Utilização de Métodos Númericos para Integrais

Jhonattan C. B. Cabral[1], **Daniel M. P. Carvalho**[1]

[1]Instituto de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN)

`jhonattan.yoru@gmail.com, danielmarx08@gmail.com`

Abstract. *Task applied in the second unit of the discipline of Numerical Calculus (DIM0404), aims to improve and apply the knowledge acquired to solve problems involving integrals.*

Resumo. *Tarefa aplicada na primeira unidade da disciplina de Cálculo Numérico (DIM0404), tem como objetivo aprimorar e aplicar os conhecimentos adquiridos para resolução de problemas envolvendo integrais.*

1. Demonstre a regra $3/8$ de Simpson

Vamos utilizar o polinômio interpolador de Newton de grau 3 para efetuar os cálculos, assim teremos:

$$P_3(x) = f[x_0]+(x-x_0)f[x_0,x_1]+(x-x_0)(x-x_1)f[x_0,x_1,x_2]+(x-x_0)(x-x_1)(x-x_2)f[x_0,x_1,x_2,x_3]$$

Agora vamos efetuar as diferenças divididas:

1. $f[x_0,x_1] = \frac{y_1-y_0}{x_1-x_0}$

2. $f[x_0,x_1,x_2] = \frac{y_2-2y_1+y_0}{x_2-x_0}$

3. $f[x_0,x_1,x_2,x_3] = \frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0} = \frac{\frac{y_3-2y_2+y_1}{x_3-x_1}-\frac{y_2-2y_1+y_0}{x_2-x_0}}{x_3-x_0}$

Em seguida determinaremos as seguintes igualdades:

x	$=$	u
x_0		0
x_1		1
x_2		2
x_3		3

Tendo em mãos os resultados das diferenças divididas e as igualdades, Podemos substituir x por u no polinômio, assim ficaremos com:

$$P(u) = y_0 + u(y_1 - y_0) + u(u-1)(\frac{-2y_1 + y_2 + y_0}{2}) + (u^3 - 3u^2 + 2u)(\frac{y_3 - y_0 + 3y_1 - 3y_2}{6})$$

Como sabemos que $\int_{x_0}^{x_3} P_3(x)\, dx = \int_0^3 P_3(u) h\, du$. Logo,

$$\int_0^3 P_3(u) h\, du = h \int_0^3 y_0\, du + h \int_0^3 u(y_1 - y_0)\, du + h(\frac{-2y_1 + y_2 + y_0}{2}) \int_0^3 u^2 - u$$

$$du + h(\frac{y_3 - y_0 + 3y_1 - 3y_2}{6}) \int_0^3 u^3 - 3u^2 + 2u$$

Resolvendo as integrais, temos:

$$h\left[uy_0\right]_{u=0}^3 + h\left[\frac{u^2}{2}(y_1 - y_0)\right]_{u=0}^3 + h(\frac{-2y_1 + y_2 + y_0}{2})\left[\frac{u^3}{3} - \frac{u^2}{2}\right]_{u=0}^3$$

$$+h(\frac{y_3 - y_0 + 3y_1 - 3_2}{6})\left[\frac{u^4}{u} - u^3 + u^2\right]_{u=0}^3$$

Atribuindo os intervalos e isolando o h, conseguimos obter:

$$h\left[3y_0 + \frac{9}{2}(y_1 - y_0) + (\frac{-2y_1 + y_2 + y_0}{2})(9 - \frac{9}{2}) + (\frac{y_3 - y_0 + 3y_1 - 3y_2}{6})(\frac{81}{4} - 27 + 9)\right]$$

$$h\left[3y_0 + \frac{9}{2}(y_1 - y_0) + \frac{9}{2}(\frac{-2y_1 + y_2 + y_0}{2}) + \frac{9}{4}(\frac{y_3 - y_0 + 3y_1 - 3y_2}{6})\right]$$

