EP3 – Laboratório de Métodos Numéricos

Adriano Elias Andrade

Data: 25/06/2023

1. Introdução:

Neste ep, existem duas partes, cada uma com sua implementação em C. A primeira parte calcula uma aproximação de integral para estimar o trabalho para os dados da tabela, e a segunda parte calcula uma aproximação de integral por Monte Carlo, e a aplica para as questões dadas. Para compilar os códigos, basta utilizar o comando “make” no terminal.

Parte I

1. Entrada:

Para a entrada da parte 1, temos o formato:

```
<método a ser utilizado (1)trapézio ou (2)Simpson>
<número de pontos interpolados>
```

Onde o número de pontos interpolados é a quantidade de pontos a serem interpolados por Lagrange, que serão utilizados na aproximação da integral. Estes pontos serão interpolados em intervalos iguais entre 0 e 30.

2. Implementação:

Na implementação da parte 1, foi utilizada a interpolação de Lagrange para aproximar uma função com os pontos da tabela dada.

Para calcular Lagrange, primeiro calculamos o denominador de cada termo L_i do polinômio de Lagrange. No programa, é utilizado um vetor global "denominador", onde os valores de denominador[i] equivalem ao denominador de L_i. Como o denominador utiliza apenas os pontos conhecidos da tabela, calculamos ele apenas uma vez. Esse processo é feito na função calculaDenominador().

Com os denominadores calculados, podemos avaliar o polinômio num ponto x. Para isso, em cada avaliação, são calculados os nominadores de cada $L_i(x)$, e guardados no vetor nominadores[i]. Com isso, retornamos a somatória do polinômio de Lagrange:

$$\sum_{i=0}^{6} y_i \frac{nominador[i]}{denominador[i]}.$$

Assim são calculados cada um os pontos a serem interpolados. Esta parte é feita na função evalF().

Por fim, com todos os pontos interpolados, podemos aproximar a integral com um dos métodos, a partir de suas fórmulas:

$$I_{trap.comp.} = \sum_{i=0}^{n-1} (x_{i+1} - x_i) \frac{y_i + y_{i+1}}{2}$$

$$I_{Simp.comp.} = \sum_{i=0}^{n-2} (x_{i+1} - x_i) \frac{y_i + 4y_{i+1} + y_{i+2}}{6}$$

3. Testes:

A seguir, alguns testes feitos para a aproximação da integral, com diferentes quantidades de pontos, para trapézio e Simpson.

nºpontos	Trapézio	Simpson
3	124,975000	103,262000
7	119,089250	116,318250
20	117,325147	117,238545
100	117,138741	117,098026
1000	117,131691	117,126519
10000	117,131622	117,131093
100000	117,131621	117,131568
1000000	117,131621	117,131616

Pelos testes, é possível notar que o método de Simpson converge rapidamente com poucos pontos, mas seu resultado fica um pouco flutuante com muitos pontos. Já o método do trapézio converge mais lentamente no começo, mas chega num resultado mais preciso no final.

Com isso é possível concluir que a regra de Simpson aproxima bem uma integral onde pouco se conhece sobre a função. Já a regra do trapézio aproxima bem uma função onde podemos escolher vários pontos.

Parte II

1. Entrada:

Para a entrada da parte 2, temos o formato:

```
<número de pontos utilizados>
```

Onde o único parâmetro é o número de pontos sorteados para fazer a aproximação por Monte Carlo, para cada uma das integrais pedidas no enunciado.

2. Implementação:

Na implementação da parte 2, sorteamos um número pedido de pontos para serem avaliados nas funções pedidas, e utilizados no método de Monte Carlo.

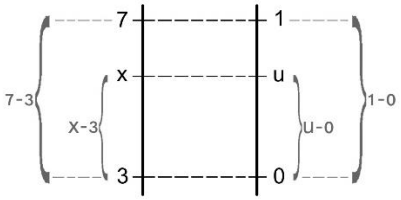
Para sortear um ponto aleatório entre 0 e 1, sorteamos 2 números com a função rand(), e dividimos o menor pelo maior. Para que funcione com qualquer dimensão, os pontos são sorteados em uma matriz tamanho: nº de pontos X dimensão.

Com esses pontos, podemos utilizar o método para cada função do enunciado:

1. $\int_0^1 sin(x)\ dx$ – Caso trivial, basta aplicar o método:

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

2. $\int_3^7 x^3\ dx$ – É necessária uma mudança de variável que transforme o intervalo de 3 a 7 em um de 0 a 1:



$$\Longrightarrow \frac{x-3}{7-3} = \frac{u-0}{1-0}$$

$$\Longrightarrow x = 4u + 3$$

Com isso,

$$dx = 4du$$

Assim, fazemos a troca:

$$\int_3^7 x^3\,dx = \int_0^1 (4u+3)^3 \cdot 4\,du$$

$$\hat{\mathrm{I}}_n = \frac{1}{n}\sum_{i=1}^{n} (4U_i+3)^3 \cdot 4$$

3. $\int_0^\infty e^{-x}\,dx$ – É necessária uma mudança de variável que transforme o intervalo de 0 a ∞ em um de 0 a 1:

Para isso, precisamos pensar numa transformação $\mathrm{u} = \mathrm{t}(\mathrm{x})$ tal que:

$$\begin{cases} 0 = t(0) \\ 1 = \lim\limits_{n\to\infty} t(n) \end{cases}$$

Uma possibilidade é:

$$u = \frac{x}{1+x}$$

Com isso,

$$x = -\frac{u}{u-1}\text{ , e }\ dx = \frac{1}{(u-1)^2}\ du$$

Assim, fazemos a troca:

$$\int_0^\infty e^{-x}\,dx = \int_0^1 e^{\frac{u}{u-1}} \cdot \frac{1}{(u-1)^2}\,du$$

$$\hat{\mathrm{I}}_n = \frac{1}{n}\sum_{i=1}^{n} e^{\frac{U_i}{U_i-1}} \cdot \frac{1}{(U_i-1)^2}$$

4. $g(x,y) = \begin{cases} 1, se\ x^2+y^2 \le 1 \\ 0,\ caso\ contrário \end{cases}$ – Basta utilizar o método para o intervalo [0,1], que resultará na área da circunferência no 1º quadrante, ou seja, $\frac{1}{4}$ da área total dela, π.

Portanto, calculamos por Monte Carlo e multiplicamos o resultado por 4:

$$\frac{\pi}{4} = \int_0^1 \int_0^1 g(x,y)\,dx\,dy$$

$$\pi \approx 4\hat{\mathrm{I}}_n = \ 4 \cdot \frac{1}{n}\sum_{i=1}^{n} g(U_1^i, U_2^i)$$

3. Testes:

A seguir, alguns testes feitos para cada função do enunciado, com diferentes quantidades de pontos utilizados.

nº de pontos	$\int_0^1 sin(x)\,dx$	$\int_3^7 x^3\,dx$	$\int_0^\infty e^{-x}\,dx$	π
10	0,434543	579,555922	0,918461	2,50000
100	0,452902	573,96538	1,007228	3,28000
1000	0,462307	582,383368	1,002778	3,17600
10000	0,461814	582,780435	0,997709	3,14000
100000	0,459815	579,047920	1,000551	3,14108
1000000	0,459564	580,175070	0,999811	3,14120
10000000	0,459796	580,001848	0,999992	3,141577
Valor real	≈0.459697	580	1	π

É possível notar que com o aumento no número de pontos, além de mais próximo do resultado da integral, a variância entre os diferentes testes aleatórios também diminui.