INSTITUTO FEDERAL DE MINAS GERAIS

Bacharelado em Ciência da Computação

Disciplina: Cálculo Numérico

Trabalho Prático 01

Professor: Diego Mello da Silva

Formiga-MG
18 de junho de 2014

Sumário

1 Introdução

Este documento apresenta a especificação do trabalho prático 01 da disciplina de Cálculo Numérico, no valor de 15 pontos.

O trabalho consiste na implementação de métodos numéricos para resolver equações polinomiais, mais especificamente os métodos de Newton, da Bissecção e *Regula Falsi* utilizando as linguagens C e Pascal. O trabalho deverá ser testado em ambiente Linux Ubuntu sendo compilado nos compiladores GNU C Compiler (`gcc`) e Free Pascal (`fpc`).

O trabalho poderá ser feito em grupo de **até 2 alunos**. O prazo para entrega do trabalho é de 3 semanas contadas a partir da data em que for anunciado. O trabalho deverá ser postado no sistema acadêmico, na atividade correspondente da disciplina de Cálculo Numérico. Deve-se informar os membros do grupo e matrícula.

A seção 3 detalhará os requisitos da aplicação que deverá ser entregue neste trabalho prático.

2 Recomendações Gerais

O aplicativo deverá ter o nome de `root-solver.bin`, se o código for compilado no Linux, ou `root-solver.exe` se o código for compilado no Windows. Toda a implementacao do aplicativo deverá ser feita em um único arquivo de código fonte, denominado `root-solver.pas` ou `root-solver.c`. Para organizar o código, deve-se fazer uso de procedimentos e funções capazes de implementar os requisitos do *software*, especificados na seção 3

3 Especificação

Esta seção apresenta as recomendações gerais e requisitos esperados para a implementação do aplicativo *solver* descrito neste trabalho prático de implementação. Detalhes sobre cada requisito, assim como o barema de correção serão apresentados ao longo da seção.

3.1 Requisito 01 - Entrada de Dados

A entrada e saída de dados será feita por meio de arquivos, informados como argumentos de linha de comando (ver no *slide* de introdução à disciplina como pode-se implementar em C e Pascal a leitura de argumentos por linha de comando). A sintaxe esperada para o *solver* é:

```
c:\> root-solver.exe <arquivo entrada> <arquivo saída>
```

ou

```
user@machine:~$ root-solver.bin <arquivo entrada> <arquivo saída>
```

dependendo se a aplicação foi compilada em ambiente Windows ou Linux. Independente do caso, são requeridos dois parâmetros, que correspondem ao caminho do

arquivo de entrada (ver formato no Requisito 01, descrito na seção 3.1) e ao caminho do arquivo de saída (ver formato no Requisito 02, na seção 3.2).

A entrada de dados do *solver* será dada por arquivos de entrada, escritos no seguinte formato com cada linha do arquivo de entrada iniciando por um caracter de instrução que indica o conteúdo da linha correspondente. O formato dos arquivos de entrada é dado pelo modelo a seguir:

```
d <String. Descrição com 256 caracteres de comprimento máximo>
c <String. Linha de comentário, ignorada até o '\n'>
m <Inteiro. Número máximo de iterações>
l <Real. Limite inferior do intervalo de isolamento da raíz>
u <Real. Limite superior do intervalo de isolamento do raíz>
e <Real. Epsilon, que corresponde ao erro/tolerância>
n <Inteiro. Grau do polinômio>
a <Real. Lista os (n+1) coeficientes do polinômio separados por ' '>
```

Para exemplificar, seja o polinômio $\boxed{P(x) = 3x^4 + 2x^3 - 1/2x - 2}$. Suponha que o usuário especifique na entrada que espera-se encontrar uma raíz real do polinômio com no máximo 50 iterações, sendo esta aproximação de raíz aceitável quando atingir precisão com erro na terceira casa decimal. Suponha ainda que a raíz foi previamente isolada no intervalo $\big[-100.5, +300.25\big]$. O arquivo de entrada que representa este polinômio deverá ter o seguinte conteúdo:

```
c Título do polinômio
d P(x) = 3x^4 + 2x^3 - (1/2)x - 2
c Numero máximo de iterações
m 50
c Limite do intervalo de isolamento
l -100.5
c Limite do intervalo de isolamento
u 300.25
c Erro aceitável na terceira casa decimal
e 0.0001
c Grau do polinômio
n 4
c Lista de coeficientes do tipo a_i, separados por espaço
a 3.0 2.0 0.0 -0.5 2.0
```

Cada um dos parâmetros de entrada devem ser lidos e armazenados adequadamente na aplicação de forma a serem recuperados durante o processamento. Em especial para os coeficientes do polígono observar as orientações do Requisito 03, descrito na seção 3.3.

