e algoritmos para os

computadores. Sabe-se dc antemão que essa questão só será resolvida com modificações muito fortes dos dois principais componentes dc um sistema de computação: ***software*** c ***hardware.***

Em [Almeida & Árabe| c apresentada a tabela 3.1.2 abaixo que mostra a evolução dos computadores cm termos dc desempenho c c mostrado o tipo dc CPU, deve-se observar que a introdução do processamento vctorial na arquitetura do ***Cray*** 1 possibilitou um aumento dc um fator dc 10 no desempenho.

Modelo	*Ano*	*Desempenho relativo*	*Palavra (bits)*	*Memória (bytes)*	*CPU*
IBM 700	1954	1	36	32K	Seqüencial
IBM 7000	1959	5	36	32K	Seqüencial
CDC 6600	1965	25	60	128K	Seqüencial
CDC 7600	1969	100	60	5I2K	Seqüencial
LLLIAC IV	1972	200	60	128K	64 Processadores
Cray-1	1976	2000	64	2M	Vctorial
Cyber 205	1976	2000	64	32M	Vetorial
Modernos	1991	2750000	64	256M	Vct./Par.

Tabela 3.1.2 - Evolução no desempenho dos computadores segundo [Almeida & Árabe].

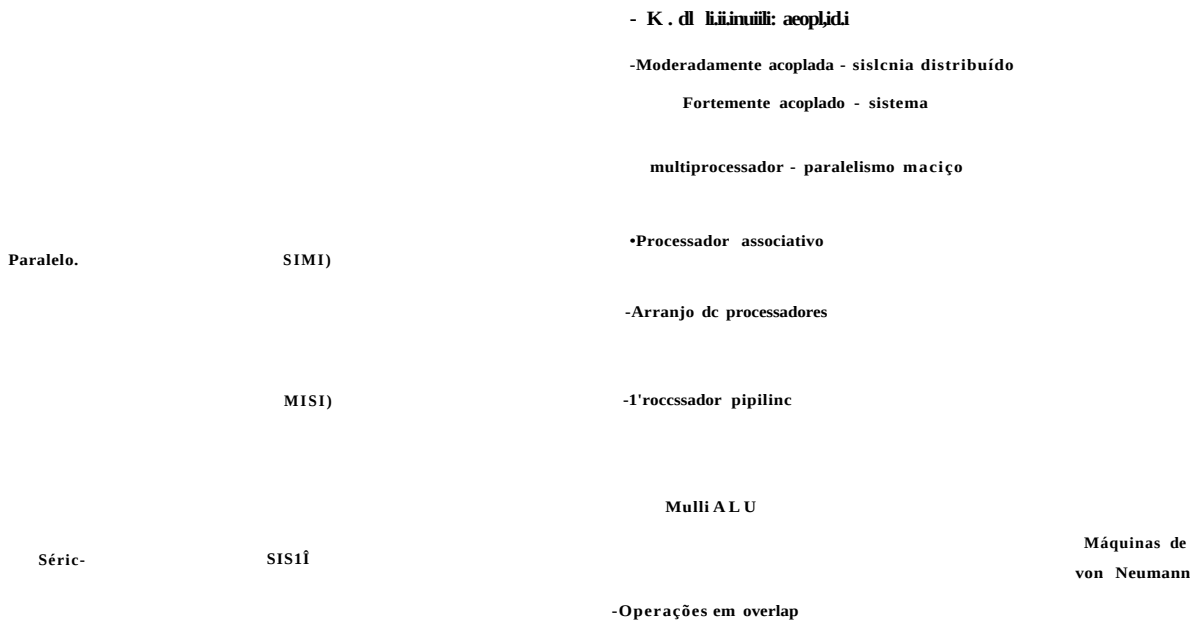
Ao longo do desenvolvimento deste capítulo as referencias que serão utilizadas como roteiro principal são: |Hnslow|, |Rietman|, |Bogni] c [Soucck|, complementando-as todas as outras referencias bibliográficas aqui apresentadas.

Nesse capítulo trataremos desta questão introduzindo conceitos dc melhoria baseadas cm vários fatores tais como: introdução de muitos processadores, pipilining cm arquiteturas convencionais e outras técnicas empregadas cm arquiteturas não-convencionais com o objetivo explícito de obter ganho dc performance, principalmente no que diz respeito à velocidade.

3.2 Taxonomia

Apesar de não existir uma taxonomia única e universal para processamento paralelo, as que foram propostas apresentam pequenas diferenças no que é fundamental e divergem quanto a aspectos mais restritos relativos aos conceitos envolvidos. Nosso objetivo aqui é então apresentar algumas dessas propostas como forma de nos familiarizarmos com o jargão próprio.

Hm [Rietman] é apresentada a seguinte taxonomia de processamento paralelo:



igura 3.2.1 - Taxonomia apresentada por Rietman

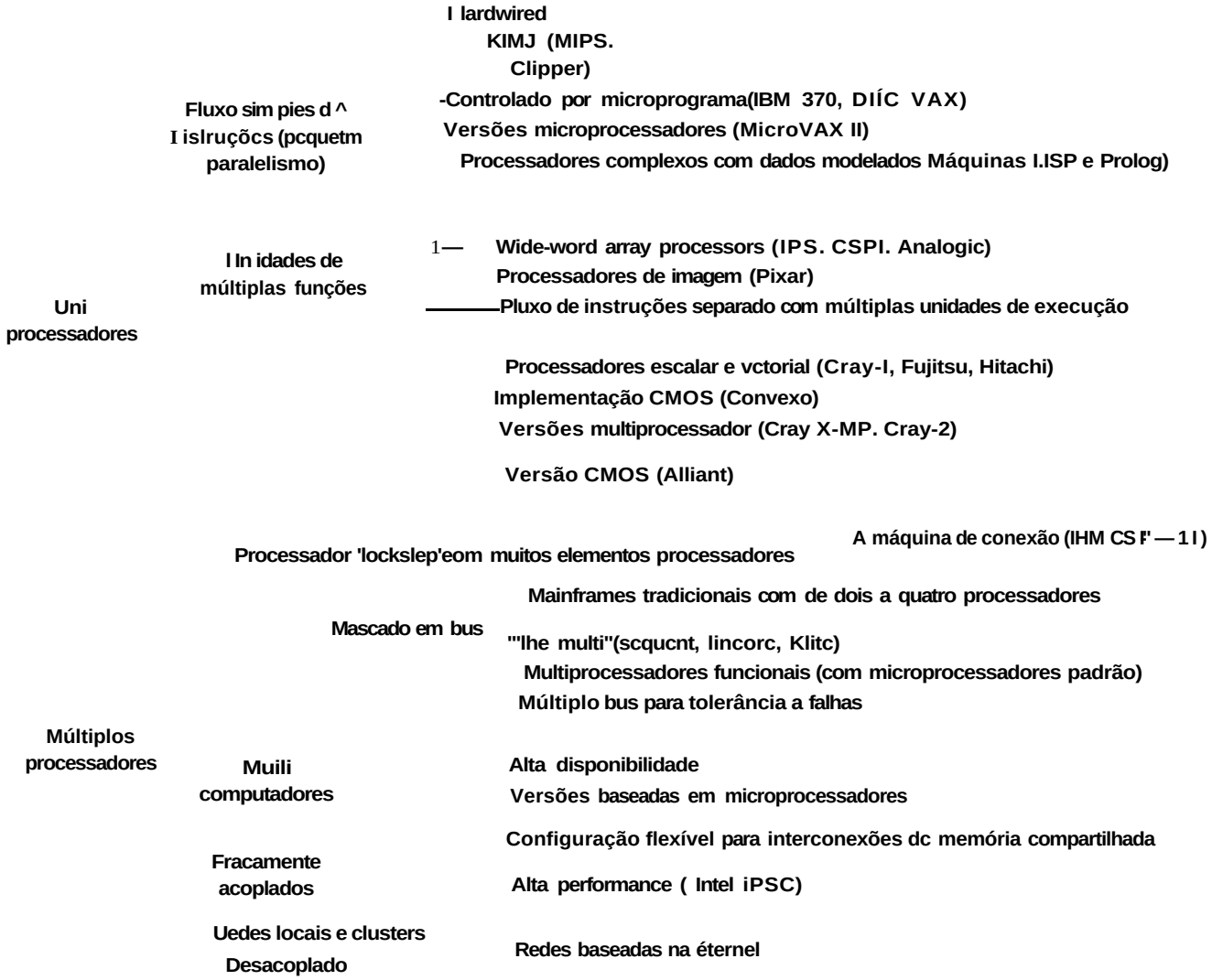
onde, SISD – *Single instruction single data stream*

MISI) - *Multiple instruction single data stream*

SIM I) - *Single instruction multiple data stream*

MI M1) - *Multiple instruction multiple data stream,* essa é uma classificação de sistemas paralelos proposta em 1972.

[Soucek] divide as arquiteturas de computadores em três grupos chamados SI SI), SI MD e Ml MD, e diferentemente dc [Rietman] não considere as arquiteturas MISD, e apresenta a taxonomia da figura 3.2.2 abaixo, muito próxima da anterior.



I-igura 3.2.2 - **Taxonomia cm árvore proposta por C. G. Bell, ver [Soucck|.**

As figuras 3.2.3, 3.2.4 c 3.2.5 abaixo ilustram, respectivamente, máquinas SISD, SIMI) e MISD, segundo [HnslowJ.

Arquitetura SISD é utilizada nas máquinas tradicionais sendo simples c fácil de programar, estão presentes na grande maioria das máquinas comerciais disponíveis, principalmente cm sistemas de pequeno porte e sistemas de medição e controle, pois nestes sistemas as memórias são relativamente pequenas c a eficiência não é vital.

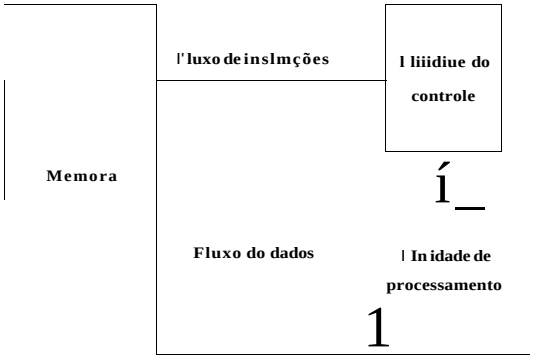


Figura 3.2.3 - Máquina SISD

Arquiteturas SIMI) permite uma utilização mais eficiente do hardware proporcionando operações substancialmente mais rápidas. Um sistema SIMI), tipicamente, consiste de elementos processadores idênticos (IíP), cada processador com sua própria memória, uma rede de interconexão e uma unidade dc controle (UC). A unidade de controle envia instruções por difusão ***(broaJca.sí)*** para todos os processadores, cada processador ativo executa instruções sobre os dados em sua própria memória, as instruções são executadas simultaneamente cm todos os EP's ativos e a rede de interconexões habilita os dados a serem transferidos entre os EP's.

lím função das máquinas SIMI) realizarem operações simultaneamente sobre diferentes itens dc dados, elas são particularmente adequadas para operações sobre matrizes e vetores (processamento de imagens, por exemplo).

