# Capítulo 5

# Integração numérica

Neste capítulo iniciamos a aplicação da linguagem Python para implementar algoritmos numéricos que resolvem integrais. Abordaremos os seguintes métodos numéricos: método do retângulo, método do ponto central, método trapezoidal, método de Simpson e o método de Monte Carlos.

## 5.1 Métodos de Integração Numérica

Existem vários métodos de Integração numérica. Estes métodos têm na estrutura de sua definição a ideia básica de interpretação geométrica da integral analítica a qual representa a área sob a curva no intervalo considerado. O método mais simples consiste em dividir o intervalo em espaços iguais, gerando pequenos retângulos. A ideia é então somar a área de todos os retângulos gerados, assim a soma dessas áreas irá se aproximar da área analítica quanto menor for o espaçamento entre os retângulos. Numa função qualquer, a ideia do retângulo pode superestimar o valor da área da função, quando o retângulo passa por cima da função, ou pode subestimar, quando o retângulo passa por baixo da função (ver Figura 5.1). Neste livro vamos abordar os seguintes métodos numéricos, cada um com implementações que visam eliminar o erro por superestimação ou subestimação.

- Método do retângulo

$$I(f) = \int_a^b f(x)dx \approx h\sum_{\mathrm{i}=1}^{N} f(x_{\mathrm{i}})$$

- Método do ponto central

$$I(f) = \int_a^b f(x)dx \approx h\sum_{i=1}^{N} f\left(\frac{a+b}{2}\right)$$

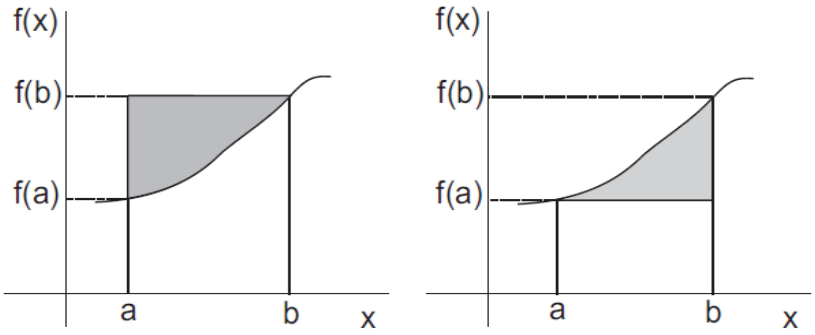


**Figura 5.1 –** Área do retângulo acima da curva: (b - a)f(b) superestimando a área (figura à esquerda). Área retângulo abaixo da curva (b - a)f(a) subestimando a área. A parte hachurada representa o quanto da área é superestimada ou subestimada, respectivamente, com a aproximação retangular.

- Método trapezoidal simples

$$I(f) = \int_a^b f(x)dx \approx \frac{[f(a)+f(b)]}{2}(b-a)$$

- Método trapezoidal composto

$$I(f) = \int_a^b f(x)dx \approx \frac{h}{2}[f(a)+f(b)] + h\sum_{i=1}^{N} f(x_i)$$

- Método de Simpson 1/3

$$I(f) = \int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 4\sum_{i}^{N} f\left(x+\frac{h}{2}\right) + f(b)\right]$$

- Método de Simpson 1/3 composto

$$I(\mathrm{f}) = \int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 4\sum_{i=1}^{n/2} f(a+(2i-1)h) + 2\sum_{i=1}^{n/2-1} f(a+2ih) + f(b)\right]$$

O número de subintervalo (n) no intervalo [a, b] deve ser um número par.

### 5.1.1 O método do Retângulo

Como o nome indica, o método consiste em aproximar a área embaixo da curva, ou seja, a integral no intervalo, pela área do retângulo formado pela largura do intervalo e o valor da função, considerada constante durante todo o intervalo, sendo assim a altura do retângulo. Considere a integral no intervalo [a, b], a integral aproximada da função **f** (x) neste intervalo é dada por:

$$I(f) = \int_{a}^{b} f(a)dx \approx f(a)(b-a) = hf(a)$$

onde h = b-a é o espaçamento ou subintervalo discreto que separa os pontos **a** e **b**, ver Figura 5.2. Podemos considerar ainda o valor da função no ponto final **b**, a integral é então calculada por:

$$I(f) = \int_{a}^{b} f(b)dx \approx f(b)(b-a) = hf(b)$$

No primeiro caso, utilizando-se o valor da função no primeiro ponto **f** (a) temos que o valor da área sob a curva é subestimado, pois, a área do retângulo formado é menor que o valor da área que queremos. No segundo caso, utilizando-se o valor da função no segundo ponto **f** (b) o valor da área é superestimado, pois, neste caso a área do retângulo é maior que a área abaixo da curva. Você logo teria uma solução razoável, utilizar o valor médio da função entre a e b, ou seja, **f** ((a+b) / 2), esta solução é feita no método do ponto central, dado pela equação:

$$I(f) = \int_{a}^{b} f(x)dx \approx h\sum_{i=1}^{N} \mathrm{f}\left(\frac{a+b}{2}\right)$$