3.2 Requisito 02 - Saída de Dados

Ao receber o arquivo de entrada o aplicativo *solver* deverá calcular uma solução numérica para o problema usando os três métodos especificados (bissecção, *regula*

falsi e newton-raphson). Como saída deste processamento espera-se que o aplicativo crie um arquivo de saída contendo os resultados de todos os métodos.

Tal arquivo de saída deve ser escrito como um arquivo HTML (*hyper text markup language*), ou seja, deve conter *tags* HTML para título, corpo do arquivo e tabelas. As tabelas conterão os resultados computados pelos métodos numéricos acima relacionados, iteração por iteração, até encontrar uma solução dentro dos limites de erro aceitáveis ou confirmar que o método não convergiu dentro da quantidade máxima de iterações especificada.

O arquivo deverá conter um cabeçalho inicial, cujo título será a string de descrição informada via arquivo de entrada, com três sub-seções, uma para cada método numérico. Em cada uma delas haverá uma tabela, com os campos a seguir:

1. **Bissecção**: campos `Iter`, `l`, `f(l)`, `u`, `f(u)`, `x`, `f(x)` e `Delta` (ver slides da disciplina para seguir modelo)

2. ***Regula-Falsi***: campos `Iter`, `l`, `f(l)`, `u`, `f(u)`, `x`, `f(x)` e `Delta` (ver slides da disciplina para seguir modelo)

3. **Newton-Raphson**: campos `Iter`, `x`, `f(x)`, `f'(x)` e `Delta` (ver slides da disciplina para seguir modelo)

Além da tabela deverá haver uma string final dizendo se o método convergiu ou não, e qual foi a tolerância de erro usada no cálculo. O modelo de HTML esperado poderá ser encontrado nos anexos, na seção 5.5. Recomenda-se para consultas o livro [Powell] ou qualquer outro material que contenha explicação sobre as tags HTML. É de responsabilidade do grupo estudar as tags adequadas para gerar o arquivo de saída no formato esperado. As próximas sub-seções descrevem os demais requisitos de *software* que o aplicativo `solver` deverá implementar para cumprir com a especificação deste trabalho prático.

3.3 Requisito 03 - Estrutura de Dados de um Polinômio

O arquivo de entrada contêm diversos parâmetros para serem usados pelos métodos numéricos durante o cômputo de uma aproximação da raíz do polinômio de grau n. Uma delas consiste nos coeficientes a_i, com $i = 1, 2, \ldots, n$ que são suficientes para expressar um polinômio na forma $\boxed{a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0}$.

Neste requisito espera-se que o grupo implemente uma estrutura de dados para armazenar um polinômio. Tal estrutura de dados poderá ser implementada usando-se registros da linguagem Pascal ou estruturas da linguagem C. Nela deverá existir campos para armazenar o grau do polinômio, assim como a lista de coeficientes. O grupo deverá escolher a melhor estrutura de dados para representar o polinômio. Tal estrutura deverá ser preenchida na leitura do arquivo de entrada, e será também utilizada como saída do método que implementa uma derivada analítica de $P(x)$, no Requisito 04 (seção 3.4).

3.4 Requisito 04 - Cálculo da Devivada $D(x)$ de $P(x)$

O método de Newton-Raphson é um método baseado em tangente e, portanto, necessita de valores $f'(x)$ calculados em sua equação de recorrência para determinar o próximo número da sequência que aproxima-se de ξ.

Nesta implementação, que lida com cálculo de raízes de equações polinomiais, especifica-se a construção de um procedimento computacional que recebe o polinônio $P(x)$ como entrada e devolve um novo polinômio $D(x)$ como saída. O polinômio pode facilmente ser implementado usando a regra analítica de derivação de polinômios. Tal polinômio deverá utilizar a estrutura de dados descrita no Requisito 03 (seção 3.3).

3.5 Requisito 05 - Avaliação de $P(x)$ e $D(x)$ usando Horner

Durante o cômputo das aproximações os algoritmo da Bissecção, *Regula Falsi* e Newton-Raphson avaliam os polinômios $P(x)$ e $D(x)$. Computacionalmente, o método de Horner é o método mais eficiente de realizar tal avaliação, e é especificado neste requisito para ser usado pelos métodos numéricos citados. O grupo deve, portanto, implementar uma função computacional que recebe como parâmetro a estrutura de dados de polinômio e um valor a ser avaliado, x, e devolve como resposta o valor de $f(x)$ avaliado (seja $f(x)$ o polinômio original $P(x)$ ou a sua derivada $D(x)$).

