LabNum - EP3

Lourenço Henrique Moinheiro Martins Sborz Bogo - 11208005

Sumário

1 Parte 1

1.1 Implementação

A primeira parte do EP pedia para que, usando os dados de uma tabela, interpolássemos um certa função usando um polinômio e depois integrássemos o polinômio numericamente usando a Regra do Trapezóide e a Regra de Simpson.

Para a parte da interpolação, optei por fazer usando o Polinômio de Newton, pois achei que seria mais fácil.

Implementei o método usando as seguintes funções:

`calcula_den(int index)` Essa função calcula a seguinte produtória $\prod_{i=0}^{index-1} x_{index} - x_i$.

`calcula_p(int grau, double xp)` Avalia o polinômio de ordem grau no ponto xp. Essa função só funciona caso os coeficientes necesssários para calcular esse polinômio já tenham sido calculados antes.

`calcula_a(int index)` Calcula o coeficiente do termo de grau index do Polinômio de Newton.

`calcula_as()` Calcula todos os coeficientes do polinômio.

`trapezoid(int n)` Pega n pontos igualmente espaçados do polinômio de newton (já construído) e integra usando a Regra do Trapezóide.

`simpsons(int n)` Pega $2n$ pontos igualmente espaçados do polinômio de newton (já construído) e integra usando a Regra de Simpson.

1.2 Resultados

Percebi com testes que, por mais que a Regra de Simpson demore mais pra rodar, ela é muito mais eficiente no quesito de conseguir um resultado muito melhor, particionando o polinômio menos vezes.

Partições	Trapezóide	Simpson
10	117.832290	117.130914
100	117.138600	117.131621
1000	117.131691	117.131621
10000	117.131622	117.131621
100000	117.131621	117.131621
1000000	117.131621	117.131621
10000000	117.131621	117.131621
100000000	117.131621	117.131621

Percebe-se na tabela que, particionando o conjunto 10^2 vezes e usando a regra de simpson, consegue-se o mesmo resultado até a sexta casa decimal que particionando o conjunto 10^5 vezes e usando a Regra do Trapezóide.

2 Parte 2

2.1 Implementação

Fiz dois programas nessa parte:

`pt2.c` Esse programa resolve as integrais dos itens 1, 2 e 3 do enunciado, usando o Método de Monte Carlo. O programa recebe como parâmetro o número de variáveis aleatórias que serão usadas para aplicar o método e o número de sementes que serão testadas (caso esse número seja 100, serão testadas as sementes de 0 a 100). Sua saída será o menor erro encontrado (e sua semente), o maior erro encontrado (e sua semente) e o erro médio, para cada um dos métodos.

As integrais que calculei nesse programa e as substituições usadas foram as seguintes:
$\int_0^1 \sin(x)dx$
$\int_3^7 x^3 dx = 4\int_0^1 (4t+3)^3 dt$
$\int_0^\infty e^{-x}dx = \int_0^1 e^{-x}dx + \int_1^\infty e^{-x}dx \rightarrow x = \frac{1}{u} \quad dx = \frac{-1}{u^2}du \rightarrow \int_0^1 e^{-x}dx + \int_1^0 \frac{-e^{\frac{-1}{u}}}{u^2}du = \int_0^1 e^{-x}dx + \int_0^1 \frac{e^{\frac{-1}{u}}}{u^2}du = \int_0^1 (e^{-x} + \frac{e^{\frac{-1}{x}}}{x^2})dx$
Depois de achar as substituições, foi muito simples. Apenas gerei valores aleatórios entre 0 e 1, apliquei na função que quero integrar e tirei a média dos resultados. Discutirei os resultados encontrados na seção de resultados.

`circ.c` Aqui eu resolvi o item 4 do enunciado usando o Método de Monte Carlo. Foi feita basicamente a mesma coisa que no programa descrito no item anterior, apenas com uma alteração para tratar o fato de que estamos em duas dimensões: ao invés de gerar 1 valor aleatório, eu gerei um par de valores aleatórios entre 0 e 1 e passei para a função que $g(x, y)$ dada no enunciado.

Tirei a média dos resultados, multipliquei por 4, e voilá! π (pelo menos perto o suficiente).

2.2 Resultados

Colocarei aqui tabelas com os erros que obtive ao comparar as funções integradas analiticamente com as integradas por Monte Carlo:

Função	Número de V.A.	Menor Erro	Maior Erro	Erro Médio
$\sin x$	10	3.610268e-05	2.328637e-01	6.322206e-02
x^3	10	1.113892e-01	3.824100e+02	9.294158e+01
e^{-x}	10	2.390419e-05	1.184496e-01	3.206498e-02
$\sin x$	100	4.442498e-06	8.549161e-02	1.889440e-02
x^3	100	1.299647e-01	1.218060e+02	2.797397e+01
e^{-x}	100	2.870791e-05	4.890248e-02	1.003721e-02
$\sin x$	1000	6.023429e-06	2.526216e-02	6.077690e-03
x^3	1000	3.933176e-02	3.620151e+01	8.935365e+00
e^{-x}	1000	1.481880e-06	1.434760e-02	3.167431e-03
$\sin x$	10000	3.760302e-06	8.087467e-03	1.978368e-03
x^3	10000	7.316047e-03	1.157567e+01	2.853298e+00
e^{-x}	10000	3.004806e-06	4.082099e-03	9.929621e-04
$\sin x$	100000	1.330556e-06	2.676081e-03	6.605803e-04
x^3	100000	3.156454e-03	3.921654e+00	9.587524e-01
e^{-x}	100000	2.250376e-06	1.508875e-03	3.319549e-04
$\sin x$	1000000	6.454527e-08	8.011643e-04	1.959469e-04
x^3	1000000	1.348677e-03	1.181796e+00	2.842198e-01
e^{-x}	1000000	3.699904e-07	4.150512e-04	1.036093e-04

Podemos perceber que a função x^3 é a que se comporta da pior maneira em relação ao método, tendo erro médio 10^3 vezes mais alto que os outros métodos.

Agora para o círculo:

Número de V.A.	Menor Erro	Maior Erro	Erro Médio
10	0.458407	3.98126	2.62904
100	0.000515128	0.706994	0.196243
1000	0.000488647	0.178986	0.0444891
10000	5.07252e-05	0.0575342	0.013288
100000	1.34666e-05	0.0183472	0.00404787
1000000	7.78275e-08	0.00525208	0.00126605

OBS: Foram testadas 1000 sementes para cada teste acima.