



Cálculo Numérico
assunto desta lista
T. Praciano-Pereira

alun@:

Univ. Estadual Vale do Acaraú	10 de maio de 2009
página da disciplina	www.calculo-numerico.sobralmatematica.org
Documento produzido com L ^A T _E X	sis. op. Debian/Gnu/Linux

Lista 05
tarcisio@member.ams.org
Dep. de Computação

0.1 Informações

Por favor, se você usar o método antigo para entrega desta lista, em papel, prenda esta *folha de rosto* na solução, preenchendo com os seus dados, ela será usada na correção.

Se você quiser entregar o trabalho eletronicamente, envie o arquivo para o meu e-mail ou entregue em CD na secretária do Curso de Computação.

Por favor, siga as instruções sobre nomes de arquivos, leia as intruções na página da disciplina.

Data da entrega da lista: dia 18 de Maio, segunda-feira.

Se o trabalho for feito em equipe, basta um único trabalho ser entregue e neste caso, no cabeçalho, devem estar os nomes completos de tod@s @s alun@s junto com os seus respectivos e-mails. O número de membros de uma equipe não devem ultrapassar três.

0.1.1 Objetivo

Esta lista está baseada no programa `raizes.cc` e conseqüentemente na biblioteca `raizes.h`.

Palavras chave raizes de funções polinomiais, busca binária, método da tangente e da secante.

0.1.2 Avaliação do trabalho

Leia na página da disciplina a este respeito.

Acrescente estas perguntas como última questão do trabalho, ela não será avaliada, mas será usada na correção do planejamento.

- Você encontrou alguma coisa interessante no trabalho ? indique qual.
- Do ponto de vista de “objetividade”, você tem alguma crítica quanto à estrutura do trabalho ? especifique.
- Quando eu elaborar a correção, quais são os itens que você gostaria que eu discutisse de forma mais cuidadosa, dentre as questões do trabalho.

0.2 Exercícios

1. A função `main()` no programa `raizes.cc` é discutida nesta questão. Compile e rode o programa para avaliar o seu funcionamento e leia o programa para responder as questões. Ao rodar o programa pela primeira vez, responda as questões com “enter” para usar os dados pre-definidos.

- (V)[](F)[] A função `main()` executa as seguintes etapas
 - Anuncia o programa;
 - Recebe os dados da malha;
 - chama a função `varredura()`
 - apresenta o relatório final: `relatorio_final()`.
- (V)[](F)[] A função `main()` indiretamente calcula as raízes de uma função, chamando a função `varredura()`.
- (V)[](F)[] A função `main()` diretamente calcula as raízes de uma função.
- (V)[](F)[] Na função `main()` são declaradas três variáveis, do tipo `malha`, do tipo `resultado` e do tipo `int`
- (V)[](F)[] Na função `main()` estão definidas três tipos novos de dados.
- (V)[](F)[] A função `varredura()` devolve um inteiro para `main()` que representa a quantidade de raízes encontradas.

2. A função `varredura()` leia esta função, ela se definida em `raizes.h`.

- (V)[](F)[] A função `varredura()` recebe um número do tipo `float` e o uso como passo para percorrer a malha selecionada.
- (V)[](F)[] A função `varredura()` recebe um dado tipo malha que tem três campos representando o início e o fim de um intervalo e o passo da malha definida neste intervalo.
- (V)[](F)[] A função `varredura()` executa o loop principal do programa percorrendo a malha com o passo selecionado. A cada passo da malha ela faz dois testes
 - verifica se f troca de sinal;
 - verifica se f' troca de sinal;
- (V)[](F)[] A função `varredura()` é que seleciona um dos métodos, *da tangente*, *da secante* ou *da busca binária*.
- (V)[](F)[] A função `varredura()` chama a função `seleciona_metodo()` e é esta que seleciona um dos métodos, *da tangente*, *da secante* ou *da busca binária*.