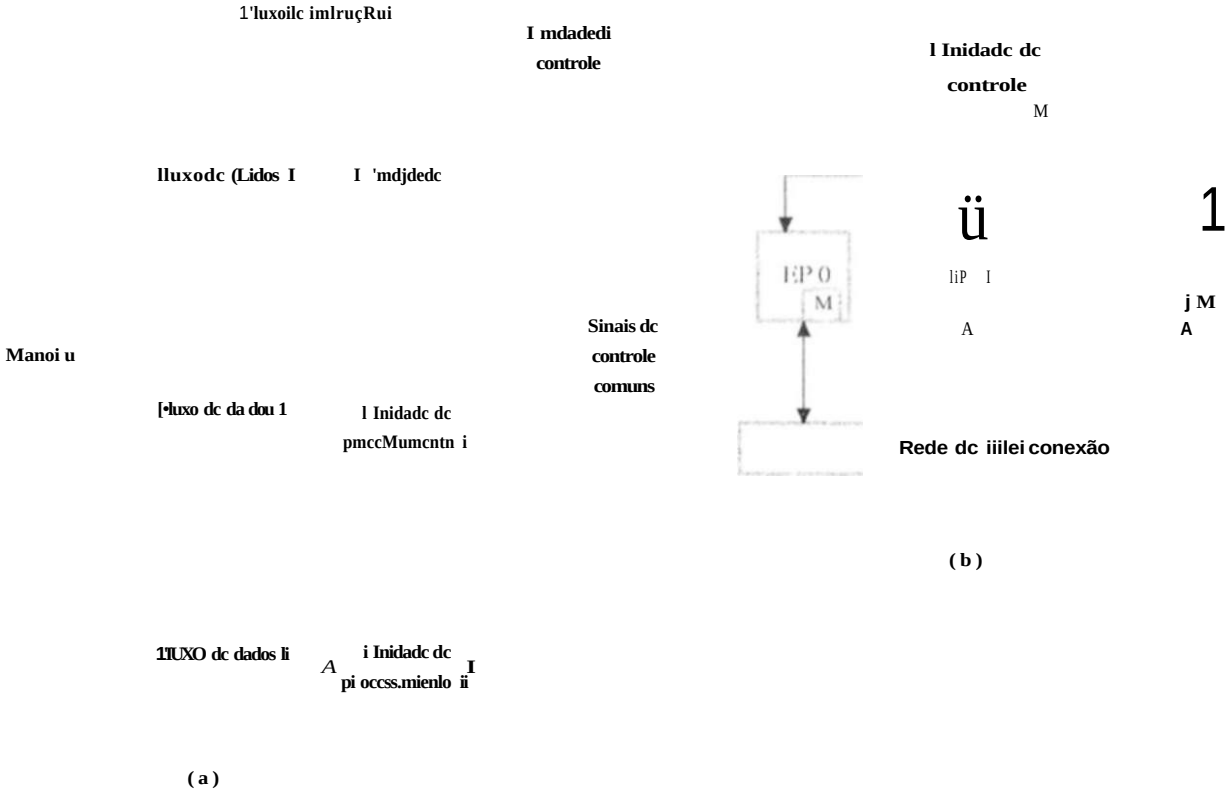


Figura 3.2.4 - Máquina SIMI), (a) apresentada cm |Enslow| e (b) apresentada cm |Soucek|.

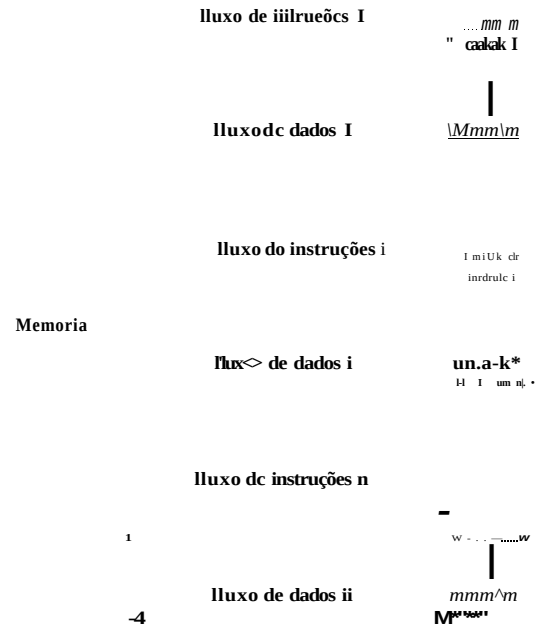


Figura 3.2.5 - Máquina MISD.

Arquitetura MIMI) permite que habilite-se um certo número de programas independentes mas relacionados para execução concorrente. O processamento concorrente é a forma global de paralelismo, denotando operações independentes de uma coleção de atividades computacionais simultâneas, uma máquina concorrente dessa forma utiliza múltiplos processadores fracamente acoplados para realizar muitas operações.

As arquiteturas paralelas de acordo com [Soucekj podem ser divididas em dois grupos de acordo com a forma como os processadores se comunicam: Máquinas baseadas em *BUS* c máquinas não baseadas em *BUS.* Na primeira ainda é possível distinguir claramente entre dois tipos de sub-arquiteturas, quais sejam: sistema fracamente e fortemente acoplados (o primeiro tem múltiplos processadores c memória comum, o segundo tem memória local e as vezes tem memória global para compartilhamento de dados. A principal arquitetura não baseada em *BUS* é a categoria denominada hipercubo que será vista no item **3**.4.**5.3.**

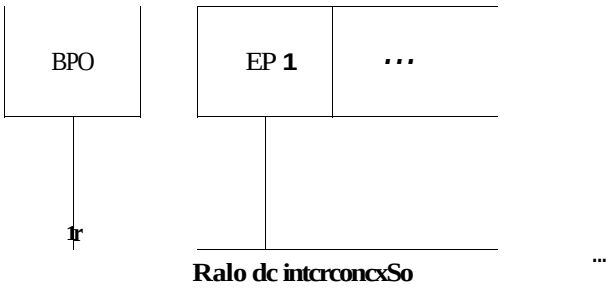


Figura 3.2.Ó - Exemplo dc máquina MIMI) por [Soucek|.

Em **[Bogni]** é apresentada uma taxonomia baseada em níveis de paralelismo com exemplos de máquinas que foram implementadas de tal forma:

Arquitetura	Característica	Sistema
Entrelaçada	Unidade -E única	IBM 3033
Unidades	Escalar	CDC 6600
multifuncionais	Escalar e ***pipeline***	[BM 360/91
Processador adosado	Especializado	AP-1208
	Grande quantidade de unidades	ELI
SIMD	Rede simples de interconexão	[LLIAC IV
	Unidades escalares e vetoriais	BSP
	Sistólicos	PSC
	Processamento de imagens	MPP
M1MD	Comerciais (até 4 CPU's)	IBM 308X
.	Cross-bar	S-l
	Grande quantidade de processadores	Ultracomputer Cm
	Cluster	

Tabela **3.1.3** - Taxonomia baseada cm níveis de paralelismo proposta por [Bogni]

Em [Ilindin] a seguinte taxonomia é apresentada:

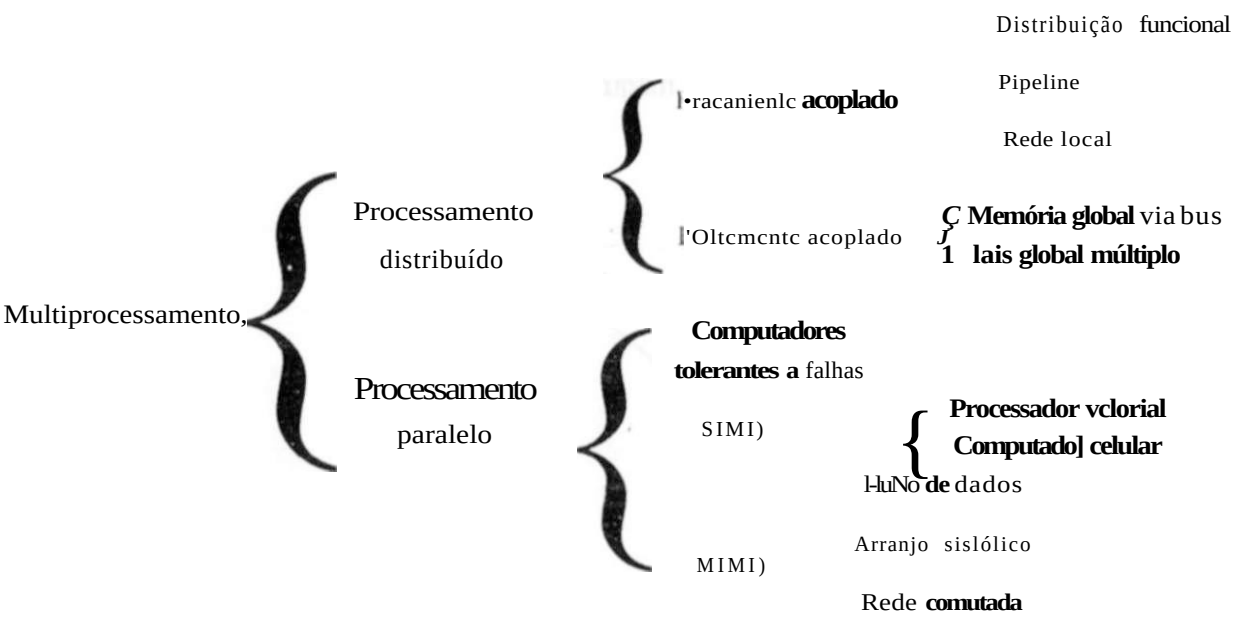


Figura 3.2.7 - Taxonomia proposta por |l iindin|.

Neste trabalho não adotaremos completamente nenhuma dessas taxonomias propostas c procuraremos mostrar as concepções que nos parecem mais importantes, tanto do ponto dc vista comercial/mercadológico, quanto tecnológico e didático ou mais curiosas em termos dc exotismo de concepção.

3.3 Evolução da arquitetura das máquinas monoprocessador

3.3.1 Máquina dc estados finitos

Uma máquina dc estados finitos é um dispositivo que possui tres conjuntos dc valores e uma função dc transição. Um conjunto dc estados E = { c_0, c|,..., e„}, um conjunto dc entradas I = {i_0, ii,...,i,,} c um conjunto de saídas possíveis S = **{ Sn, Si,...,Sk}** c uma função dc transição dada por:

f = E x I

nesse tipo de máquina, em qualquer instante de tempo, podemos afirmar que seu estado é um dos estados e, (l-. A figura 3.3.1.1 abaixo ilustra uma dessas máquinas.

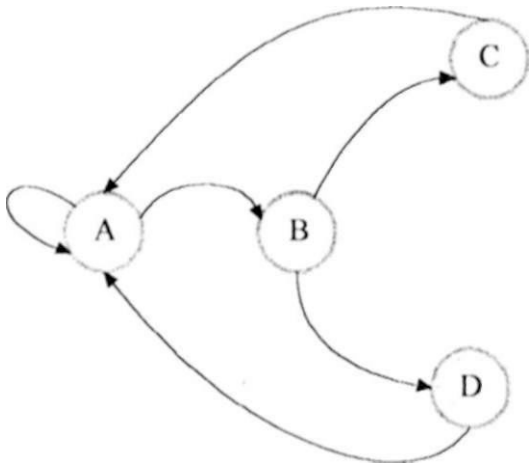


figura .1.3.**1.1** - Exemplo de uma máquina de estados finitos.

Essa figura representa, por exemplo, uma máquina de venda de refrigerantes que dispõe de dois produtos para entrega ao cliente. No estado **A** ela aguarda a introdução da ficha de habilitação, em **B** cia aguarda uma opção do cliente por um dos produtos disponíveis, feita essa opção a máquina irá para um dos estados de liberação do produto (**C** ou **D**) e volta ao estado inicial (**A**) aguardando a próxima ficha. **A** pergunta aqui c: o que essa máquina tem a ver com os computadores digitais? a resposta é: um computador digital é uma máquina de estados finitos reconligurável, isto é, a cada programa que carregamos em sua memoria fazemos com que uma nova função seja assumida do ponto de vista do usuário.

3.3.2 Máquina de Babbagc/von Neumann

A máquina de Babbagc/von Neumann é um dispositivo essencialmente seqüencial que, a partir de um conjunto de dados de entrada gera um conjunto dc dados de saída. **0** esquema básico dessa máquina está mostrado na figura 3.3.2.1 abaixo.

Apesar da grande maioria das máquinas comerciais existentes serem uma variante da máquina de Babbagc /von Neumann com alguma sofisticação, o que se verifica é que essa concepção carrega consigo 0 que se convencionou chamar de gargalo de von Neumann. Esse gargalo impõe um esquema seqüencial de execução dos processamentos a ele submetidos. Como podemos ver todas as operações estão sujeitas a uma atuação da unidade central de processamento, com o intuito de minimizar esse problema algumas soluções tem sido propostas, analisaremos algumas dessas propostas daqui em diante.