3.6 Requisito 06 - Método da Bissecção

Neste requisito o grupo deverá implementar o método da bissecção para encontrar uma raíz real isolada no intervalo $\big[l, u\big]$.

3.7 Requisito 07 - Método da *Regula Falsi*

Neste requisito o grupo deverá implementar o método baseado em aproximação linear denominado *regula falsi* para encontrar uma raíz real isolada no intervalo $\big[l, u\big]$.

3.8 Requisito 08 - Método de Newton-Raphson

Neste requisito o grupo deverá implementar o método baseado em tangente de newton-raphson para encontrar uma raíz real isolada no intervalo $\big[l, u\big]$. Como o método utiliza apenas um valor como ponto de partida da aproximação, pode-se escolher o limite inferior u como 'chute' inicial.

3.9 Requisito 09 - Corretude dos Resultados

Para que o trabalho prático tenha cumprido com sua proposta é preciso que a implementação do aplicativo `solver` gere resultados corretos. Neste requisito deve-se verificar o resultado dos métodos para diferentes entradas de dados, algumas fornecidas pelo grupo (5 arquivos de entrada) e algumas usadas pelo professor da

disciplina. Somente pontuará neste requisito o grupo que completar os testes com 100% de corretude nos resultados.

OBS: é preciso fornecer os 5 (cinco) arquivos de entrada junto com o código fonte da aplicação.

3.10 Requisito 10 - Documentação do Código

Este requisito lida com a documentação da implementação em código por meio de comentários. Para atingir o objetivo deste requisito o grupo deverá comentar o cabeçalho do arquivo de entrada, informando o propósito da implementação, comentar cada função, procedimento ou afim, indicando o objetivo do procedimento assim como as entradas e saídas esperadas, comentar as principais estruturas de dados da aplicação e indicar em cada procedimento qual é o número do requisito que ele implementa.

4 Barema

A tabela a seguir descreve o barema usado pelo professor da disciplina como orientação para correção. Ele contêm o valor de cada requisito do trabalho que, se completamente cumprido, totalizará 15 pontos. Cabe ressaltar que trabalhos fora do especificado (codificação, linguagem, compilação e ambiente) não serão corrigidos e valerão zero.

Requisito	**Ptos**
Req 01 - Entrada de Dados	1.0
Req 02 - Saída de Dados	3.0
Req 03 - Estrutura de Dados de Polinômio	0.5
Req 04 - Cálculo da Derivada $D(x)$ de $P(x)$	1.0
Req 05 - Avaliação de $P(x)$ e $D(x)$ usando Horner	1.0
Req 06 - Método da Bissecção	1.5
Req 07 - Método da *Regula Falsi*	1.5
Req 08 - Método de Newton-Raphson	1.5
Req 09 - Corretude dos Resultados	3.0
Req 10 - Documentação do Código	1.0
TOTAL	**15.0**

5 Anexos

Esta seção contém material complementar para implementação deste trabalho prático, que podem ser usados para orientar as implementações dos métodos numéricos e formato dos arquivos de saída.

5.1 Pseudo-código do Algoritmo de Horner

Algoritmo de Horner, adaptado de [Campos]

Algorithm 1 MetodoHorner(c, n, a)

1: { Avalia polinômio $P(x) = c_1x^n + c_2x^{n-1} + \cdots + c_nx + c_{n+1}$, com $x = a$ }
2: $y \leftarrow c_1$
3: **for** $i \leftarrow 2$ **to** $n + 1$ **do**
4: $\quad y \leftarrow y \times a + c_i$
5: **end for**
6: **return** y

5.2 Pseudo-código do Algoritmo de Bissecção

Algoritmo da Bissecção, adaptado de [Campos]

Algorithm 2 MetodoBisseccao(a, b, f, $Toler$, $MaxIter$)

