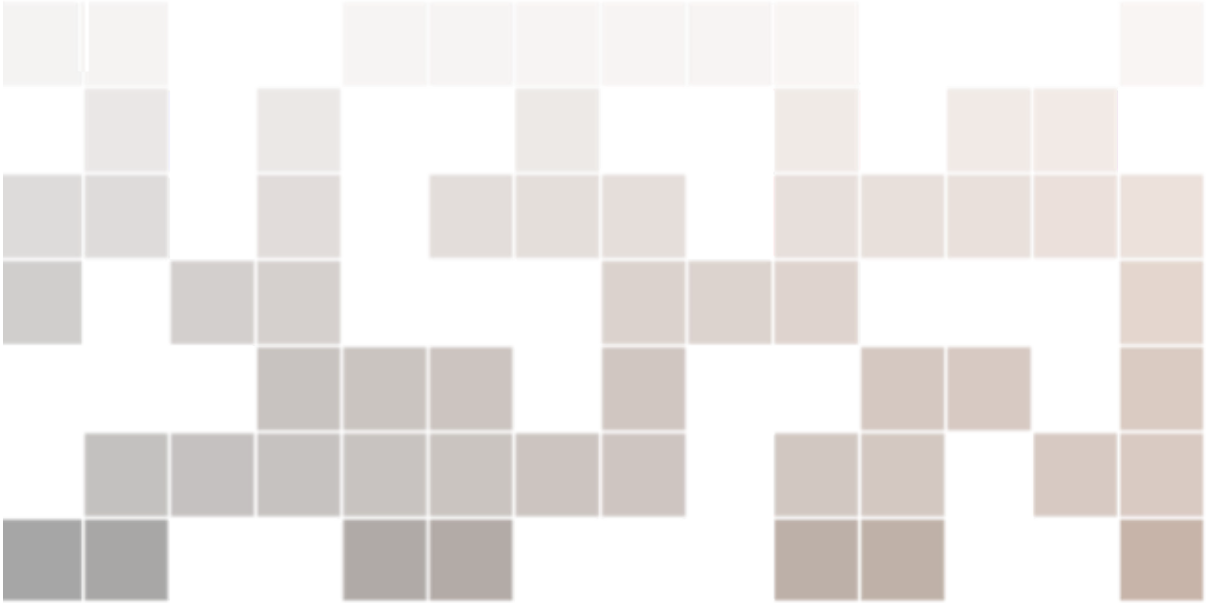
BEC & BCC

Cálculo Numérico

Apostila 42[1]

Prof. Eduardo Heredia

eduardo.heredia@sp.senac.br



[1] https://nerdice.com/42-a-resposta-para-vida-o-universo-e-tudo-mais

1 Sumário

2 Plano de Ensino

2.1 Ementa

Aborda a identificação do melhor algoritmo para resolver um problema numérico, bem como a implementação de algoritmos clássicos e a análise desses algoritmos sob diversos pontos de vista: eficiência, precisão, convergência, estabilidade, aplicabilidade e restrições. Apresenta os principais métodos computacionais para resolução de problemas clássicos: determinação de zeros de funções não-lineares, sistemas de equações lineares, aproximação de funções, técnicas de interpolação, cálculo de áreas e funções de densidade.

2.2 Metodologia

2.2.1 Aulas Teóricas

Aulas expositivas, com análise e discussão dos temas propostos, ministradas na 1ª parte das aulas

2.2.2 Aulas Práticas

Exercício prático sobre os temas abordados nas aulas teóricas, aplicados na 2ª parte das aulas.

2.2.3 Atividades Discentes Orientadas

Atividades propostas para desenvolvimento fora da sala de aula com a finalidade de complementar o aprendizado.