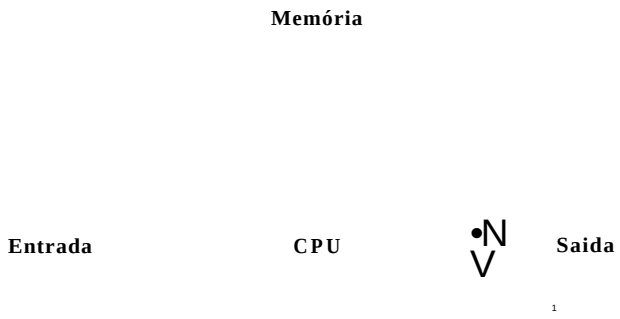


Figura 3.3.2.1 - Esquema básico da máquina de von Neumann

3.3.3 **Acesso direto a memória**

Uma forma dc tentar reduzir o tempo gasto com as operações de l/O foi proposto na metade da década dc 50 com a introdução do DMA (***direct memory access***) que permitiu a transferência direta dc dados entre unidades de entrada e memória e entre memória e unidades de saída utilizando apenas sinais de

controle da CPU de modo a habilitar a unidade de l/O envolvida e a memória, este esquema esta mostrado na figura 3.3.3.1 abaixo.

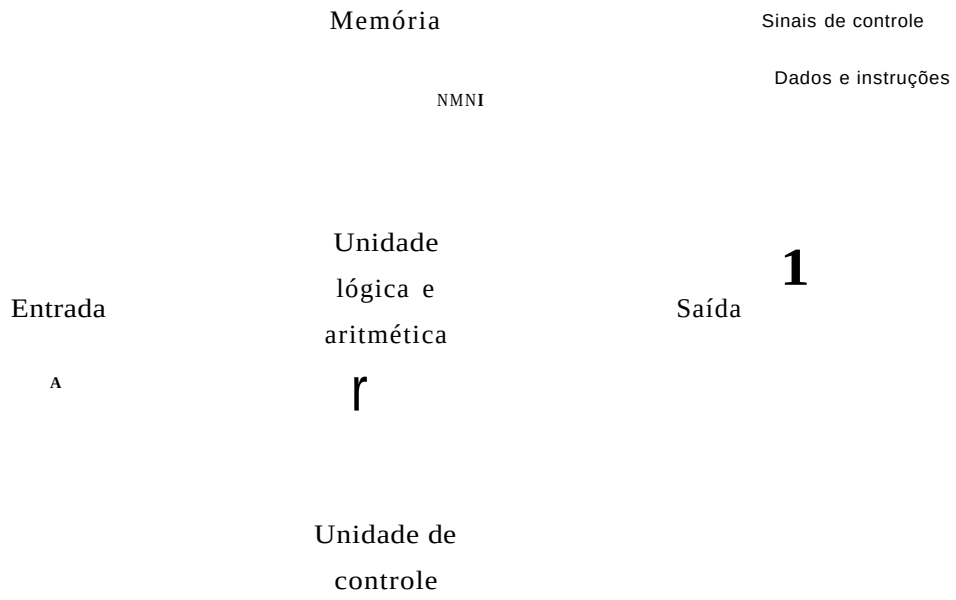


Figura 3.3.3.1 - Esquema básico do uso de DMA

3.3.4 **(anal de** I/**O**

O canal de l/O de início tinha um funcionamento ou semi-independente ou na forma de operações de l/O buflferizadas, isso liberava ainda mais a unidade central de processamento para realizar outras tarefas de forma concorrente. Esse esquema está mostrado na figura 3.3.4.1 abaixo.

Essa nova concepção, com o intuito de aumentar a velocidade dc processamento, foi um evento muito importante nessa escalada dc evolução porque o canal de I/O tornou-se um processador especializado nesse tipo de operação, seu controle era exercido através de um software residente no processador principal, o programa do canal deveria especificar a parcela de memória a ser utilizada, a quantidade de dados a ser transferida, o dispositivo a ser utilizado c a operação a ser executada, o

processador principal ficava livre para executar outras tarefas sofrendo uma interrupção do canal quando a tarefa fosse concluída **ou** algum erro tivesse sido detectado.

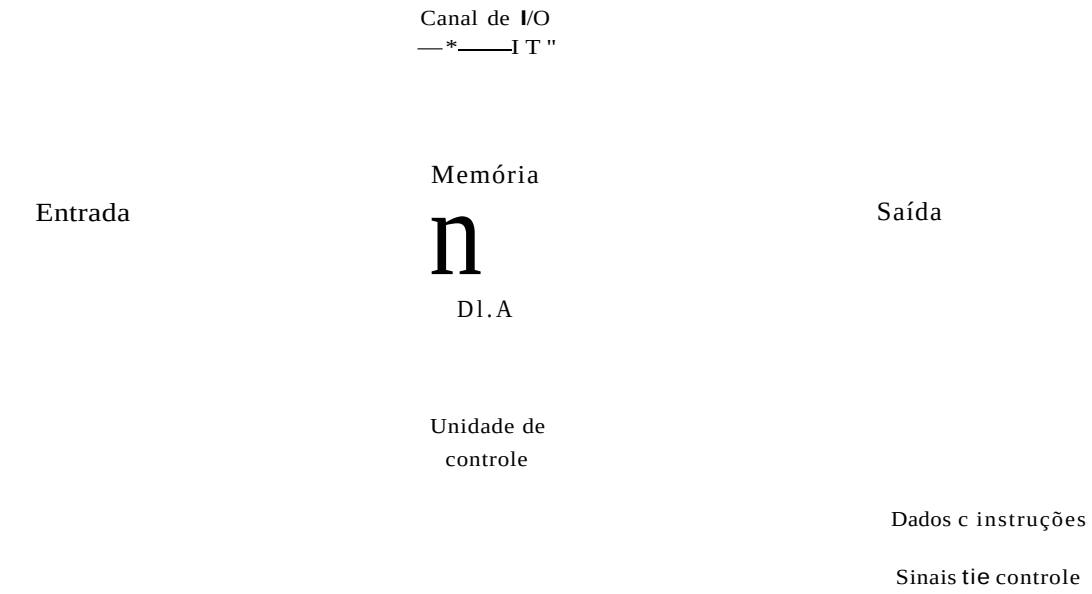


Figura 3.3.4.1 - Esquema utilizando o canal de I/O.

3.3.5 Co-proccssador aritmético

A introdução de processadores especializados que auxiliassem o processador principal cm tarefas especificas já havia sido iniciada com a introdução de um processador especializado em operações dc l/O, então a introdução do co-processador aritmético é uma extensão dessa idéia, o objetivo agora é liberar a CPU das tarefas de processamento de operações aritméticas, permitindo um acréscimo de performance relativa a velocidade de processamento, c algumas tarefas começam ser executadas cm paralelo. Aliado a tudo isso tivemos um desenvolvimento significativo dos sistemas operacionais. O esquema básico dessa arquitetura com co-processador aritmético é mostrado na figura 3.3.5.1 abaixo.

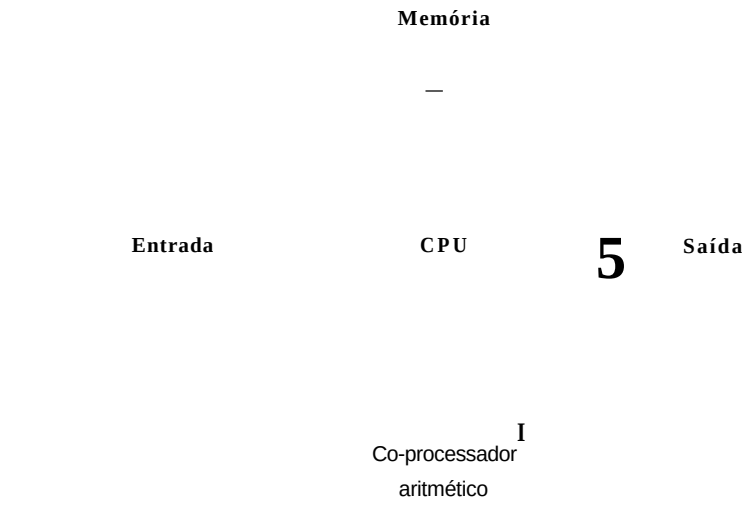


figura 3.3.5.1 - Iisqucma com co-processador aritmético.

3.3.6 ***Pipeline***

O processo de execução de uma instrução cm um computador digital envolve basicamente quatro passos, listes são: busca da instrução na memória ***(fetch),*** dccodificação da instrução para sua identificação ***(insíruclion deaxting),*** busca do operando que será utilizado na execução da instrução ***(operándfelcti)*** e finalmente sua execução. Uma máquina monoprocesso típica realizaria estes quatro passo: de forma seqüencial. Em um pipeline otimiza-se a execução dessas tarefas paralelizando-as, isto é, busca-se a primeira instrução e quando esta estiver sendo decodificada nova busca estará em curso, este es juema está ilustrado na figura 3.3.6.1 abaixo.

r.isM14				li				*k*				*h*
Pauso 3			1.				Í2				*h*	
Passo 2		**1.**				*h*				I_3		
r.i-.su i	**1.**				*h*				I_3			
Passo / Tempo	1	2	3	4	5	6	7	8	9	10	11	12

Figura **3.3.6.**1 (a) - Três ciclos completos realizados seqüencialmente

Pano i				II	**b**	***h***						
Passa 1			1,	l_2	**b**							
Passo 2		1.	***h***	13								
1'asso 1	1.	***h***	1,									
Passo / Tempo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**

Figura **3.3.6.**1 (b) - Três ciclos completos realizados através dc*pipeline*.

Verificamos através de simples inspeção das duas figuras acima que o esquema *pipeline* traz um ganho incremental de tempo considerável, este esquema de racionalização do uso do tempo é utilizado em máquinas de maior número de processadores como um método adicional dc ganho de tempo.

Uma classificação para o processamento pipeline apresentada em |Almeida & Arabe| é a seguinte:

i) ***Pipètining*** aritmético - consiste na segmentação das unidades lógico-aritméticas.

ii) ***Pipètining*** dc instrução - basicamente o esquema mostrado na figura **3.3.6.**1 (a) c (b) acima.

iii) ***Pipètining*** de processadores - processadores especializados arranjados cm série processando uma sequencia de dados.

3.3.7 Máquinas RISC x máquinas CISC

A crescente complexidade das arquiteturas dos computadores, função do avanço da tecnologia dc integração de circuitos, conduziu a uma também crescente complexidade do conjunto de instruções, tendo essa tendência sido agrupada no que se denominou máquinas CISC *{complex inslniction sei computeis).* O que impulsionou essa tendência foi a idéia de que um conjunto de instruções mais poderoso levaria a um processo dc simplificação dos compiladores, além do que conseguiria-se um aumento de velocidade do

processo de programação visto que as instruções de máquina estariam cada vez mais parecidas com uma sentença de linguagem de alto nível, isto é, o assembler mais próximo das linguagens de programação de alto nível. Algumas pesquisas sobre essa tendência e sobre algumas implementações demonstraram que essa ideia era falsa (constatou-se elevado tempo de projeto, aumento da probabilidade de erros e em alguns casos implementações inconsistentes). Além disso, os compiladores ao executar tarefas não recorriam as instruções mais complexas (que eram a razão de ser dessa concepção), ver [Bogni],

Estas instruções complexas eram, em muitos casos, responsáveis pela diminuição dc performance no sistema como um todo, somado a isso a unidade de controle tinha que ser mais complexa que o usual para poder suportá-las (quando microprogramada essa complexidade adicional provoca um aumento de tempo em sua própria execução).

Como alternativa a essa estrutura apareceu a concepção RISC ***{reduced instruction sei computer).*** As hipóteses de construção são: i) um conjunto reduzido de instruções pode projetar uma arquitetura VLSI que faça uso dos recursos de forma mais eficiente que cm CISC; ii) deve-se conseguir uma redução no tempo de execução das instruções.

O projeto RISC' I desenvolvido em Berkeley tinha os seguintes critérios:

1) Executar uma instrução por ciclo de máquina;

2) Iodas as instruções deveriam ter o mesmo tamanho;

3) Acessar a memória com instruções do tipo ***LOAD/STORE;***

4) Suporte dc linguagem dc alto nível.

RISC 1 tem 3 I instruções do tipo lógico-aritméticas, acesso a memória, ***jump*** e controle. As instruções, dados c endereçamento têm 32 bits, a figura 3.3.7.1 abaixo ilustra o formato de instrução.

Figura 3.3.7.1 - Formato de instrução em RISC I.