1: **if** $(f(a) \cdot f(b) > 0)$ **then**
2: $\quad$ **escreva** "Função não muda de sinal nos extremos do intervalo"; **abandone**;
3: **end if**
4: $DeltaX \leftarrow (b - a)/2$; $Iter \leftarrow 0$
5: **repeat**
6: $\quad CondicaoOk \leftarrow 1$
7: $\quad x \leftarrow (a + b)/2$
8: $\quad$ **escreva** $Iter$, a, b, x, $f(a)$, $f(b)$, $f(x)$, $DeltaX$
9: $\quad$ **if** $\Big(((DeltaX \leq Toler)$ `AND` $(\texttt{abs}(f(x)) \leq Toler))$ `OR` $(Iter > MaxIter)\Big)$ **then**
10: $\quad\quad CondicaoOk \leftarrow 0$
11: $\quad$ **else**
12: $\quad\quad$ **if** $\Big(f(a) \cdot f(x) > 0\Big)$ **then**
13: $\quad\quad\quad a \leftarrow x$
14: $\quad\quad$ **else**
15: $\quad\quad\quad b \leftarrow x$
16: $\quad\quad$ **end if**
17: $\quad\quad DeltaX \leftarrow DeltaX/2$; $Iter \leftarrow Iter + 1$
18: $\quad$ **end if**
19: **until** $(CondicaoOk = 0)$
20: $Raiz \leftarrow x$
21: **return** $Raiz$

5.3 Pseudo-código do Algoritmo *Regula Falsi*

Algoritmo *Regula Falsi*, adaptado de [Campos]

Algorithm 3 AlgoritmoRegulaFalsi(a, b, $Toler$, $MaxIter$, F)

if $(f(a) \cdot f(b) > 0)$ **then**
  escreva "função não muda de sinal nos extremos"; **abandone**
end if
if $(f(a) > 0)$ **then**
  $t \leftarrow a$; $a \leftarrow b$; $b \leftarrow t$
end if
$Iter \leftarrow 0$; $x \leftarrow b$
repeat
  $DeltaX \leftarrow -F(x)/(F(b) - F(a)) \cdot (b - a)$
  $x \leftarrow x + DeltaX$
  escreva $Iter$, a, $F(a)$, b, $F(b)$, x, $F(x)$, $DeltaX$
  if $\big(Iter \geq MaxIter\big)$ **then**
    escreva "Erro no cálculo da raíz por regula falsi"; **abandone**
  end if
  if $(F(x) < 0)$ **then**
    $a \leftarrow x$
  else
    $b \leftarrow x$
  end if
  $Iter \leftarrow Iter + 1$
until $\big(($ `abs` $(DeltaX) \leq Toler)$ `AND` (`abs` $(F(x)) \leq Toler)\big)$
return x

5.4 Pseudo-código do Algoritmo de Newton-Raphson

Algoritmo de Newton-Raphson, adaptado de [Campos]

Algorithm 4 AlgoritmoNewtonRaphson(x_0, $Toler$, $MaxIter$, f, f')

{ Encontra uma raíz para $f(x)$, partindo de um valor inicial x_0}
$Iter \leftarrow 0$; $x \leftarrow x_0$
$Fx \leftarrow f(x_0)$
$Dx \leftarrow f'(x_0)$
escreva $Iter$, x, Fx, Dx
repeat
  $DeltaX \leftarrow -\Big(\dfrac{Fx}{Dx}\Big)$
  $x \leftarrow x + DeltaX$
  $Fx \leftarrow f(x)$
  $Dx \leftarrow f'(x)$
  $Iter \leftarrow Iter + 1$
  escreva $Iter$, x, Fx, Dx, $DeltaX$
  if $\big((Dx = 0)$ `OR` $(Iter \geq MaxIter)\big)$ **then**
    escreva "Erro no cálculo da raíz por NR"; **abandone**
  end if
until $\big(($ `abs` $(DeltaX) \leq Toler)$ `AND` (`abs` $(Fx) \leq Toler)\big)$
return x

5.5 Modelo de Arquivo HTML de Saída do `solver`

Nesta seção apresentaremos um modelo de arquivo HTML esperado como saída pelo aplicativo *solver* que conterá os resultados numéricos dos métodos da bissecção, *regula falsi* e newton-raphson. O presente modelo exemplifica uma possível execução do arquivo de entrada o polinômio $\boxed{P(x) = 3x^4 + 2x^3 - {}^1\!/_2 x - 2}$.