$$h\left[\frac{9}{2}y_1 - \frac{3}{2}y_0 + \frac{9}{4}(-2y_1 + y_2 + y_0) + \frac{3}{8}(y_3 - y_0 + 3y_1 - 3y_2)\right]$$

$$h\left[-\frac{3}{2}y_0 + \frac{9}{4}y_0 - \frac{3}{8}y_0 + \frac{9}{2}y_1 - \frac{9}{2}y_1 + \frac{9}{8}y_1 + \frac{9}{4}y_2 - \frac{9}{8}y_2 + \frac{3}{8}y_3\right]$$

$$h\left[\frac{3}{8}y_0 + \frac{9}{8}y_1 + \frac{9}{8}y_2 + \frac{3}{8}y_3\right]$$

Resultado:

$$\frac{3}{8}h(y_0 + 3y_1 + 3y_2 + y_3)$$

2. **Implemente as regras compostas do trapézio, $1/3$ de Simpson e $3/8$ de Simpson. Utilize-as para integrar a função**

$$f(x) = x\sin(x)$$

no intervalo [-5, 5]

2.0.1. *trapezio.cpp*

```cpp
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <math.h>

double funcao(double );
double (*func) (double) = funcao;
double trapezio(double , double , int , double (*func)(double));

int main()
{
        //Pre-definindo o numero de iteracoes e os intervalos,
        //sem interacao com o usuario.
        int n = 6; // Numero de sub intervalos
        double a = -5;
        double b = 5;

        double result = trapezio(a, b, n, func);

        std::cout << std::endl;
        std::cout << ">>Resultado aproximado da integral: ";
        std::cout << result << std::endl;
        std::cout << std::endl << std::endl;

        return 0;
}

double funcao(double x)
{
        return x * std::sin(x);
        //return std::pow(x,2); //debug
}

double trapezio(double a, double b, int n, double (*func)(double))
{
        double h = (b-a)/n;
        double vet_x[n+1];
        double vet_y[n+1];

        double width = (std::abs(a) + std::abs(b))/n;
```

```

        vet_x[0] = a;
        vet_y[0] = func(vet_x[0]);
        for(int i = 1; i < n+1; ++i)
        {
                vet_x[i] = vet_x[i-1] + width;
                vet_y[i] = func(vet_x[i]);
        }

        //ex: Regra do trapezio composta para 7 pontos: (aplicada 6 vezes)
        //h[f(x0)/2 + f(x1) + f(x2) + f(x3) + f(x4) + f(x5) + f(x6)/2 ]

        double result = vet_y[0]/2 + vet_y[n]/2; //f(x0)/2 + f(xn)/2

        for(int i = 1; i < n; ++i) //Restante dos valores sao somados
        {
                result = result + vet_y[i];
        }
        result = h * result;
        return result;
}
```

2.0.2. $1 - 3Simpson.cpp$

```
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <math.h>

double funcao(double );
double (*func) (double) = funcao; //Ponteiro para a funcaoo
double um_tres_simpson(double , double , int , double (*func)(double)); //Metodo 1/3 simpson

int main()
{
        //Pre-definindo o numero de iteracoes e os intervalos,
        //sem interacao com o usuario.
        int n = 6; // numero de sub intervalos (tem que ser par)
        double a = -5;
        double b = 5;

        double result = um_tres_simpson(a, b, n, func);

        std::cout << std::endl;
```

```
        std::cout << ">>Resultado aproximado da integral: ";
        std::cout << result << std::endl;
        std::cout << std::endl << std::endl;
        std::cout << "Para mais detalhes sobre a funcao
           integrada e o metodo  1/3 de Simpson, consultar o
           codigo." << std::endl;


        return 0;
}

double funcao(double x)
{
        return x * std::sin(x);
        //return std::pow(x,2);
}

double um_tres_simpson(double a, double b, int n, double (*func)
    (double))
{
        double h = (b-a)/n;
        double vet_x[n+1];
        double vet_y[n+1];

        double width = (std::abs(a) + std::abs(b))/n;

        vet_x[0] = a;
        vet_y[0] = func(vet_x[0]);
        for(int i = 1; i < n+1; ++i)
        {
                vet_x[i] = vet_x[i-1] + width;
                vet_y[i] = func(vet_x[i]);
        }

        //Ex da 1_3 Simpson composta para 7 pontos (Ou seja,
           aplicada 3 vezes):
        //h/3*[ f(x0) + 4*f(x1) + 2*f(x2) + 4*f(x3) + 2*f(x4) +
           4*f(x5) + f(x6) ];

        double result = vet_y[0] + vet_y[n]; //f(x0) + f(xn)
        for(int i = 1; i < n; ++i)
        {
                if(i%2 != 0) //Valor em posicao impar,
                   multiplica por 4
                {
                        result = result + 4 * vet_y[i];
                }
                else //Valor em posicao par, multiplica por 2
                {
```

```
                        result = result + 2* vet_y[i];
                }
        }
        result = (h/3) * result;
        return result;
}
```