2.3 Avaliação

2.3.1 Instrumentos

2.3.1.1 Caso opte por realizar o EP:

- Avaliações individuais, **P1** e **P2,** cuja média terá peso 0,5 na Média Final (MF)
- Atividades discentes orientadas, cuja média terá peso 0,1 na Média Final (MF)
- EP (Exercício Problema), com peso 0,3 na Média Final (MF)
- Projeto Integrador, **PI**, com peso 0,1 na Média Final (MF)

2.3.1.2 Caso opte por não realizar o EP:

- Avaliações individuais, **P1** e **P2,** cuja média terá peso 0,8 na Média Final (MF)
- Atividades discentes orientadas, cuja média terá peso 0,1 na Média Final (MF)
- Projeto Integrador, **PI**, com peso 0,1 na Média Final (MF)

2.3.2 Discente sem PI

Aluno sem PI, terá o acréscimo de 0,1 no peso das avaliações individuais

2.4 Planilha de Notas

2.4.1 ADO´s e EP

https://docs.google.com/spreadsheets/d/1SAwa0wSoCaSfPi4IPmHwEXQVSkf-8vhKZb26XJii-y8/edit?usp=sharing

3 Recursos Adicionais

3.1 Censo da Disciplina

Preencha o seguinte formulário (você precisará de uma conta na **Google**):

https://docs.google.com/forms/d/e/1FAIpQLScuCP87XBtp9k1CcWgCTAAWNoFC-j-1UDBUpNID90gIf6CbJA/viewform?usp=sf_link

3.2 Repositório Local (Senac SP)

Nas dependências do Senac é possível acessar o seguinte repositório local:

```
\\10.135.0.60\cas-uni educacional\Graduacao\Engenharia de Computação\Eduardo Heredia\02 Disciplinas\2018-1\BEC Cálculo Numérico
```

3.3 Repositório Remoto (GitHub)

https://github.com/eheredia2511/senac-bec-cn-1s2018

3.4 Máquina Virtual Linux (formato OVA)

http://www.mediafire.com/file/dnm3t2tacfn4cfv/BEC_CN_MV_Linux_Antergos_Deepin_64_bits_rev_01.ova

3.5 Repositório GitHub do Aluno

3.5.1 Identificação do Repositório da Disciplina

Nome do repositório	senaccn

4 Material de Aula

4.1 Material 0x01: Esquenta

4.1.1 Objetivo

Implementar algoritmo computacional para converter um número natural 0 <= X <= 2^{32} na base **N** (2 <= N <= 36) para a base **M (** 2 <= N <= 36 **)**..

4.1.2 Requisitos

- Deve ser um utilitário de linha de comando escrito em C, versão 2011, utilizando apenas a biblioteca padrão, com a seguinte sintaxe de uso:

  `n2m` `<número a converter> <base do número> <base desejada>`

- A saída deverá ser a representação do `<número a converter>` na base desejada, seguido por uma mudança de linha (`\n`, na linguagem C).

  `Exemplo de execução do programa: converter` 256_{10} `para base 16`

  ```
  $ ./n2m 256 10 16
  $ 100
  $
  ```

- O código desenvolvido deverá fazer todas as consistências nos valores de entrada de forma a não ter um comportamento errático. Nenhuma saída deverá ser gerada pelo programa caso haja erros nos dados de entrada ou a conversão não puder ser realizada.

- Seguindo o padrão Linux, no caso de sucesso, o programa deverá terminar com o status 0; em caso de erro, com status 1.

- Bases maiores que 10, por exemplo 11, deverão seguir o exemplo da base 16, ou seja, neste caso, utilizar os caracteres **0,1,2,3,4,5,6,7,8,9** e **A**. A base 36, por outro lado, utiliza todos os caracteres **0,1,2,3,4,5,6,7,8,9** e as letras de **A** a **Z**.

4.1.3 Entrega

Pasta em seu repositório no GitHub (branch master), nomeada **senac-cn-ado01**, contendo SOMENTE os seguintes arquivos:

n2m.c	Implementação da interface do utilitário de linha de comando
lib.c	Código para conversão entre bases
lib.h	Cabeçalho associado ao fonte lib.c
leiame.txt	Arquivo texto, codificado em utf-8, explicando o algoritmo implementado.

Compilação e Linkedição de seu programa:

```
gcc -W -Wall -pedantic -std=c11 n2m.c lib.c -lm -o n2m
```

ATENÇÃO

- Caso haja *warnings* ou *erros de compilação,* a ADO será considerada como não concluída!
- Observe que os arquivos têm os nomes grafados em letras minúsculas.

Data de entrega: até **21h00** do dia **20/03/2018,** via *commit* em seu repositório

4.2 Material 0x02: Erros Absoluto e Relativo

4.2.1 Objetivo

Implementar solução computacional baseada na Série de Taylor para calcular os valores de seno e cosseno de um arco dado em radianos e os erros relativo e absoluto, tomando-se como base as funções sin(x) e cos(x) da biblioteca padrão (*math.h*).