Como lodo processador moderno RISC **fay.** uso da pré-busca de instruções.

0 quadro 3.3.7.1 abaixo traz uma ilustração comparativa de RISC I com outros projetos RISC, e o quadro 3.3.7.2 uma comparação entre RISC I e outros processadores..

	IBM 801	R1SCI	MIPS
Ano	: 1980	1982	1983
Número de instruções	120	39	55
Tamanho de instrução	32	32	32
Tecnologia	ECL	NMOS	NMOS

Quadro 3.3.7.1 - Comparação de RISC I com outros projetos.

	|iprocessador RISC	^processador 68000	u.processador Z8002
Ano	1982	1980	1979
Instruções práticas	31	61	110
Registros gerais	32	15	14
Modos de endereçamento	2	14	12
Tamanho de endereço	32	24	**16**
Freqüência de relógio	7.5	10	6
Soma reg. a reg (u.s)	0.4	0.4	0.7

Quadro 3.3.7.2 - Comparação de RISC I com outros processadores.

3.4 Evojucão da arquitetura das máíiiiinas multiprocessadores

A crescente necessidade de potência de processamento força empresas, universidades e centros de pesquisas a procurar suprir essa demanda. Duas abordagens basicamente têm sido utilizadas: melhorar o desempenho das máquinas monoprocessador, como visto na seção anterior, a outra é mudar o paradgma de von Neumann. Procuraremos agora mostrar essa segunda abordagem de uma forma evolutiva.

3.4.1 Arranjo de processadores

Um arranjo de processadores c simplesmente uma coleção de elementos processadores com uma distribuição espacial em formato de grade, freqüentemente em formato retangular. A junção de uma linha e uma coluna do arranjo forma um nó, este é constituído dc um processador seqüencial convencional completo com sua própria memória e portas de l/O (serial e/ou paralela) que são responsáveis pelas interconexòes com os outros processadores.

A figura 3.4.1.1 abaixo ilustra um arranjo de processadores que tem um *host* no canto superior esquerdo, esta é uma característica dos arranjos dc processadores, quando não têm um *host,* apresentam um vetor sobre uma margem. Em operação os elementos processadores são sincronizados por um sistema de relógio no processador *host.* Dados transferidos para e do arranjo são simultâneos para evitar conflito. Cada processador executa a mesma instrução sincronizado com os processadores vizinhos, produzindo um vetor ou matriz de dados a partir de um vetor ou matriz dc dados de entrada. Esta é tipicamente uma máquina SI MD.

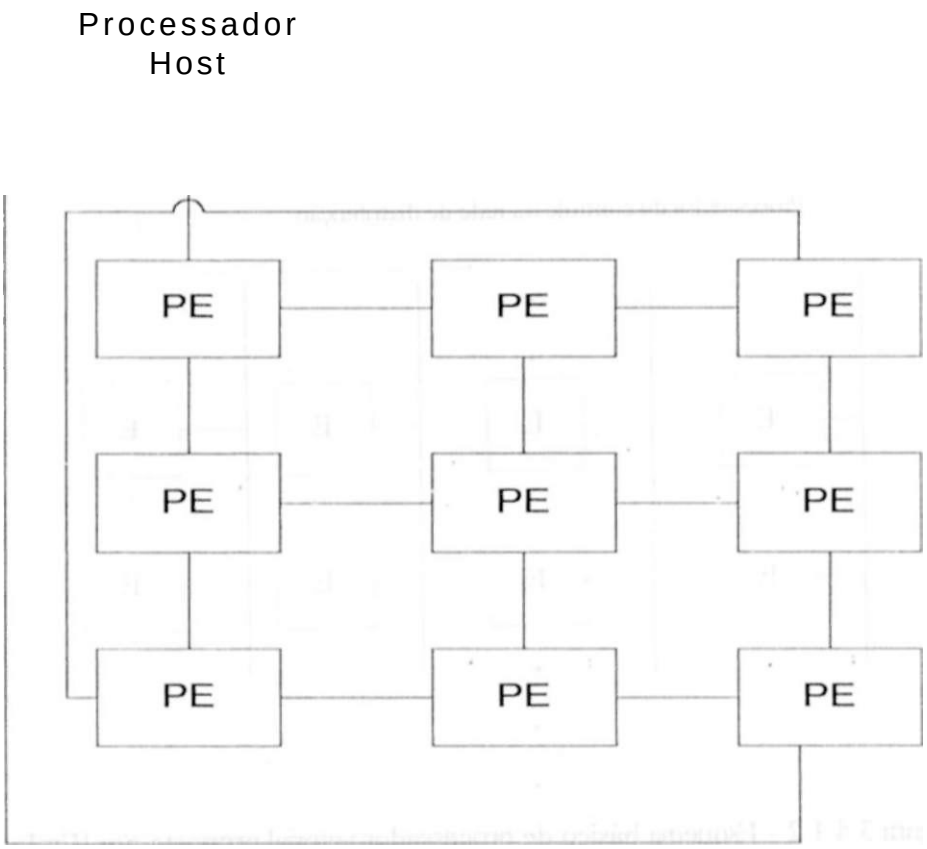


Figura 3.4.1.1 - Arranjo de processadores.

lixiste uma enorme gama de arquiteturas que se utilizam de esquemas mais ou menos iguais a este, essas arquiteturas despertam um grande interesse em universidades c centros de pesquisa de várias empresas, na tentativa dc se conseguir uma arquitetura para máquinas paralelas que tenham uma aceitação universal como a arquitetura de Babbage/von Neumann para máquinas seqüenciais tem levado estas c outras instituições a desencadear uma verdadeira corrida pela obtenção de um padrão de sucesso.

Ainda veremos outras formas de se procurar resolver o problema da concorrência e do paralelismo que tem se apresentado como o grande impeditivo da obtenção de máquinas mais poderosas, tanto do ponto dc vista de velocidade de processamento quanto de preço.

Em j línslow] c apresentada uma configuração básica dc um arranjo de processadores, reproduzida aqui na figura 3.4.1.2 abaixo.

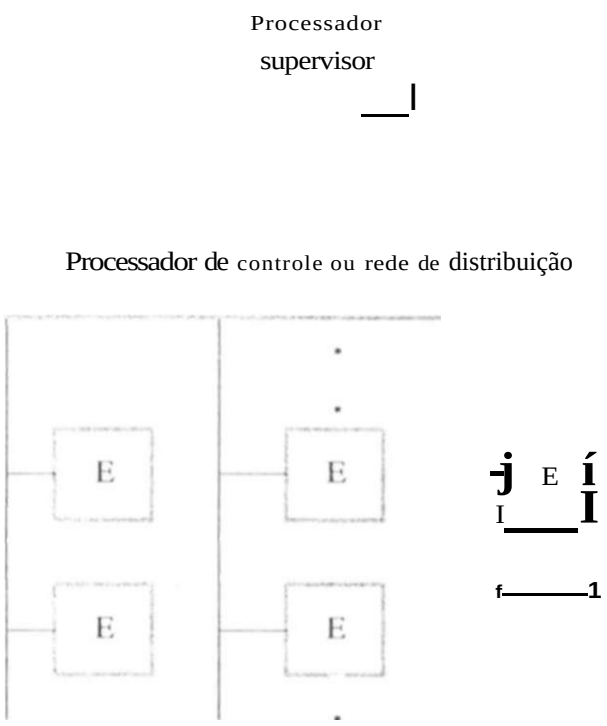


Figura 3.4.1.2 - Hsqucma básico de processador vetorial proposto por |Enslow|.

De acordo com [Bogni] em 1983 os sistemas mais poderosos continham processadores vetoriais pipeline. E seguiam duas filosofias:

- Um conjunto dc unidades funcionais pipeline cada uma delas associada a um algoritmo particular,
- Unidades pipeline multifuncionais.

A partir das primeiras experiências colhidas com os processadores vetoriais determinou-se a necessidade dc contar com uma poderosa unidade de processamento escalar cm adição aos pipelines.

Acontece que no caso de vetores de pequena dimensão e em escalares o tempo de estabelecimento degrada o comportamento do sistema (foram desenvolvidas máquinas com um poderoso conjunto de instruções escalares e vetoriais).

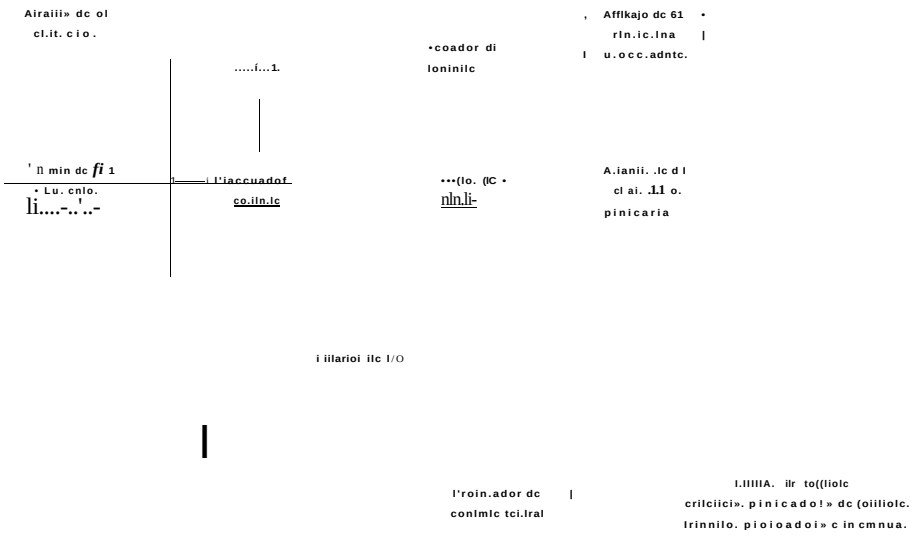
Deste modo consegue-se dar um uso geral e extensivo a estas máquinas (não sendo ainda maior devido ao seu alto custo).

Como os processadores individualmente não tem capacidade para operações independentes, pois não são máquinas completas, a organização aqui apresentada é um sistema paralelo.

0 primeiro trabalho nessa área foi o **SOLOMON** I seguido pelo **SOLOMON** II e este pelo ILLIAC IV. O SOLOMON, em síntese, foi um projeto muito ambicioso, era composto de 1024 processadores ou elementos dc execução sob controle de um processador supervisor. O SOLOMON consistia de um arranjo de 32 x 32 processadores c, como já dito, sob supervisão dc um processador. Cada elemento processador é capaz dc se comunicar com seus quatro vizinhos adjacentes. Os elementos das margens que não possuíam o conjunto completo de vizinhos usavam suas conexões livres para l/O.

O SOLOMON foi criticado por dois motivos: seu tamanho e seu mecanismo relativamente deselegante dc transferência de dados dc um PH para outro aliado a sua inflexibilidade resultante de um local fixo dc armazenamento. A maior diferença entre **SOLOMON** I c II foi a tecnologia utilizada c a velocidade dos elementos processadores.

Como sucessor do SOLOMON apareceu o ILLIAC IV, composto de 256 processadores bem mais poderosos que os utilizados cm SOLOMON, estes processadores foram arranjados de, tal forma que ficamos com 4 quadrantes de 64 processadores cada (arranjo dc 8 x 8) com unidade de controle separada para cada quadrante, este esquema está ilustrado na figura 3.4.1.3 abaixo.



lisura 3.4.1.3 - l.squcm.i lógico do II.I.IAC IV.

O grande problema da exploração comercial deste arranjo é a preparação de programas que utilizem a sua capacidade de processamento e a construção de filtros e tradutores para que se possa migrar (analisar e produzir) soflwarcs existentes em máquinas seqüenciais.

Outros arranjos são possíveis [Rietman] afirma que arranjos com os elementos marginais cm forma dc cilindros, espirais e toros são possíveis, além disso o formato desta conexão depende da aplicação.

Uma outra forma dc montagem é o chamado autômata celular que são modelos em espaço discreto que podem ser usados para modelar qualquer sistema no universo, segundo [Rietman]. O autômata celular foi inventado por John von Neumann nos anos 40, este sistema tem sido utilizado em biologia para modelar sistemas em nível de atividade celular, grupos de células e populações, em química para modelar sistemas moleculares e crescimento dc cristais, em física tem sido utilizado para estudar a interação entre partículas em diversos sistemas dinâmicos, em computação ele tem sido utilizado para modelar sistemas de processamento paralelo, ou máquinas que se auto-reproduzem. Uma máquina autômata celular poderia ser chamada sintetizador universal, ver [Rietman] que dedica o capítulo 5 de seu livro ao estudo dessa abordagem.