3. A função `raiz_tangente()`

- (a) $(V)[\](F)[\]$ Esta função usa a expressão do quociente de diferença de segunda ordem para testar se o gráfico de uma função é côncavo ou convexo.
- (b) $(V)[\](F)[\]$ Se o gráfico for convexo, semelhante a uma parábola com abertura para baixo, e crescente a função considera a reta tangente no primeiro extremo do intervalo, se for convexo e decrescente considera a tangente no segundo extremo do intervalo.
- (c) $(V)[\](F)[\]$ Se o gráfico for côncavo, semelhante a uma parábola com abertura para cima, e decrescente a função considera a reta tangente no primeiro extremo do intervalo, se for côncavo e decrescente considera a tangente no segundo extremo do intervalo.
- (d) $(V)[\](F)[\]$ Em todos os casos, esta função verifica duas condições, *se o módulo de f é pequeno e se um teto máximo de operações foi executado.*
4. raiz_secante() Esta função recebe duas variáveis, uma do tipo `malha` e outra do tipo inteiro. Ela recebe um sub-intervalo com a malha fina associada ao mesmo onde vai testar se existe raiz do tipo secante (com troca de sinal).
5. (a) $(V)[\](F)[\]$ Não há teto de operações, ela calcula indefinidamente a raiz pelo método da secante.
- (b) $(V)[\](F)[\]$ Há um teto de operações que é utilizado na expressão `teto*(fabsf(f(c)) > delta)`.
- (c) $(V)[\](F)[\]$ A expressão `teto*(fabsf(f(c)) > delta)` é a multiplicação de um número inteiro por uma desigualdade, portanto um produto absurdo.
- (d) $(V)[\](F)[\]$ A expressão `teto*(fabsf(f(c)) > delta)` é a multiplicação de um número inteiro pela avaliação de uma desigualdade, portanto o produto absurdo de dois inteiros um dos quais pode ser zero que representa o falso em computação coma linguagem C.
- (e) $(V)[\](F)[\]$ O trecho de programa
- ```
if (f(a)*f(c) < 0) b = c;
else a = c;
```
- serve para selecionar se há uma troca de sinal no intervalo  $[a, c]$  ou no intervalo  $[c, b]$
- (f)  $(V)[\ ](F)[\ ]$  A declaração
- ```
ofstream dados;
```
- serve para criar uma variável do tipo arquivo para entrada de dados do programa. Ela será usada em
- ```
dados << c << ' ' << f(c) << endl;
```
- onde faz o registra de uma raiz encontrada

- (g)  $(V)[\ ](F)[\ ]$  A declaração
- ```
ofstream dados;
```
- serve para criar uma variável do tipo arquivo para saída de dados do programa. Ela será usada em
- ```
dados << c << ' ' << f(c) << endl;
```
- onde faz o registra de uma raiz encontrada

#### 6. A função `raiz_binaria()`

```
resultado raiz_binaria (malha mal, int teto)
{
 ofstream dados; // figura
 resultado data = {mal.a, 'método da busca binária'};
 float a=mal.a,b=mal.b,delta=mal.delta;
 float c = (a + b)/2.0; // ponto médio do intervalo
 // se teto==0, para! , se |f(x)| < delta, para!
 dados.open('dados',ios::app); // figura
 while((teto*(fabsf(f(c)) > delta)))
 { // teto=iterações delta é a precisão
 if (f(a)*f(c) < 0)
 b = c;
 else
 a = c; // descobre intervalo de troca de sinal
 // c = (a + b)/2.0; // aqui tem um erro
 teto--; // desconta o número de iterações se for zero, para
 dados << c << ' ' << f(c) << endl; // figura
 }
 dados << c << ' ' << f(c) << endl; // figura
 dados.close(); // figura
 data.raiz = c;
 return(data);
};
```

A programadora observou que esta função estava errada e marcou o erro com um comentário “aqui tem um erro”, no programa não havia esta linha.

- (a)  $(V)[\ ](F)[\ ]$  Sem a linha marcada pela programadora, o programa funciona perfeitamente, e a programadora está errada.
- (b)  $(V)[\ ](F)[\ ]$  O defeito observado pela programadora consiste em atribuir ao ponto médio  $c$  a um dos novos extremos de intervalos em função de troca de sinal.
- (c)  $(V)[\ ](F)[\ ]$  Com a correção feita pela programadora, esta função procura o ponto médio de cada novo intervalo e seleciona aquele sub-intervalo em que há troca de sinal, isto é feito pelo teste

```
if (f(a)*f(c) < 0)
 b = c;
else
 a = c;
```

- (d) (V)[](F)[] A declaração `ofstream dados;` cria uma variável do tipo arquivo para ser usado como entrada de dados do programa.
- (e) (V)[](F)[] A declaração `ofstream dados;` cria uma variável do tipo arquivo para ser usado como saída de dados do programa.
- (f) (V)[](F)[] A função ainda tem um erro, há dois locais em que o programa lança o ponto médio no arquivo referenciado pela variável `dados`.