4.2.2 Referências

Série de Taylor	http://mathworld.wolfram.com/TaylorSeries.html

4.2.3 Requisitos

- Deve ser um utilitário de linha de comando escrito em C, versão 2011, utilizando apenas a biblioteca padrão e o utilitário *gnuplot* para geração dos gráficos.

  `trig`

- Após a execução do programa, quatro arquivos deverão ser gerados:

`seno.png`	Plotagem da função seno(x) entre 0 e 4π rad
`cosseno.png`	Plotagem da função cosseno(x), entre 0 a 4π rad
`seno.dat`	Tabela com valores do arco, do seno(arco), do erro absoluto e do erro relativo, separados pelo caracter <TAB> (em C, '\t')
`cosseno.dat`	Tabela com valores do arco, do cosseno(arco), do erro absoluto e do erro relativo, separados pelo caracter <TAB> (em C, '\t')

- Utilize números de ponto flutuante de precisão dupla (em C, 'double').

- Óbvio, mas relevante de se lembrar, os valores de seno(x) e cosseno(x) a serem plotados/tabelados, devem ser calculados pelas funções desenvolvidas pelos alunos e não por aquelas da biblioteca padrão!

- SUGESTÃO: gere os gráficos com o utilitário *gnuplot* utilizando a função C ***popen***.

4.2.4 Entrega

Pasta em seu repositório no GitHub (branch master), nomeada **senac-cn-ado02**, contendo **SOMENTE** os seguintes arquivos:

trig.c	Implementação da interface do utilitário de linha de comando
lib.c	Código das funções trigonométricas e cálculo dos erros
lib.h	Cabeçalho associado ao fonte lib.c
leiame.txt	Arquivo texto, codificado em utf-8, com breve explicação sobre o projeto

Compilação e Linkedição de seu programa:

```
gcc -W -Wall -pedantic -lm -std=c11 trig.c lib.c -o trig
```

ATENÇÃO

- Caso haja *warnings* ou *erros de compilação,* a ADO será considerada como não concluída!
- Observe que os arquivos têm os nomes grafados em letras minúsculas.

Data de entrega: até **21h00** do dia **27/03/2018,** via *commit* em seu repositório

4.3 Material 0x03: Erros “Assustadores”!

4.3.1 Objetivo

Implementar algoritmo computacional para estimar valor da constante de Euler (***e***) a partir da expressão:

$$\lim_{n\to\infty}(1+\frac{1}{n})^n$$

4.3.2 Referências

Número de Euler	http://mathworld.wolfram.com/e.html

4.3.3 Requisitos

- Deve ser um utilitário de linha de comando escrito em C, versão 2011, utilizando apenas a biblioteca padrão e o utilitário *gnuplot* para geração dos gráficos.

- Após a execução do programa, três arquivos deverão ser gerados, quais sejam:

`euler_flt.png`	Plotagem da aproximação de ***e*** com ***n*** variando de 10^0 a 10^{20}. ***n*** deve ser definida como ***float***.
`euler_dbl.png`	Plotagem da aproximação de ***e*** com ***n*** variando de 10^0 a 10^{20}. ***n*** deve ser definida como ***double***.
`euler_ldbl.png`	Plotagem da aproximação de ***e*** com ***n*** variando de 10^0 a 10^{20}. ***n*** deve ser definida como ***long double***.

- SUGESTÕES:
  - Gere os gráficos com o utilitário *gnuplot* utilizando a função C ***popen***
  - Utilize escala logarítmica no eixo das abscissas (valores de ***n***)

4.3.4 Entrega

Pasta em seu repositório no GitHub (branch master), nomeada **senac-cn-ado03**, contendo **SOMENTE** os seguintes arquivos:

euler.c	Implementação da interface do utilitário de linha de comando
lib.c	Código das funções

lib.h	Cabeçalho associado ao fonte lib.c
leiame.txt	Arquivo texto, codificado em utf-8, com breve explicação sobre o projeto

Compilação e Linkedição de seu programa:

```
gcc -W -Wall -pedantic -lm -std=c11 euler.c lib.c -o euler
```

ATENÇÃO

- Caso haja *warnings* ou *erros de compilação,* a ADO será considerada como não concluída!
- Observe que os arquivos têm os nomes grafados em letras minúsculas.