2.0.3. $3 - 8Simpson.cpp$

```
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <math.h>

double funcao(double );
double (*func) (double) = funcao; //Ponteiro para a funcao
double tres_oito_simpson(double , double , int , double (*func)(
   double)); //Metodo 3/8 simpson

int main()
{
        //Pre-definindo o numero de iteracoes e os intervalos,
        //sem interacao com o usuario.
        int n = 6; // numero de sub intervalos (tem que ser par)
        double a = -5;
        double b = 5;

        double result = tres_oito_simpson(a, b, n, func);

        std::cout << std::endl;
        std::cout << ">>Resultado aproximado da integral: ";
        std::cout << result << std::endl;
        std::cout << std::endl << std::endl;
        std::cout << "Para mais detalhes sobre a funcao
           integrada e o metodo  1/3 de Simpson, consultar o
           codigo." << std::endl;


        return 0;
}

double funcao(double x)
{
        return x * std::sin(x);
        //return std::pow(x,2);
}

double tres_oito_simpson(double a, double b, int n, double (*
   func)(double))
```

```
{
        double h = (b-a)/n;
        double vet_x[n+1];
        double vet_y[n+1];

        double width = (std::abs(a) + std::abs(b))/n;

        vet_x[0] = a;
        vet_y[0] = func(vet_x[0]);
        for(int i = 1; i < n+1; ++i)
        {
                vet_x[i] = vet_x[i-1] + width;
                vet_y[i] = func(vet_x[i]);
        }

        //Ex de 3/8 Simpson composta para 7 pontos(Ou seja, aplicada duas vezes):
        //3h/8 * [ f(x0) + 3*f(x1) + 3*f(x2) + 2*f(x3) + 3*f(x4) + 3*f(x5) + f(x6) ]

        double result = vet_y[0] + vet_y[n];
        for(int i = 1; i < n; ++i)
        {
                if(i%3 == 0) //Se o indice e multiplo de 3, multiplica f[i] por 2
                {
                        result = result + 2 * vet_y[i];
                }
                else //Se nao, multiplica por 3
                {
                        result = result + 3 * vet_y[i];
                }
        }
        result = (3*h/8) * result;
        return result;
}
```

2.1. Calcule o resultado de forma analítica

Para o cálculo utilizaremos o método de integração por partes:

$$\int_{x=-5}^{5} x \sin(x)\, dx$$

Vamos supor que:

$$u = x \qquad v = -\cos(x)$$

$$du = 1dx \qquad dv = \sin(x)dx$$

Sabendo da igualdade

$$\int u\,dv = uv - \int v\,du$$

$$= -x\cos(x) - \int_{x=-5}^{5} -\cos(x)\,1dx$$

$$= [-x\cos(x) - (-\sin(x))]_{x=-5}^{5}$$

Aplicando os intervalos

$$= [-5\cos(5) + \sin(5)] - [-(-5)\cos(-5) + \sin(-5)]$$

Resultado:

$$= -4.754470$$

2.2. Compare os resultados dos métodos numéricos para a mesma quantidade de pontos

Para cada um dos métodos, há um certo número mínimo de pontos para que possam ser aplicados de forma composta. Para o método do trapézio, a quantidade de pontos necessária para que ele funcione é de 2 + K. Para o método 1/3 Simpson, essa quantidade é dada por 3 + 2K e para o método 3/8 Simpson, essa quantia é de 4 + 3K. Em todos os três métodos esse K é um número natural maior ou igual a 0.

