Relatório EP3 - MAC0210

Fernanda Itoda 10740825

Intergração Numérica

1 Parte 1

A primeira parte do EP3 constitui na implementação da Interpolação de Newton ou Lagrange, Integração pelo Método do Trapézio Composto e pelo Método de Simpson Composto.

Para interpolar a função $g(x) = F(x)cos(x)$, foi utilizada a interpolação de Newton, que utiliza o conceito de Diferenças Divididas:

$$P_n(x) = f[x_0]+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+...+f[x_0,x_1,...,x_n](x-x_0)...(x-x_{n-1})$$

$$= \sum_{j=0}^{n}\Big(f[x_0,x_1,...,x_j]\prod_{i=0}^{j-1}(x-x_i)\Big)$$

onde $f[x_i] = f(x_i)$ e $f[x_i,...,x_j]= \frac{f[x_{i+1},...,x_j]-f[x_i,...,x_{j-1}]}{x_j-x_i}$ (Diferenças Divididas).

- A função *void newtonDividedDiff (float x[], float y[][10], int n)*calcula as Diferenças Divididas através do vetor *y[][0]*, que contém os valores de *g(x)*, e as armazena nas colunas seguintes do mesmo vetor (tabela).
- A função *float evalP (float x[], float y[][10], float x0, int n)*calcula *P(x)*em função de *x0*.

A seguir, é realizada a integração pelo Método do Trapézio Composto:

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{r}[f(t_{i-1}+f(t_i)].$$

Nesse método, o erro é de ordem $O(h^2)$.

- A função *float compositeTrap (float x[], float y[][10], int n)* computa a integral através desse método, com $h = \frac{x_{r-1}-x_0}{r} = 1$, $r = 30$ e utilizando a função interpolada *P(x)*.

Por fim, é realizada a integração pelo Método de Simpson composto:

$$\int_a^b f(x)dx \approx \frac{h}{3}\Big[f(a)+2\sum_{k=1}^{r/2-1} f(\int_a^b f(x)dx \approx \frac{h}{3}\Big[f(a)+2\int_a^b f(x)dx \approx \frac{h}{3}\Big[f(a)+2\sum_{k=1}^{r/2-1} f(t_{2k})+4\sum_{k=1}^{r/2} f(t_{2k-1}$$

Nesse método, o erro é de ordem $O(h^4)$.

- A função *float compositeSimpson (float x[], float y[][10], int n)* computa a integral através desse método, com $h = \frac{x_{r-1}-x_0}{r} = 1$, $r = 30$ e utilizando a função interpolada *P(x)*.

Observações

1. Da tabela fornecida, foram utilizados os campos *x(em metros)*e *F(x)cos(θ(x))*.
2. Foi utilizada a interpolação de Newton ao invés de Lagrange visando o menor esforço computacional (menor número de operações realizadas).
3. Para $r = 30$, os valores obtidos foram 117.209053 para a integração pelo Método do Trapézio Composto e 117.131348 para a integração pelo Método de Simpson composto. Os valores tornam-se mais próximos conforme r aumenta, até que o erro de arredondamento se sobrepõe ao erro de interpolação. Por exemplo, até $r = 1000$ os valores estão tornando-se mais próximos, mas para $r = 10000$, os valores voltam a se afastar.
4. Como o Método de Simpson apresente uma aproximação melhor, conclui-se que a integral da função $F(x)cos(x)$ deve ser mais próxima do resultado apresentado por ese método.

2 Parte 2

A segunda parte do EP3 constitui na implementação da integração pelo Método de Monte Carlo unidimensional e multidimensional. As funções integradas e seus valores exatos são:

1. $\int_0^1 sin(x)dx = 0.4596976941318603$
2. $\int_3^7 x^3 dx = 580$
3. $\int_0^\infty e^{-x} dx = 1$
4. $\pi = 3.14159265359$

A aproximação pelo método de Monte Carlo $\hat{I}_n$ unidimesional é dada por:

$$\hat{I}_n = \sum_{i=1}^{n} \frac{g(U_i)}{n}.$$

Para o caso geral:

$$I = \int_a^b g(x)dx$$

A aproximação pelo método de Monte Carlo $\hat{I}_n$multidimensional é dada por:

$$\hat{I}_n = \frac{1}{n}\sum_{i=1}^{n} g(U_1^i, U_2^i, ..., U_d^i)$$

onde n representa os conjuntos de d variáveis aleatórias independentes e uniformemente distribuídas em [0,1].

- A função *double monteCarlo (int n, int id)* realiza a integração pelo Método de Monte Carlo, unidimensional e multidimensional, de acordo com a função a ser aproximada. O identificador *id*indica qual a função em questão, logo, a função é chamada quatro vezes, uma para cada função.

Observações

1. Para as três primeiras funções analisadas, foi utilizada a integração unidimensional. Para a aproximação de π, foi utilizada a integração multidimensional.

2. As v.a. foram geradas através da função *rand*e *srand*, da biblioteca *stdlib.h*. Essas funções geram valores pseudo-aleatórios do tipo *int*, sendo necessária a tranformação para *double*. Para que seus valores fossem no intervalo [0,1], foi usado o comando *rand()/RAND_MAX*, visto que RAND_MAX é o valor máximo gerado por *rand()*.

3. Como algumas funções forneciam intervalos diferentes de $x = 0$ à $x = 1$, após gerar os valores, foi acrescentado uma expressão conversora ($(v.a.) \times (b - a) + a$) para garantir que os valores finais fossem dentro do intervalo estipulado. Por exemplo, na função $\int_3^7 x^3dx$, os valores de x devem ser entre 3 e 7, logo, $(v.a.) \times (7 - 3) + 3$ é o valor final a ser aplicado em x^3.

4. Especificamente para a função $\int_0^\infty e^{-x}dx$, devido ao intervalo [0,∞[o processo foi diferente. Substituindo $u = \frac{1}{x+1}$ e $du = -u^2dx$, temos que:

$$\int_0^1 \frac{e^{1-\frac{1}{u}}}{u^2}du = 1.$$

   Logo, os valores sorteados para u e em seguida usados para calcular a integral.

5. Para esse relatório, a semente utilizada para gerar os números aleatórios foi 37490234. Os valores obtidos foram, para

   n=100:

- $\int_0^1 sin(x)dx \approx 0.431874$
- $\int_3^7 x^3dx \approx 526.255685$
- $\int_0^\infty e^{-x}dx \approx 0.998193$
- $\pi \approx 3.400000$

n=1000:

- $\int_0^1 sin(x)dx \approx 0.454852$
- $\int_3^7 x^3dx \approx 568.053930$
- $\int_0^\infty e^{-x}dx \approx 1.011088$
- $\pi \approx 3.092000$

n=10000:

- $\int_0^1 sin(x)dx \approx 0.462108$
- $\int_3^7 x^3dx \approx 583.459904$
- $\int_0^\infty e^{-x}dx \approx 1.003738$
- $\pi \approx 3.130400$

6. A partir dos valores obtidos, é possível perceber que os valores tendem a se aproximar do valor real conforme o valor de iterações aumenta.