Universidade Estadual de Santa Cruz – UESC

Relatório de Implementações de Métodos da Disciplina Análise Numérica

Relatório de implementações realizadas por Levy Marlon Souza Santiago

Disciplina Análise Numérica.

Curso Ciência da Computação

Semestre 2017.2

Professor Gesil Sampaio Amarante II

Ilhéus – BA
2018

ÍNDICE

Linguagem(ns) Escolhida(s) e justificativas

A linguagem escolhida foi o Python, pois além de ser uma linguagem em que o autor deste relatório já é familiarizado, existem algumas bibliotecas para cálculos matemáticos e manipulação de funções que já estão implementadas para se utilizar nesta linguagem. A versão do python utilizada foi a versão 3.5.3. O Sistema Operacional usado para implementar os métodos foi o Linux distribuição Ubuntu 17.04.

Especificações Arquivo de Entrada

Biblioteca usada:

Para todos os métodos foi necessário usar uma biblioteca chamada sympy. Para instalar esta biblioteca no Python3, foi utilizado o seguinte comando no terminal Linux: $ sudo pip3 install sympy. Foi preferível usar esta biblioteca para todos os métodos porque além de implementar funções para cálculos matemáticos (exponencial, seno, cosseno...), também implementa manipulação de funções e equações (f(x) = x + y + z, x + y = 1...). Por isso foi aberta esta nova sessão para explicar como são representadas as funções matemáticas a partir desta biblioteca.

Também foi utilizada em todos os métodos uma biblioteca chamada sys. Esta biblioteca permite usar argumentos em python, e ela foi usada para inserir como argumento do programa o caminho do arquivo de entrada.

Para poder executar qualquer método, deve-se digitar pyhon3 (ou python, caso a versão 3 esteja configurada como padrão na máquina), o nome do arquivo .py e depois o nome do arquivo de entrada como parâmetro.

$ python3 nomeDoMetodo.py entrada.txt

Algumas representações

Abaixo se encontram as representações de algumas funções matemáticas que podem ser usadas no arquivo de entrada:

- x^2 = pow(x, 2) ou x**2;
- Raiz quadrada de x = sqrt(x);
- Seno de x = sin(x);
- Arcoseno de x = asin(x)
- Cosseno de x = cos(x);
- Tangente de x = tan(x);
- Número de Euler (e) elevado a x = exp(x);
- Log de x na base y = log(x, y);
- pi é uma constante já definida (3,1415…..);

Observações

Abaixo se encontram algumas observações que é preciso se atentar ao gerar o arquivo de entrada:

- O programa não reconhece 5x, mas sim 5*x;

Método de Interpolação por Lagrange

Estratégia de Implementação:

Para o cálculo dos polinômios L0, L1, L2, …, foi criada uma função ‘calculaL’, que recebe como parâmetro: ‘k’ (a iteração atual do somatório do método), ‘xn’ (a lista em que os valores de x estão) e ‘n’ (o tamanho da lista ‘xn’). E, a partir da função ‘lagrange’, que implementa o método em si, essa função é chamada várias vezes para calcular o L naquela iteração e o somatório do método ser calculado. Os parâmetros da função ‘lagrange’ são: ‘fn’ (os valores da função para cada x), ‘xn’ (a lista dos valores de x). Para implementar este método foi feito o uso da função Poly da biblioteca sympy, que ajuda na manipulação de polinômios.

Um fato importante é que esse método só aceita a variável ‘x’ para o uso em uma função de entrada. Pois é o símbolo que foi definido no programa.

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada do método em questão foi organizado na seguinte ordem: Primeiro os valores da função para cada valor de x, depois os valores de x, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando a função P(x) gerada a partir das entradas. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3…

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

15

8

-1

-1

0

3

Resultado:

P(x) = 1.0*x**2 - 6.0*x + 8.0

Problema 2:

0

6

24

60

1

3

4

5

Resultado:

P(x) = 1.0*x**3 - 3.0*x**2 + 2.0*x

Problema 3:

16.44

20.08

24.53

2.8

3.0

3.2

Resultado:

P(x) = 10.1250000000001*x**2 - 40.5250000000028*x + 50.5300000000075

Dificuldades enfrentadas

Não houveram dificuldades neste método uma vez que ele não exigiu novas habilidades diferentes daquelas utilizadas no primeiro relatório.