Como devemos usar a mesma quantia de pontos para a aplicação dos métodos nessa questão, vamos escolher a quantia mínima que faça com que todos os três funcionem. O método do trazpézio cresce de 1 em 1, 1/3 de Simpson cresce de 2 em 2, e o 3/8 de Simpson de 3 em 3, fazendo algumas observações, conclui-se que o número de pontos que faz com que todos funcionem simultâneamente é dado por 6X + 1, onde X é um natural maior que zero. Assim o número de pontos que será utilizado para comparar os resultados será de 7.

Os resultados são elencados a seguir, como há 7 pontos, tem-se então 6 intervalos para calculo do valor da integral.

Resultado real: -4.754470

Trapézio: * Resultado aproximado da integral: -4.57841

1/3 de Simpson: * Resultado aproximado da integral: -4.46389

3/8 de Simpson: * Resultado aproximado da integral: -2.15408

Dado o número de pontos considerados, o método que melhor se aproximou do valor real foi o do trapézio, isso pode ser explicado levando-se em consideração a forma de onda da função no intervalo considerado. Para um número maior de pontos os outros dois métodos tendem a ser cada vez mais eficazes.T

2.3. Exemplifique com um gráfico hachurado cada uma das fórmulas compostas (plote o gráfico de $f(x)$ junto com os polinômios interpoladores)

A seguir são aparesentados os três gráficos referentes à aplicação composta dos métodos de integração.