3.4.2 O arranjo sistólico

Este c um tipo dc arranjo dc processadores freqüentemente utilizados para operação dc multiplicação vetor-matriz e processamento de imagens (operação de geração da transformada dc I ourier). A figura 3.4.2.1 abaixo ilustra uma operação deste tipo. Um arranjo sistólico c um outro tipo de arranjo dc processadores. 0 poder de computação de uma máquina pode ser classificado de acordo com as seguintes restrições: limite de I/O ou limite dc **CPU.** No primeiro, não existe velocidade suficiente nos dispositivos dc l/O para dar vazão ao lluxo dc dados enviado pela CPU. No segundo, não existe suficiente recursos computacionais para atender a demanda por recursos dos dados na taxa em que estes estão chegando. Uma abordagem que tenta resolver este problema é o arranjo sistólico. Sua concepção é uma extensão do pipeline. Os operandos são bombeados c executados sob um ***while*** c após, bombeados para a saída lista ação de bombeamento é similar aquela do coração c daí a denominação de arranjo sistólico.

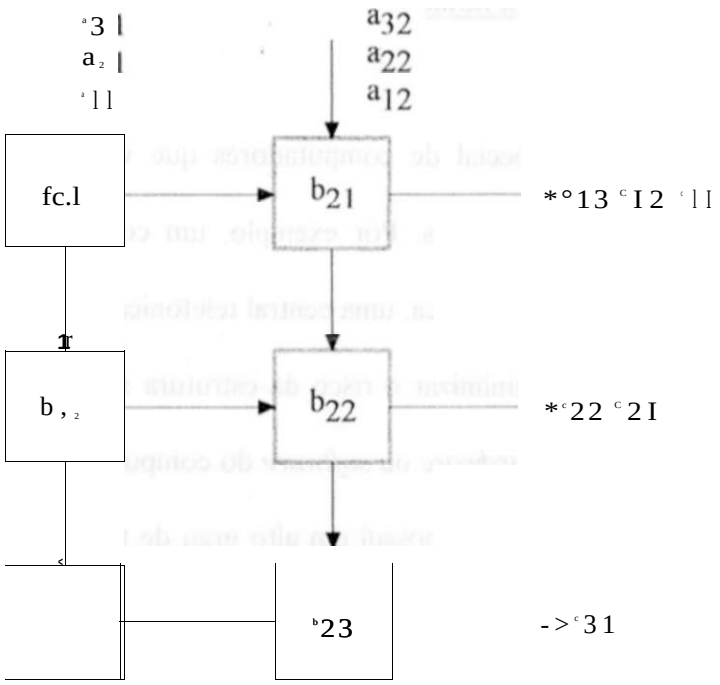


Figura 3.4.2.1 - Arranjo sistólico dc processadores

A figura 3.4.2.1 acima mostra um arranjo sistólico configurado para multiplicação dc matrizes. A matriz dc entrada A é operada sobre a matriz li armazenada no arranjo para gerar a matriz ***C.*** Hm cada ciclo do arranjo uma coluna do produto é gerada.

Hm |Song| é apresentado um algoritmo de reconhecimento de um arranjo bidimensional de processadores com o intuito de torna-lo tolerante a falhas. Resultado do desenvolvimento de um arranjo unidimensional chamado ***Warp*** desenvolvido na universidade ***Carnegie Mellon*** e produzido pela G.E. [Song] aponta dois motivos para o interesse em reconfigurabilidadc:

i) Prover tolerância a falhas;

ii) Aumentar a programabilidadc.

Um interesse adicional é melhorar a sobrevivência do arranjo para aplicações onde reparos manuais não são viáveis e onde há o requisito mínimo dc duração da vida útil do equipamento. Esse algoritmo suporia um modelo geral tanto para o programa como para o arranjo físico de processadores.

3.4.3 Computadores tolerantes a falha

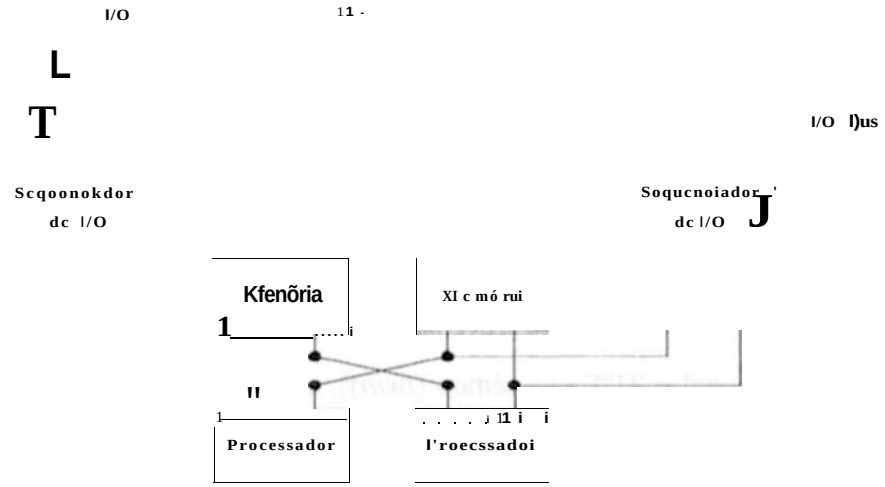
Esta é uma classe especial dc computadores que vem se tornando um item obrigatório na especificação dc determinados projetos. Pôr exemplo, um computador que monitora e comanda uma instalação petrolífera, uma planta química, uma central telefônica por programa armazenado, etc.

A idéia principal aqui é minimizar o risco da estrutura sob comando/monitoração ficar entregue a própria sorte devido a falha dc ***hardware*** ou ***software*** do computador responsável pelo sistema.

Iodo sistema multiprocessador possui um alto grau de tolerância à falha quando comparado a um sistema monoprocessador, contudo, é possível dar ênfase especial aos aspectos da performance do sistema usando elementos redundantes que podem ser duplicados c reconfigurados de forma similar aos sistemas multiprocessadores. Uma quantidade definida de análise de projetos dc engenharia esta envolvida para

determinar a confiabilidade de qualquer configuração, c para isso não existe um padrão de decisão estruturado, cada decisão deriva da forma da configuração a ser examinada.

A figura 3.4.3.1 abaixo ilustra o esquema lógico do S1RU ***(Slrapped-down Imrtial Reference Unit)*** desenvolvido no ***Instrumentation laboratory do MIT.***



Kigura 3.4.3.1 - Projeto tolerante a talha do MIT/I l. SIRIJ.

('orno podemos verificar todas as vias de interconexão, o conjunto processador/memória, o sequenciador são duplicados; isso implica necessariamente cm uma diminuição muito grande da probabilidade de falha total da máquina.

Um outro esquema de computador tolerante a falha é o projeto da central telefônica AXE10B da Eríccson instalada em várias empresas dc telecomunicações do mundo inteiro, lista central foi concebida de forma hierárquica no que diz respeito aos processadores e c construída com dispositivos duplicados trabalhando com divisão de carga ou no sistema ***stand-by*** como mostrado abaixo na figura 3.4.3.2.

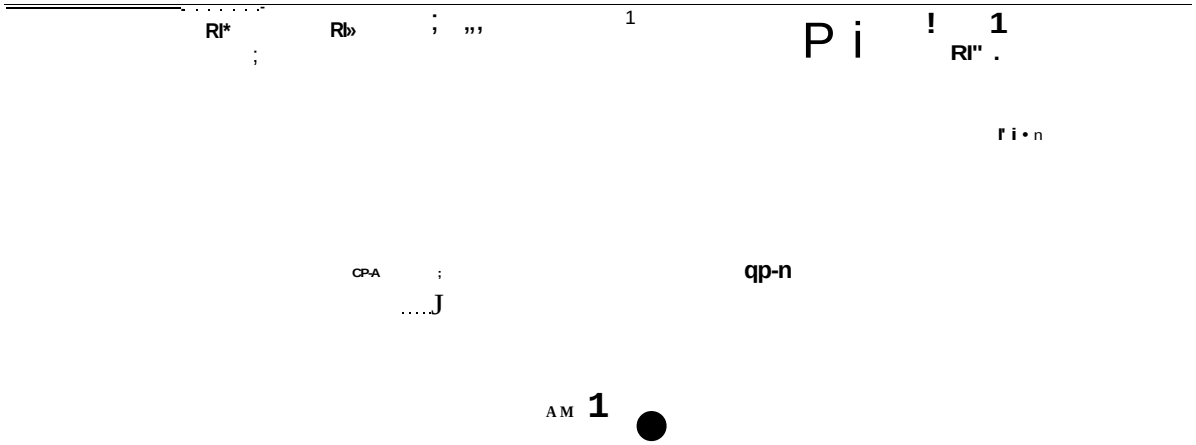


figura 3.4.3.2 - Kxqucma básico do A1V.2 1 I 10 contrai telefônica Kriccson.

Na figura acima RI* representa um processador regional, CP um processador central e AMU é a unidade de manutenção automática.

Os processadores regionais trabalham em divisão de carga, isto é, eles trabalham aos pares (P.P-RPT, onde RP é o processador regional e RPT seu gêmeo (/vW//)), cada processador em condições normais se ocupa da metade dos órgãos que terá de assumir em caso de falha de seu gêmeo. A função destes processadores c, basicamente, ler/escrever de/em pontos de operação, sao responsáveis por tarefas rotineiras e menos nobre na estrutura total de processamento.

Os processadores centrais trabalham juntos executando as mesmas instruções defasados de uma microinstrução, isso permite um esquema de automanutenção que é realizado com a ajuda da AMU ***{aulomalic mailenance uni/)*** Como os dois processadores centrais trabalham em um esquema de ***stand-by,*** eles enviam o resultado de cada execução para a AMU que faz a comparação destes valores se os resultados são iguais a AMU assume que tudo está correndo bem, sc por outro lado os resultados são diferentes a AMU paralisa o processamento de ambos os processadores e envia-lhes um programa para ser executado cujo resultado é conhecido e a partir da resposta dos processadores suspeitos decide quem está funcionando corretamente e quem está defeituoso.

Um outro exemplo deste tipo de máquina está descrito em [Song], ele descreve um arranjo onde são utilizados processadores poderosos e altamente reconfiguráveis ao qual já nos referimos no item anterior.

3.4.4 **Máquinas de fluxo de dados**

listas máquinas em sua maioria são protótipos desenvolvidos em universidades, algumas (nitras somente existem como máquinas simuladas em outros processadores. Existem entretanto implementações comerciais como c o caso do NIiC uPD 7281.

Os computadores convencionais são chamados dc computadores de controle de fluxo porque os programas controlam a operação inteira. Um novo conceito conhecido como computador de fluxo dc dados tem sido difundido. Neste, a execução de uma instrução é habilitada sempre que os operandos estão disponíveis. Cada instrução em uma máquina de fluxo dc dados é implementada com um gabarito que consiste do operando c suas entradas e uma saída. Pôr exemplo, para executar Z 2(X (Y) teríamos uma implementação como a mostrada na figura 3.4.4.1 abaixo.

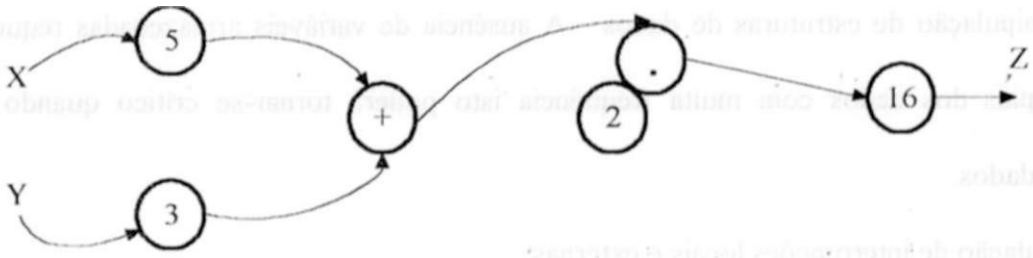


Figura 3.4.4.1 - Grafo de fluxo de dados.

listas idéias são facilmente concretizadas através de circuitos somadores, multiplicadores, etc. em pastilhas VLSI c são simplesmente montadas em um arranjo apropriado para fornecer o resultado desejado

Neste tipo dc arquitetura deve-se considerar com rigor certos mecanismos próprios dela, quais sejam:

- ***Check*** das condições de habilitação dos nodos e detecção de nodos habilitados - Podemos representar uma operação com vários arcos de entrada correspondentes aos operandos. No entanto, um nó

deve estar habilitado antes dc realizar uma operação, isto ocorre quando temos ***tokens*** disponíveis nos arcos de entrada do nó.