Arquivo de Saída: `saida-solver.html`

$$P(x) = 3x^4 + 2x^3 - 0.5x - 2$$

Método: Bissecção

Iter	l	f(l)	u	f(u)	x	f(x)	Delta
1	-1.5000000	7.1875000	-0.5000000	-1.8125000	-1.0000000	-0.5000000	0.5000000
2	-1.5000000	7.1875000	-1.0000000	-0.5000000	-1.2500000	2.0429688	0.2500000
3	-1.2500000	2.0429688	-1.0000000	-0.5000000	-1.1250000	0.5202637	0.1250000
4	-1.1250000	0.5202637	-1.0000000	-0.5000000	-1.0625000	-0.0443878	0.0625000
5	-1.1250000	0.5202637	-1.0625000	-0.0443878	-1.0937500	0.2233152	0.0312500
6	-1.0937500	0.2233152	-1.0625000	-0.0443878	-1.0781250	0.0859348	0.0156250
7	-1.0781250	0.0859348	-1.0625000	-0.0443878	-1.0703125	0.0199069	0.0078125
8	-1.0703125	0.0199069	-1.0625000	-0.0443878	-1.0664062	-0.0124552	0.0039062
9	-1.0703125	0.0199069	-1.0664062	-0.0124552	-1.0683594	0.0036719	0.0019531
10	-1.0683594	0.0036719	-1.0664062	-0.0124552	-1.0673828	-0.0044051	0.0009766

Encontrou a raíz real -1.0673828 com precisão epsilon = 0.001

Método: Regula Falsi

Iter	l	f(l)	u	f(u)	x	f(x)	Delta
0	-0.5000000	-1.8125000	-1.5000000	7.1875000	-0.7013889	-1.6133633	0.7986111
1	-0.7013889	-1.6133633	-1.5000000	7.1875000	-0.8477893	-1.2450064	-0.1464004
2	-0.8477893	-1.2450064	-1.5000000	7.1875000	-0.9440841	-0.8276532	-0.0962948
3	-0.9440841	-0.8276532	-1.5000000	7.1875000	-1.0014886	-0.4902977	-0.0574045
4	-1.0014886	-0.4902977	-1.5000000	7.1875000	-1.0333230	-0.2697013	-0.0318345
5	-1.0333230	-0.2697013	-1.5000000	7.1875000	-1.0502012	-0.1421662	-0.0168781
6	-1.0502012	-0.1421662	-1.5000000	7.1875000	-1.0589255	-0.0732354	-0.0087243
7	-1.0589255	-0.0732354	-1.5000000	7.1875000	-1.0633744	-0.0372769	-0.0044489
8	-1.0633744	-0.0372769	-1.5000000	7.1875000	-1.0656272	-0.0188577	-0.0022528
9	-1.0656272	-0.0188577	-1.5000000	7.1875000	-1.0667639	-0.0095103	-0.0011367
10	-1.0667639	-0.0095103	-1.5000000	7.1875000	-1.0673363	-0.0047889	-0.0005725
11	-1.0673363	-0.0047889	-1.5000000	7.1875000	-1.0676244	-0.0024090	-0.0002881
12	-1.0676244	-0.0024090	-1.5000000	7.1875000	-1.0677693	-0.0012118	-0.0001449
13	-1.0677693	-0.0012118	-1.5000000	7.1875000	-1.0678421	-0.0006095	-0.0000729

Encontrou a raíz real -1.0678421 com precisão epsilon = 0.001

Método: Newton-Raphson

Iter	x	f(x)	f'(x)	Delta
0	-1.5000000	7.1875000	-27.5000000	———
1	-1.2386364	1.8801303	-14.0987692	0.2613636
2	-1.1052822	0.3293847	-9.3733063	0.1333542
3	-1.0701414	0.0184806	-8.3351297	0.0351407
4	-1.0679243	0.0000698	-8.2723541	0.0022172
5	-1.0679158	-0.0000002	-8.2721148	0.0000084

Encontrou a raíz real -1.0679158 com precisão epsilon = 0.001

Arquivo gerado automaticamente pelo aplicativo `solver.bin`

6 Bibliografia Recomendada

Referências

[Ruggiero] RUGGIERO, Márcia; LOPES, Vera Lúcia da Rocha. Cálculo Numérico - Aspectos Teóricos e Computacionais. 2^a edição. Editora Pearson Makron, 1996. ISBN: 978-85-346-0204-4.

[Sperandio] SPERANDIO, Décio; MENDES, João Teixeira. Cálculo Numérico: Características Matemáticas e Computacionais dos Métodos Numéricos. Editora Pearson Prentice Hall, 2003. ISBN: 85-87918-74-5.

[Barroso] BARROSO, Leonidas; CAMPOS FILHO, Frederico Ferreira. Cálculo Numérico (Com Aplicações). 2^a edição. Editora Harbra, 1987. ISBN: 85-29400-89-5

[Campos] CAMPOS FILHO, Frederico Ferreira. Algoritmos Numéricos, 2^a edição. Editora LTC (Grupo GEN), 2007. ISBN: 85-21615-37-8.

[Powell] POWELL, Thomaz A. HTML and CSS - The Complete Reference, Fifth Edition. Editora McGrall Hill, 2010. ISBN: 978-0071741-70-5