Método da Interpolação de Newton

Estratégia de Implementação:

Para a implementação do método, foi criada uma função ‘calculaF’ que vai fazer o cálculo do f [xn, xn-1, …, x1, x0] de forma recursiva. Essa função é chamada várias vezes pela função ‘intrNewton’ para realizar o cálculo para cada combinação dos valores de x na função f. Ambas as funções recebem como parâmetro o ‘fn’ (lista de valores da função f para cada valor de x) e ‘xn’ (lista de valores de x).

A recurssão do ‘calculaF’ implementa duas bases. Primeiro, se o tamanho de ‘xn’ for 1, então a função retorna f [x0]. Segundo, seo tamanho for igual a 2, então é usada a fórmula do método para calcular f [x0, x1]. Se não for nenhum dos casos, quer dizer que existe mais de dois valores de x (x0, x1, x2, etc). Assim, recursivamente a função vai dividindo a lista até o momento que sobre 2 elementos ou 1 elemento, caindo assim em uma das bases.

Neste método também está definido o ‘x’ como variável única para o uso nas funções de entrada.

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada do método em questão foi organizado na seguinte ordem: Primeiro os valores de x, depois os valores da função para cada valor de x, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando a função P(x) gerada a partir das entradas. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3...

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

4

5

6

0.27

0.38

0.51

Resultado:

P(x) = 0.01*x**2 + 0.02*x + 0.03

Problema 2:

1.00

1.10

1.15

1.000

1.048

1.118

Resultado:

P(x) = 6.13*x**2 - 12.393*x + 7.263

Problema 3:

-1

0

3

15

8

-1

Resultado:

P(x) = 1.0*x**2 - 6.0*x + 8.0

Dificuldades enfrentadas

Não houveram dificuldades neste método uma vez que ele não exigiu novas habilidades diferentes daquelas utilizadas no primeiro relatório.

Método MMQ Contínuo

Estratégia de Implementação:

A função que implementa o método é o ‘mmqContinuo’. Ela recebe como parâmetro: ‘fx’ (a função que passará pelo método), ‘intervalo’ (o intervalo de integração) e ‘base’ (a base de sub-espaço). Para as integrações foi usada a função ‘integrate’ da biblioteca sympy, que realiza tanto a integral indefinida quanto a definida. Para criar as matrizes foi usado o tipo ‘Matrix’ da mesma biblioteca, que é o retorno da função ‘zero’, a qual cria uma matriz do tipo ‘Matrix’ que é inicializada com zeros.

Após a primeira parte do método, ou seja, até o momento em que o sistema é gerado, o método usado para resolvê-lo, foi a fatoração LU (da biblioteca usada). Pois como visto no primeiro relatório, esse é um dos métodos mais eficientes. O retorno é uma lista contendo o resultado do sistema. O que é feito no fim da ‘mmqContinuo’ é a construção de um polinômio, multiplicando o primeiro termo por 1, o segundo por x, o terceiro por x^2 e assim por diante.

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada tem a seguinte ordem: Primeiro a função a qual será aplicado o método, depois o intervalo de integração e por fim a base de sub-espaço, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando a função f(x) aproximada por um polinômio representado pela base. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3...

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

x**4 - 5*x

-1

1

1

x

x**2

Resultado:

f(x) ~= 0.857*x**2 - 5.0*x - 0.086

Problema 2:

(x**3 - 1)**2

0

1

1

x

x**2

Resultado:

f(x) ~= -1.214*x**2 + 0.057*x + 1.019

Problema 3:

1/(x+2)

-1

1

1

x

x**2

Resultado:

f(x) ~= 0.159*x**2 - 0.296*x + 0.496

Dificuldades enfrentadas

Uma dificuldade encontrada foi o fato de que a função da biblioteca ‘sympy’ que implementa a fatoração LU só aceita o tipo ‘Matrix’, o que anteriormente não era sabido no momento da implementação do método.

Método de MMQ Discreto

Estratégia de Implementação:

A estratégia usada para a implementação do método foi calcular primeiramente os pares (u0, u0), (u1, u0), etc, a matriz que contém os valores dos pares e o vetor dos pares (y, u0), (y, u1), etc. Depois de calculados, foi utilizada a mesma função de fatoração LU do método anterior para resolver o sistema. Depois do sistema resolvido, a partir do resultado encontrado, foi construída a função de aproximação P(x).