- Destinação de nós para processadores - Para que uma operação seja executada c necessário pelo menos um processador, sua escolha deve ser feita com muito critério para não afetar o comportamento da máquina

- Distribuição de ***tokens*** entre nós - Quando uma operação é realizada em um nó como resultado leremos a geração de ***tokens*** que devem ser encaminhados para os nós dependentes daquele.

- Diferenciação dos ***tokens*** - Deve-se diferenciar os ***tokens*** pertencentes a níveis difei entes para que as operações possam ser realizadas de forma correta.

- Comunicação com ambiente externo - as configurações propostas até o momento utilizam o fluxo de dados como um co-processador, então esta comunicação é necessária para receber dados, fornecer resultados e receber o programa a executar.

- Manipulação de estruturas de dados - A ausência de variáveis armazenadas requer que sejam realizadas cópias dos dados com muita freqüência isto poderá tomar-sc crítico quando se trata dc estruturas de dados.

- Manipulação de interrupções locais e externas

- Detecção dc fim de programa

- Administração dc memória

- Administração de processadores

- Comportamento das redes de interconexão, memória c processadores

- Diagnóstico de falhas

As arquiteturas de fluxo de dados podem ser classificadas cm . estáticas e dinâmicas. Na estática os nós dos grafos do programa são carregados na memória antes que o processo se inicie e teremos um único

nível ativo por vez. Uma arquitetura dinâmica permite a execução de vários níveis do nó simultaneamente e além disso estes nós podem ser carregados em tempo de execução.

Mais informações sobre esse tipo de máquina pode ser obtido cm [Rietman], c principalmente em [Bogni] que além de analisa-las do ponto de vista de concepção faz uma análise comparativa c traz um estudo sobre sua programação.

3.4.5 Redes

3.4.5.1 Locais

Este tipo dc construção tem sido muito utilizado atualmente tanto em empresas (automação de escritório, automação industrial), cm universidades c centros de pesquisa cm todo o mundo.

As principais características dessa tecnologia são: normalmente estão restritas a uma instituição (universidade, empresa, etc), estão restritas geograficamente a pequenas e médias distancias (tipicamente alguns poucos quilômetros).

Alguns aspectos devem ser evidenciados aqui, tais como: meios de transmissão típicos, topologias disponíveis c protocolos de acesso ao meio.

Os meios dc transmissão disponíveis são:

- ('abo trançado - composto de dois fios metálicos enrolados cm espiral (para manter constantes as propriedades elétricas garantindo um melhor desempenho, como este tipo dc condutor é muito tradicional cm sistemas telefônicos está bastante disponível, suporta taxas de transmissão da ordem de dezenas dc Mbits/s cm pequenas distâncias, tipicamente algumas centenas dc metros.

- Cabo coaxial - composto de um condutor cilíndrico dentro de um tubo metálico concêntrico que serve a um só tempo como condutor de retorno c como blindagem elclroslática. lambem é uma tecnologia

tradicional c está bastante disponível no mercado, suporta taxas de transmissão da ordem de até algumas centenas de Mbits/s em distancias de alguns milhares de metros (podendo atingir distâncias mirto maiores com a utilização de repetidores).

- Fibra ótica - são suporte para transmissão de luz. infravermelha (faixa de freqüência de 10 - l()·⁵l lz). liste é o meio mais imune a interferências eletromagnéticas, com alta capacidade de transporte de informação, que apresenta ainda alguns inconvenientes quando comparada ao par trançado ou mesmo ao cabo coaxial, por exemplo, as emendas tão simples dc fazer cm par trançado c cabo coaxial ainda não são trivialmente feitas cm fibra ótica. No entanto, permite altas taxas de transmissão (tipicamente centenas de Mbits/s) cobrindo grandes distâncias físicas sem repetidor, pois seu nível de atenuação é muito baixo.

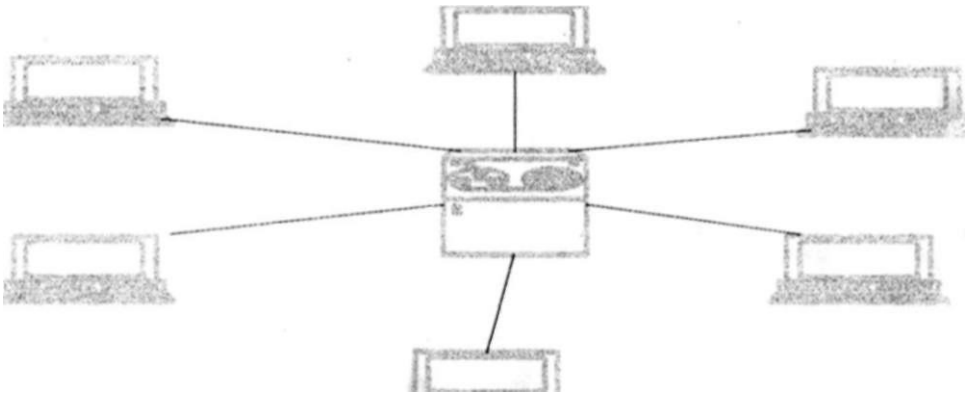
Km lermos dc topologia possível temos tipicamente quatro formas diferentes de compor uma rede local, a saber:

i) Rede cm estrela - este tipo dc estrutura é de concepção bastante simples c apresenta algumas desvantagens que devem ser consideradas quando tivermos que nos decidir por qual topologia optar:

- Presença de um elemento central que compromete a confiabilidade do sistema;

- Necessidade de loteamento dc mensagens;

- Número de estações é limitado pela capacidade do nó central. A figura 3.4.5.1.1 abaixo ilustra esse tipo de construção.



K ı R ii n i ·< S i í i K e d «
B i l r a i * .

ii) Rede cm anel - essa construção c simples no entanto deve-se ter o cuidado dc escolher um protocolo dc acesso ao meio que minimize as deficiências próprias dessa configuração, tais como.

- ***V.m*** caso de falha em um dos nós não permitir interrupção do anel;
- Hm caso de falha a rede poder se reconfigurar automaticamente;
- Interfaces rede/estação simples e baratas.

A figura **3.4.5.1.2** abaixo ilustra o esquema em anel.

iii) Barra comum - este é um dos esquemas mais confiáveis para topologia dc rede pois o fato de uma estação falhar não altera **o** comportamento do resto da rede (isto pode vir a acontecer se escolhermos **o** protocolo dc acesso ao meio dc forma equivocada), alem de que as interfaces para conexão dc uma estação ao barramento são simples c baratas. Nesse tipo de configuração, teoricamente **o** número de estações é ilimitado, porem na prática sabemos que esta limitação existe. A figura **3.4.5 1.3** abaixo ilustra essa montagem.

iv) Malha - esta topologia exige uma capacidade de roleamento e análise de congestionamento em entroncamentos sendo portanto a de mais alto grau de complexidade dc crescimento e-roteamento dc mensagens, sendo dc pouca aplicação em redes locais. A figura **3.4.5.1.4** abaixo ilustra esta construção.

Existem alguns métodos de acesso ao meio físico (protocolo) que já estão bastante difundidos na comunidade dc redes locais, tais como: CSMA-CD, Token ring, TCP/IP, anel com escaninhos, etc. Hsta é uma tecnologia já consolidada c vastamente utilizada como forma dc compartilhar recursos caros e aumentar a velocidade das aplicações.

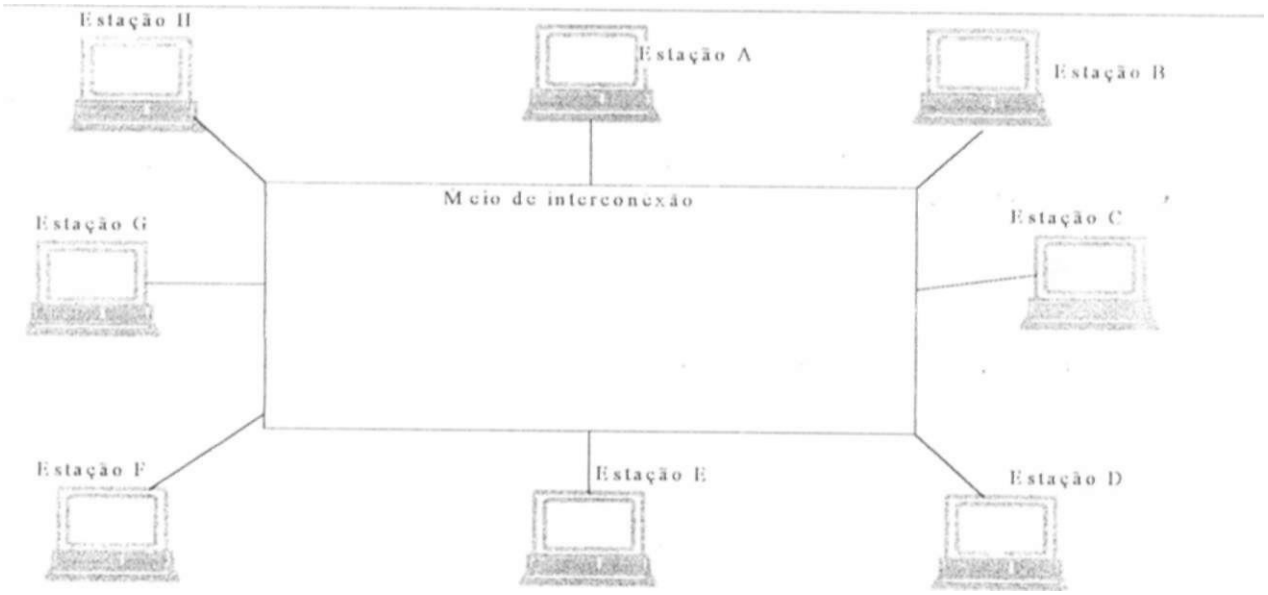
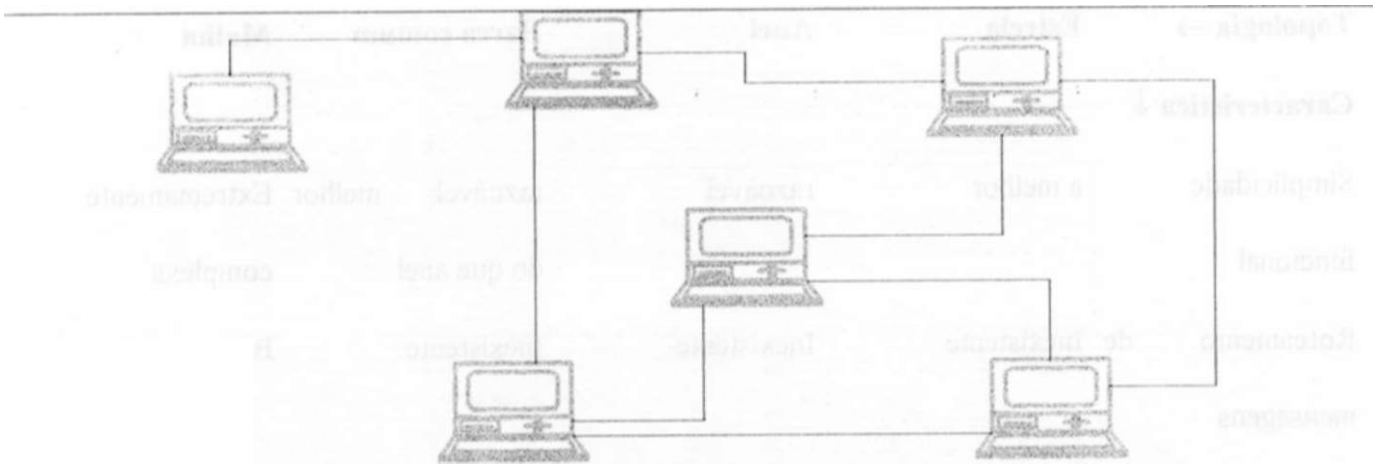


Figura 3.4.5.1.2 - Rcdc cm anel.