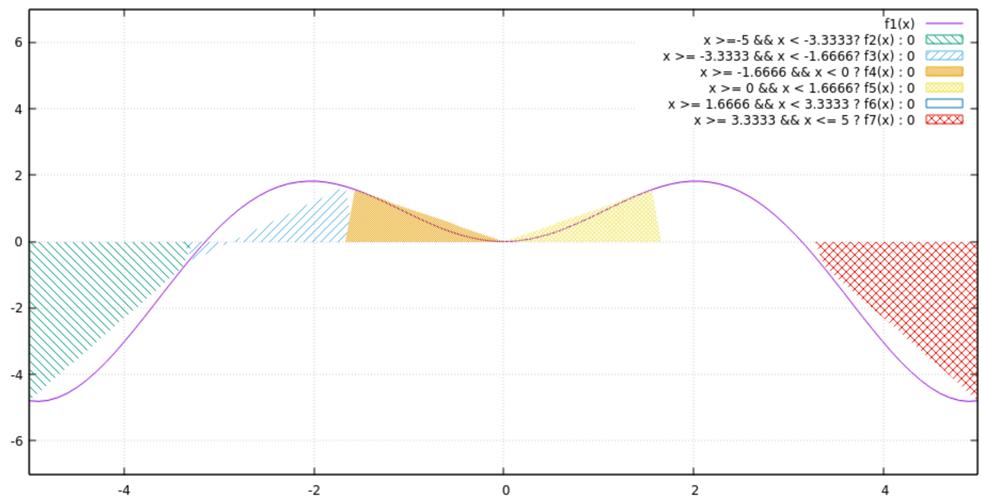

Figura 1. Método do trapézio, 6 aplicações

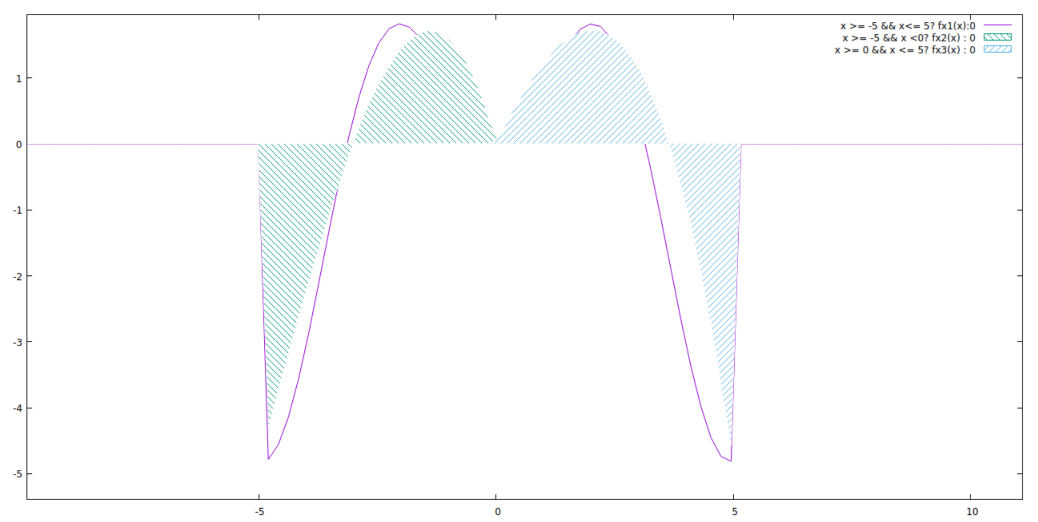

Figura 2. Método de 1/3 de Simpson, 3 aplicações

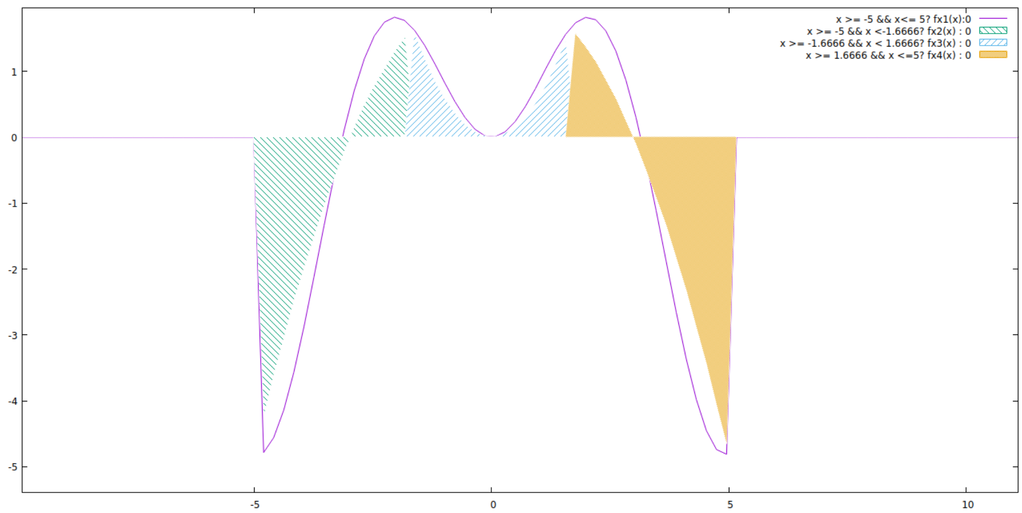

Figura 3. Método de 3/8 de Simpson, 2 aplicações

Pode-se notar que a soma das áreas das seis funções (retas) geradas pela interpolação dos 7 pontos ao longo do intervalo de -5 a 5, cobrem bem a figura da função que estamos analisando (x*sin(x)). Fazendo jus ao bom resultado do método na aproximação da área real da função.

Em segundo lugar, na qualidade da aproximação, vem o método 1/3 de Simpson, que usando 7 pontos é aplicado 3 vezes. E com um pior desempenho, para 7 pontos, há o método 3/8 de Simpson, que é aplicado 2 vezes, gerando assim duas funções da interpolação dos seus pontos em cada uma das aplicações.