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada tem a seguinte ordem: Primeiro os valores de x, depois os valores da função para cada valor de x e por fim a base de sub-espaço, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando a função f(x) aproximada por um polinômio representado pela base. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3...

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

-1

0

1

2

0

-1

0

7

1

x

x**2

Resultado:

f(x) ~= 2.0*x**2 + 0.2*x - 1.6

Problema 2:

-2

-1

0

1

2

0

0

-1

0

7

1

x

Resultado:

f(x) ~= 1.4*x + 1.2

Problema 3:

-3

-1

1

2

3

-1

0

1

1

-1

1

x

x**2

Resultado:

f(x) ~= -0.198*x**2 + 0.116*x + 0.903

Dificuldades enfrentadas

Não houve nenhuma dificuldade enfrentada uma vez que este método é bem parecido com o anterior.

Método do Trapézio

Estratégia de Implementação:

Para a implementação do método, foi criada uma função ‘gerarIntervalos’ onde são gerados os valores dos intervalos usando o valor de ‘h’ que de acordo com o método é (‘fim do intervalo’ - ‘inicio’)/numero de subintervalos. Esta função é chamada pela função ‘metodoTrapezio’, onde o método é implementado. Primeiramente o ‘h’ é calculado. Depois é chamada a função ‘gerarIntervalos’ passando como parâmetro o inicio do intervalo, o fim e o número de subintervalos. E por fim é aplicado o resto do método em cima dos dados gerados.

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada tem a seguinte ordem: Primeiro a função a ser integrada, depois o intervalo de integração e por fim o número de subintervalos, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando o resultado da integração definida. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3...

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

exp(x)

0

1

10

Resultado:

I(f(x)) ~= 1.72

Problema 2:

exp(x)*cos(x)

0

1.2

6

Resultado:

I(f(x)) ~= 1.639

Problema 3:

sqrt(x)

1

1.30

6

Resultado:

I(f(x)) ~= 0.321

Dificuldades enfrentadas

Como este método é quase a mesma implementação do método anterior, então não houve nenhuma dificuldade ao implementá-lo.

Método de Simpson 1/3

Estratégia de Implementação:

A implementação deste método utilizou quase os mesmos recursos do método anterior. A única mudança é a diferença na implementação dos próprios métodos.

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada tem a seguinte ordem: Primeiro a função a ser integrada, depois o intervalo de integração e por fim o número de subintervalos, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando o resultado da integração definida. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3...

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

exp(x) * cos(x)

0

1.2

3

Resultado:

I(f(x)) ~= 1.6487

Problema 2:

1/x

1

3

2

Resultado:

I(f(x)) ~= 1.1

Problema 3:

ln(x) + 3*x**2

2

4

2

Resultado:

I(f(x)) ~= 58.1588

Dificuldades enfrentadas

Como este método é quase a mesma implementação do método anterior, então não houve nenhuma dificuldade ao implementá-lo.

Método de Simpson 3/8

Estratégia de Implementação:

A estratégia para implementação é quase a mesma que o método anterior. Muda apenas algumas operações.

Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada tem a seguinte ordem: Primeiro a função a ser integrada, depois o intervalo de integração e por fim o número de subintervalos, entradas separadas por um enter. O arquivo pode conter outras entradas seguindo esta mesma ordem, lembrando que cada nova entrada (novo problema) deve ser separada por dois espaços (dois Enters).

O arquivo de saída é gerado informando o resultado da integração definida. Sequencialmente, são informados as respostas das outras entradas, caso houveram outras.

Problema teste 1, 2, 3...

Abaixo estão três entradas que foram testadas neste método e seus respectivos resultados:

Problema 1:

exp(x) * cos(x)

0

1.2

3

Resultado:

I(f(x)) ~= 1.6387

Problema 2:

1/x

1

3

2

Resultado:

I(f(x)) ~= 1.1872

Problema 3:

ln(x) + 3*x**2

2

4

2

Resultado:

I(f(x)) ~= 70.0396

Dificuldades enfrentadas

Como este método é quase a mesma implementação do método anterior, então não houve nenhuma dificuldade ao implementá-lo.

Considerações Finais

É importante lembrar que as implementações destes métodos não estão completamente revisadas e testadas, por isso, podem haver alguns casos que gerem algum problema. Mas por fim, grande parte dos problemas podem ser resolvidos com estas implementações.