Abaixo na tabela 3.4.5.1.1 está a reprodução de uma tabela comparativa das diversas topologias apresentada em [Costa et al. |.

Barramento comum

Figura **3.4.51.3** - Rede em barramento comum.



Kigura 3.4.**5**.1.4 **- Rede om malha**

Abaixo na tabela 3.4.5.1.1 está a reprodução de uma tabela comparativa das diversas topologias apresentada em | Costa ct ai. |.

Topologia -» Característica -*l*	Estrela	**Anel**	Barra comum	Malha
Simplicidade funcional	a melhor	razoável	razoável, melhor do que anel	Extremamente complexa
Roteamento de mensagens	Inexistente	Inexistente	Inexistente	Bastante complexo
Crescimento incremental	Limitado	Alto	Alto	Alto
Aplicação adequada	Envolvendo processamento central de todas as informações	Sem limitação	Sem limitação	Sem limitação
Desempenho	Baixo. Todas as mensagens passam pelo nó central	Alto. Possibilidade de mais de uma mensagem ser transmitida ao mesmo tempo	Mesmo	Alto. Pode se adaptar ao volume de tráfego existente
Confiabilidade	Pouca	Boa, desde que sejam tomados cuidados adicionais	A melhor de todas	Boa, devido a existência de caminhos alternativos
Retardo de transmissão	Médio	Baixo	O mais baixo de todos	Alto

Tabela 3.4.5. I I - Aspectos comparativos das diversas topologias de redes locais.

3.4.5.2 Redes crosshar

A característica principal dc um sistema multiprocessador é o compartilhamento de memória c dispositivos de l/O Quem supre essa demanda por compartilhamento é a rede dc interconexão, existem diferentes formas físicas dc redes dc interconexão, a mais simples c o barramento simples mostrado na figura 3.4.5.1.3 acima. Apesar dessa montagem ser bastante confiável e relativamente barata, ela introduz um componente crítico no sistema que pode causar uma falha total do sistema como resultado de um mal funcionamento de algum circuito dc interface do barramento. Existem duas formas de minimizar essa questão: a primeira c colocar memória c dispositivo dc l/O localmente para cada processador a outra seria diminuir a probabilidade dc ocorrência de paralização total do sistema devido ao problema já apontado, isso é possível duplicando-sc o barramento como já mostrado no caso do RPB da central AXIi na figura 3.4.3.2. Sc o número dc barramento for aumentado cm uma quantidade finita dc unidades a tendência c termos um caminho disponível para cada módulo de memória, isto nos leva a figura 3.4.5.2.1 abaixo.

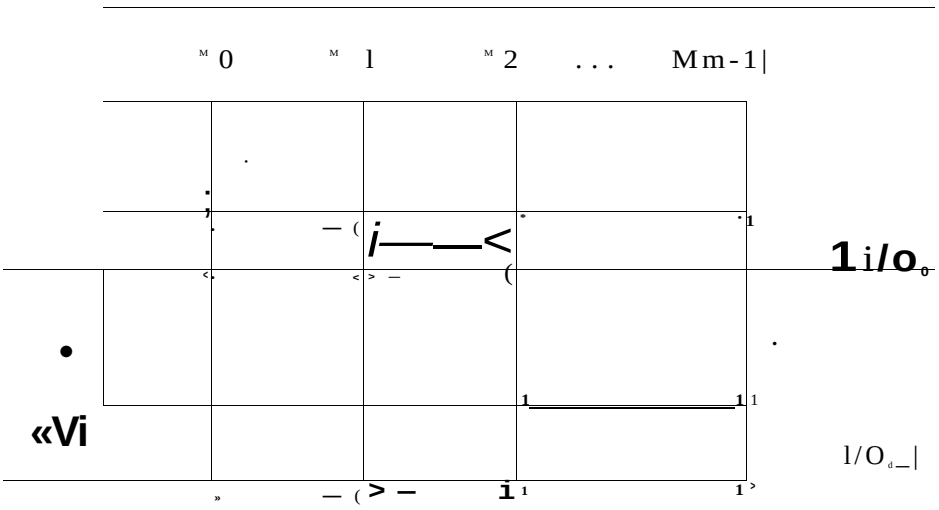


Figura 3.4.5.2.1 - Rede crossbai.

Essa configuração permite acesso simples as portas do sistema pelas unidades funcionais, nessa estrutura qualquer módulo de memória pode se conectar a qualquer processador ou a qualquer unidade de l/O. Uma conexão ***full-lime*** é estabelecida entre as unidades até que toda transferencia seja concluída, esta técnica é conhecida como comutação espacial ***(space-division switching),*** este é um esquema que durante muito tempo foi utilizado em centrais telefônicas e que hoje encontra-se superado.

3.4.**5.3** llipercubos

Essa de acordo com [Soucek] é a principal categoria dc arquitetura não baseada em BUS, em vez disso, os hipercubos se baseiam em canais diretos de acesso a memória entre processadores vizinhos e suas memórias. Cada unidade de processamento chamada nodo, pode se comunicar diretamente com seus vizinhos próximos no espaço n-dimcnsional no qual ele foi projetado e construído. A topologia dc hipercubo foi desenvolvida por Seitz, seguindo conceitos propostos por Sullivan e Brashkow, ver [Soucek]. O hipercubo é um n-cubo binário, também referido como hipercubo binário ou hipercubo booleano.

Um hipercubo dc dimensão 2 tem quatro nodos, cada um num canto de um quadrado. Ciada nodo está habilitado a se comunicar diretamente com dois outros nodos. Um hipercubo tridimensional é mostrado na figura 3.4.5.3.1 abaixo, liste hipercubo tem oito nodos, um em cada canto do cubo, cada nodo sc comunica diretamente com outros três nodos. Cubos de mais alta dimensão são construídos a partir destas estruturas básicas.

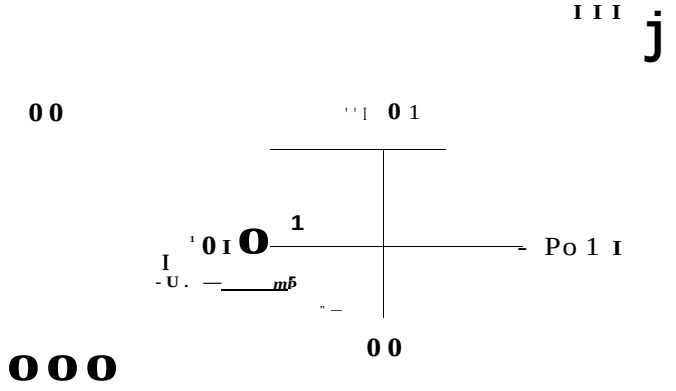


Figura 3.4.5.3.1 - llipcrcubo

Um exemplo dc implementação prática c a família iPSC™. Este c um exemplo dc sistema MIMO com 32, 64 e 128 microcomputadores conectados cm hipercubo, foi projetado c desenvolvido pela INTEL, Cada processador executa uma porção de um grande programa concorrentemente com outros processadores, a concorrência é um paralelismo interativo que permite operações assíncronas dos processadores em um sistema multiprocessadores. Os nodos são conectados por canais dc comunicação **ponto-a-ponto,** e, cada processador é conectado diretamente a um ***hosl*** local (o gerenciador dc hipercubo) através de um canal global de comunicações. **0** gerenciador de cubo é um sistema **microcomputador** INTEL 310 rodando sob **XENIX.** Cada nodo é constituído de um 80286 como processador central, um 80287 como unidade numérica dc ponto flutuante, 512Kbytes de RAM e 64 Kbytes dc PROM, a **figura** 3.4.5.3.2 abaixo ilustra este esquema.

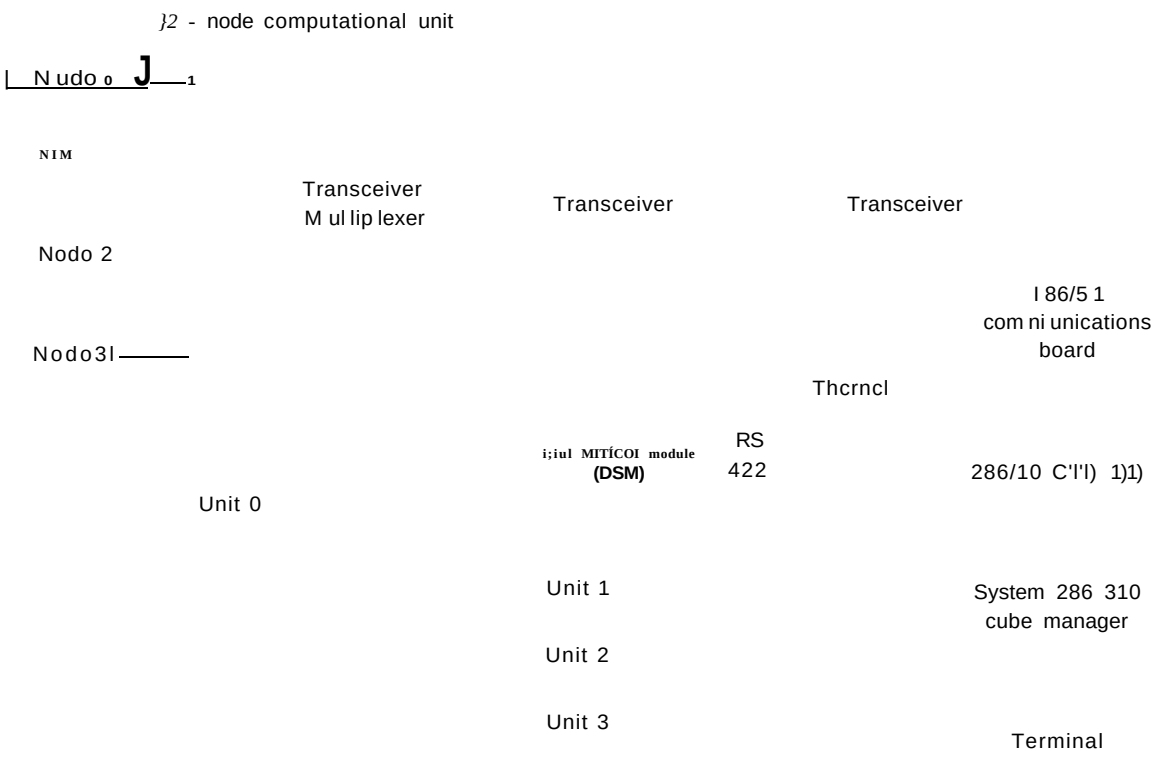


Figura **3**.4.**5.3.2 -** Fxcmplo dc hipcrcubo comercial. iPSC da INTEL

3.4.5.4 Redes neurais

O cérebro humano é um computador pessoal massivamente paralelo baseado em dispositivos orgânicos com lógica dc limiar, listes dispositivos lógicos são conhecidos como neurônios, | Rietman],

Embora apresentem muitas formas diferentes os neurônios tem algumas características cm comum (cada um contém um núcleo localizado numa posição expandida da célula denominada soma - pericário, corpo celular) figura 3.**4**.5.**4.**1.

Dois tipos de processos celulares ou tubos cheios de citoplasma, denominados axônios c dentritos, podem sair do soma. O axônio (fibra nervosa) é em geral liso e raramente se ramifica a não ser na sua terminação, ao passo que os dentritos tem freqüentemente muitas ramificações. Os neurônios dos invertebrados inferiores têm um certo número de processos axonais, razão pela qual são chamados neurônios mullipolarcs. Os neurônios dos vertebrados ou invertebrados superiores têm, em geral, somente

um axônio, embora em alguns neurônios tais como as células amácrimas, encontradas no olho humano, não tenha sido possivcl identificá-lo completamente. A maioria dos neurônios dos vertebrados são, no entanto, multipolares, visto que apresentam vários processos dentríticos.