Data de entrega: até **21h00** do dia **03/04/2018,** via *commit* em seu repositório

4.4 Material 0x04: Programação Científica

4.4.1 Objetivo

- Experimentar a linguagem Fortran através do desenvolvimento de um código computacional para encontrar a raiz quadrada de um número pelo Método das Secantes, da Bissecção e de Newton-Raphson.

4.4.2 Referências

Fortran	https://gcc.gnu.org/wiki/GFortran https://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_fortran90.pdf http://leandro.iqm.unicamp.br/leandro/shtml/didatico/simulacoes/tutorial.pdf https://software.intel.com/en-us/fortran-compilers http://fortranwiki.org/fortran/show/Libraries

4.4.3 Requisitos

- Deverão ser um utilitários de linha de comando escrito em Fortran, versão 90/95, utilizando apenas a biblioteca padrão, com as seguintes sintaxes de uso:

```
secante <número cuja raiz se quer obter>
bissec <número cuja raiz se quer obter>
newton <número cuja raiz se quer obter>
```

- A saída deverá ser o valor da raiz quadrada do número informado.

Exemplo de execução do programa: raiz quadrada de 16

```
$ ./secante 16
$ 4
$ ./bissec 16
$ 4
$ ./newton 16
$ 4
```

4.4.4 Entrega

Pasta em seu repositório no GitHub (branch master), nomeada **senac-cn-ado04**, contendo **SOMENTE** os seguintes arquivos:

secante.f95	Busca de raízes de funções pelo Método da Secante
bissec.f95	Busca de raízes de funções pelo Método da Bissecção
newton.f95	Busca de raízes de funções pelo Método de Newton-Raphson.
leiame.txt	Arquivo texto, codificado em utf-8, com breve explicação sobre o projeto

Compilação e Linkedição de seu programa:

```
gfortran secante.f95 -o secante
gfortran bissec.f95 -o bissec
gfortran newton.f95 -o newton
```

ATENÇÃO

- Caso haja *warnings* ou *erros de compilação,* a ADO será considerada como não concluída!
- Observe que os arquivos têm os nomes grafados em letras minúsculas.

Data de entrega: até **21h00** do dia **17/04/2018,** via *commit* em seu repositório

4.5 Material 0x05: Fugindo dos Erros

4.5.1 Objetivo

- Experimentar a linguagem Fortran através do desenvolvimento de um código computacional para encontrar as raízes complexas de uma função do 2o grau.

4.5.2 Referências

Fortran	https://gcc.gnu.org/wiki/GFortran https://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_fortran90.pdf http://leandro.iqm.unicamp.br/leandro/shtml/didatico/simulacoes/tutorial.pdf https://software.intel.com/en-us/fortran-compilers http://fortranwiki.org/fortran/show/Libraries

4.5.3 Requisitos

- Deverá ser um utilitários de linha de comando escrito em Fortran, versão 2008, utilizando apenas a biblioteca padrão, com as seguintes sintaxes de uso:

  `bhaskara <a> <b> <c>`

- A saída deverá apresentar as raízes complexas da função cujos coeficientes a, b e c são dados

```
Exemplo de execução do programa: raiz quadrada de 16
---------------------------------------------------------
$ ./bhaskara 1 5 6
$ r1 = -2.0
$ r2 = -3.0
```

4.5.4 Entrega

Pasta em seu repositório no GitHub (branch master), nomeada **senac-cn-ado05**, contendo **SOMENTE** os seguintes arquivos:

bhaskara.f08	Busca de raízes complexas de funções do 2o grau

Compilação e Linkedição de seu programa:

```
gfortran bhaskara.f08 -o bhaskara
```

ATENÇÃO

- Caso haja *warnings* ou *erros de compilação,* a ADO será considerada como não concluída!
- Observe que os arquivos têm os nomes grafados em letras minúsculas.

Data de entrega: até **21h00** do dia **08/05/2018,** via *commit* em seu repositório

5 EP: GA Behind Google

5.1 Entrega

Pasta em seu repositório no GitHub (branch master), nomeada **senac-cn-ep**, contendo todos os seus arquivos fonte: **ATENÇÃO** • Caso haja *warnings* ou *erros de compilação,* o EP será considerado como não concluído!
Data de entrega: até **21h00** do dia **13/06/2018,** via *commit* em seu repositório