Quando o neurônio é visto ao microscópio eletrônico, parece estar envolto por uma membrana de aproximadamente 7,5 um de espessura. Há muitas dúvidas a respeito da natureza dessa membrana, infelizmente, pois o conhecimento da sua estrutura molecular seria de grande importância para a compreensão da base fisico-química da transmissão de informação nas células nervosas, uma vez que a membrana está muito envolvida neste processo de transmissão.

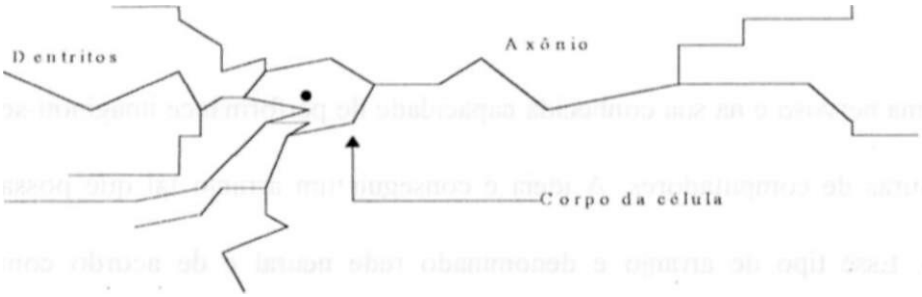
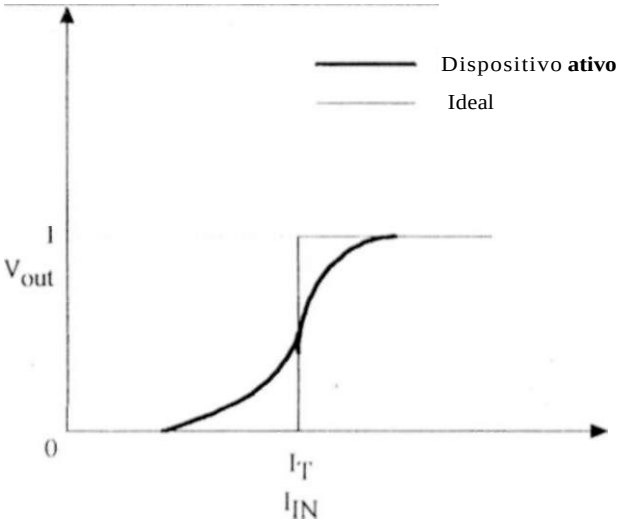


figura 3.4.5.4.1 - Um neurônio básico.

Kxislem mais de 1000 conexões de dentritos para um neurônio médio no cérebro humano. O neurônio é uma célula altamente especializada, que desenvolveu enormemente as propriedades de irritabilidade c condutibilidade. O termo sinapse tem sido aplicado a região de contato ou continuidade entre dois neurônios, entre um receptor sensitivo e um neurônio, c entre um neurônio e um efetor, ver [USHERWOOD].

Um neurônio pode ter conexões excitatória ou inibitória, a primeira dispara o neurônio c a segunda inibe o disparo. Km um certo nível limiar o neurônio irá disparar e abaixo desse nível não disparará. Os sinais dc entrada são somados no neurônio.



igura 3.4.5.4.1 - **Curva dc transferencia do dispositivo dc lógica dc limiar.**

Com base nessa visão dc sistema nervoso c na sua conhecida capacidade de performance imaginou-se uma variedade enorme de arquiteturas de computadores. A idéia é conseguir um arranjo tal que possa emular/simular o sistema nervoso. Esse tipo de arranjo é denominado rede neural e de acordo com [Soucek] formam a sexta geração de computadores.

[Soucek] situa as seguintes categorias de máquinas na sexta geração de computadores:

- Sistemas adaptativos e aprendizes
- Computadores neurais
- Sistemas neurais artificiais
- Algoritmos dc mutação genética baseados em sistemas especialistas adaptativos
- Processadores e memórias associativas
- Sistemas pseudo-associativos c nebulosos
- Hipercubos
- Sistemas concorrentes

As pesquisas nesta área são multidisciplinares e integradas envolvendo engenharia e ciência dos computadores, fisiologia, linguística, lógica, psicologia, etc.

Verificamos na figura **3.3.2.1** que as duas partes principais da máquina dc von Neumann (memória e unidade central dc processamento) devem ser otimizadas cm uso com o objetivo dc conseguirmos melhoria dc performance. A CPU ou lê ou escreve em uma determinada locação de memória a cada vez, isto significa que a CPU está ocupada todo o tempo entretanto a maior parle da memória está livre, então uma forma óbvia dc melhor utilizar o hardware é uma arquitetura baseada num grande número de unidades de processamento trabalhando cm paralelo, ver [Soucekj.

("orno já dito anteriormente diversas arquiteturas têm sido proposta, isso ao mesmo tempo que demonstra o vigor e a confiança dos pesquisadores de empresas e universidades nesse campo de pesquisa c aplicação nos coloca frente a uma verdadeira babel de conceitos e esquemas, desse modo nos limitai emos nesse trabalho a comentar o estado da arte.

O fato das pesquisas nesta área serem multidisciplinares nos leva a perceber necessidades onde antes não enxergávamos, por exemplo, as áreas de reconhecimento de voz c imagens são relativamente novas no contexto da ciência da computação, tendo-se tornado interessante do ponto de vista prático, a partir do momento que as máquinas passaram a ser dotadas de capacidade (de memória e de processamento) suficientes para permitir um custo razoável para produzi-las.

O aprendizado em redes neurais é visto como um mecanismo para procura dc um mínimo dc uma função critério multidimensional ou função erro.

Alguns resultados dc mecânica estatística são utilizados para o aprendizado em redes estocásticas recorrentes, o aprendizado de Boltzman. Bstas redes consistem de n unidades estocásticas arbitrariamente intcrconccladas onde o estado **Xj** da i-ésima unidade é ou **1** ou - I. uma máquina dc Boltzman deste modo é uma rede estocástica estável com capacidade de alcançar o mínimo global exigido.

Alguns sistemas foram projetados para emular o comportamento humano. A psicologia cognitiva tem desenvolvido modelos do comportamento humano, percepção e aprendizado c tem ligado isso com a

psicologia do cérebro. |Soucek| afirma que muitos problemas têm de ser resolvidos para a consolidação do projeto da sexta geração dc computadores, tais como:

I) Hntender melhor a natureza do entendimento. As pessoas integram conhecimento para entender objetos e idéias, recordam exemplos típicos a partir de suas experiências, e geram analogias pela modificação da experiência através da inferência dedutiva. Os meios pelos quais as pessoas prevêem mal funcionamento, iluminação mental, uso criativo da cognição e linguagem devem ser entendidas.

2) Modelagem de funções inteligentes. Devc-sc construir modelos para percepção visual, significado linguístico, memória do passado ***(long íerm memory),*** aprendizado e desenvolvimento cognitivo.

3) Investigar a interface homem-máquina em termos de ciência cognitiva.

4) Integração da pesquisa psicológica com a pesquisa fisiológica.

Outros sistemas estão sendo projetados para emular funções de comportamento do cérebro. Um grande esforço está sendo realizado sob a forma dc computadores influenciados por modelos e teorias do cérebro (em termos de arquitetura, organização, interfuncionamento, etc).

Uma outra vertente de pesquisas devida ao desenvolvimento na psicologia, fisiologia e tecnologia de computadores (surgindo cm nível de mercado e aplicações) inclui sistemas para saúde e satisfação, máquinas para leitura da mente (no sentido de leitura de alguns ou vários sinais em aplicações de neurobiologia c comportamento), interfaces homem-máquina, órgãos artificiais, analisadores, estimuladores e simuladores para uso tanto profissional quanto doméstico, ver [Soucek J e [Rietman],

3.5 Problemas clássicos de concorrência

Os problemas de concorrência são de grande importância no projeto de máquinas paralelas pois nestas máquinas algumas ou muitas tarefas têm que ser executadas dc forma concorrente, ou não podem

ser executadas simultaneamente e o projeto deve considerar estas peculiaridades e solucionar os problemas decorrentes dc tais necessidades a referencia aqui c [Peterson].

3.5.1 Exclusão mútua

Imagine que alguns processos compartilham variáveis, registros, arquivos ou outro item qualquer. Listes itens dc dados compartilhados serão utilizados pelos processos de alguma forma, por exemplo, para ler um valor ou para escrever um novo valor. Isto significa que a atualização dos itens dc dados compartilhados devem primeiro ser lidos, computar o novo valor c finalmente escrever o novo valor em seu lugar. O grande problema ocorrerá se dois processos tentarem executar esta seqüência de instruções ao mesmo tempo. A seguinte seqüência pode ocorrer:

1) **0** primeiro processo lê o valor x no objeto compartilhado

2) O segundo processo lê o valor x no objeto compartilhado

3) O primeiro processo computa um valor atualizado x' = f(x)

4) O segundo processo computa um valor atualizado x" = g(x)

5) O primeiro processo escreve x' no local compartilhado

6) **0** segundo processo escreve x" no local compartilhado destruindo o valor x\

O efeito da computação do primeiro processo foi perdido, pois agora o valor do objeto compartilhado é g(x), enquanto ele deveria ser g(f(x)) ou f{g(x)) (imagine esta operação em um sistema de compensação bancária onde a primeira operação é de retirada, por exemplo de 11$ 1000,00, e a segunda é dc depósito, de por exemplo R$ 1000000,00), um desastre!!!

Para prevenir este tipo de problema é necessário prover um mecanismo para não permitir que dois ou mais processos realizem a mesma seqüência de operações simultaneamente sobre objetos compartilhados. Exclusão mútua é uma técnica utilizada para definir uma codificação dc entrada e saída de tal forma que

não mais que um processo acesse **um** objeto (dado) compartilhado ao mesmo tempo. **A** codificação que acessa o objeto compartilhado c precisa de proteção para evitar a interferência dc outros processos c chamada de seção crítica. **A** ideia c que quando um processo está executando sua seção crítica, ele primeiro espera até o outro processo não estar executando sua própria seção crítica, então ele fecha o acesso para a seção crítica, prevendo que qualquer outro processo possa tentar entrar em sua seção crítica, tile é admitido na seção crítica, exceula-a e no final abre a seção para permitir que outro processo possa aeessa-lo.

Este problema está modelado usando rede de Petri na figura 3.5.1.1 abaixo. O lugar m representa a permissão (codificação) para entrar na seção crítica. Para um processo entrar na seção crítica ele deve ter um ***loken*** em pi ou D2, c uma permissão também deve estar presente em m. Sc os dois processos têm interesse cm acessar a seção crítica simultaneamente as transições h e l_z estão em condito e somente uma delas irá disparar. Disparando ti desabilitamos o disparo dc t? c vice-versa.

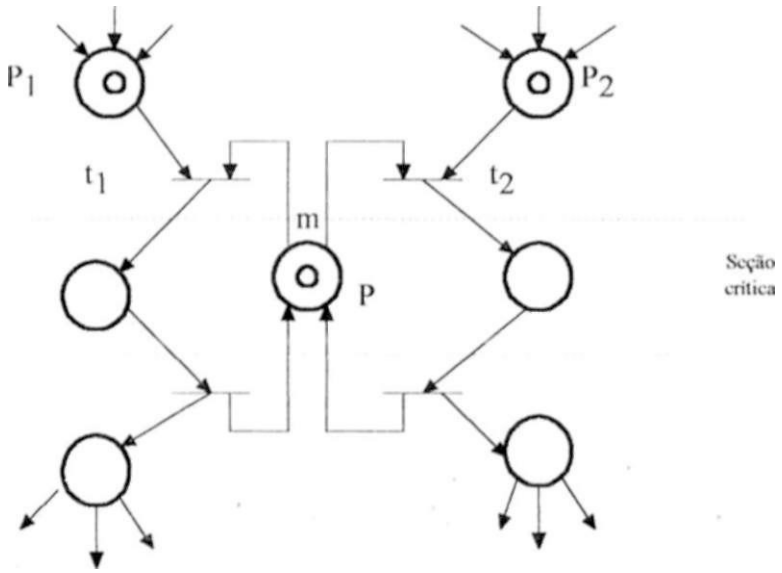


Figura 3.5.1.1 - O problema da exclusão mútua.