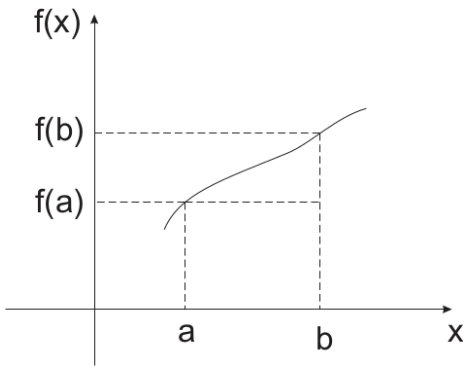


**Figura 5.2 –** Método do retângulo, a aproximação considera a área embaixo da curva dada pela área do retângulo de largura (b - a) e altura f (a) ou f (b).

### 5.1.2 O método do trapézio

Vamos considerar o cálculo da integral

$$\int_a^b f(x)dx.$$

O método considera o uso de **f** (x) em valores de x igualmente espaçados, passo h. São utilizados N pontos $x_i$ (i = 1, ..., N) no intervalo [a, b], incluindo os pontos extremos, ou seja, há N – 1 intervalos de largura h.

$$h = \frac{b-a}{N-1}$$

$$x_i = a + (i-1)h, \qquad i = 1,...,N.$$

Como começamos nossa contagem de 1, o método considera **um número ímpar de pontos** N. Na sequência, vamos utilizar a notação simplificada $\mathbf{f}_i = \mathbf{f}(x_i)$.

A ideia básica é considerar a aproximação da curva em cada intervalo como a área de um trapézio de largura h. A área de um único trapézio é

$$A = \frac{1}{2}(x_{i+1} - x_i)(f_{i+1} + f_i) = \frac{h(f_{i+1} + f_i)}{2}$$

A integral no intervalo é dada pela soma das áreas dos trapézios,

$$\int_a^b f(x)dx \approx A_1 + A_2 + ... + A_{N-1}.$$

Para pontos igualmente espaçados, a equação geral da integral será

$$\int_a^b f(x)dx \approx \frac{h}{2} f_1 + hf_2 + hf_3 + ... + hf_{N-1} + \frac{h}{2} f_N.$$

Note que como cada ponto no intervalo é contado duas vezes, eles possuem um peso h, enquanto os pontos extremos são contados uma única vez, eles têm peso h/2.

O método pode então ser escrito na forma simplificada e sucinta como:

$$\int_a^b f(x)dx \approx \frac{h}{2}(f_1 + f_N) + h\sum_{i=2}^{N-1} f_i.$$

onde $\mathbf{f}_1 = \mathbf{f}(a)$ e $\mathbf{f}_N = \mathbf{f}(b)$, ou seja, os valores extremos da função no intervalo.

Lembrando mais uma vez que N deve ser um valor ímpar. O valor de $\mathbf{f}_i$ é calculado para os pontos $x_i = a + (i - 1)h$, onde i varia de $i = 2$ a $i = N-1$.

Esta integral também pode ser reescrita para integração numérica na seguinte forma alternativa:

$$\int_a^b f(x)dx \approx \frac{h}{2}(f(a) + f(b)) + h\sum_{i=2}^{N-1} f_i.$$

Sendo agora $a = a + ih$ de tal forma que $f_i = f(a + ih)$,e o valor de $i = 1,2,...,N-1$. As duas formas são equivalentes, sendo apenas escritas de forma diferente na recorrência.

**Exemplo:** Calcular o valor numérico da integral da função: $(3/2\ \text{sen}^3(x))$. O valor desta integral pode ser determinado analiticamente tendo um valor igual a 2. Queremos determinar numericamente:

$$\int_0^{\pi} \frac{3}{2} sen^3(x)dx$$

**Solução:** Vamos utilizar o método do trapézio composto para solucionar este problema. O algoritmo básico é dado a seguir:

1. Definir a função a ser integrada f (x)
2. Entrar com os limites da integração a e b
3. Entrar com o número ímpar de pontos N
4. Definir a função trapézio (valor inicial, valor final, N)
   Início da função

   Calcular o valor do passo

$$h = \frac{b-a}{N-1}$$

   Calcular

$$soma = ( f (a) + f (b) ) / 2$$

   Faça para i = 2 até i = N-1

$$soma = soma + f(a+(i-1)h)$$

   Retornar o valor da soma
   Fim da função

5. Imprimir o valor retornado da função trapézio.

O algoritimo anterior e implementado no programa integra_trapezio.py a seguir.

```
#integra_trapezio.py
'''Metodo do trapezio para integração
Programador: Elinei Santos
Data da ultima revisão:03/09/2017'''
from numpy import *
def funcao(x):
    return (3./2)*pow(sin(x),3.0)
def trapezio(a, b, N):
    h = (b-a)/(N-1) #passo da integração
```

```
    soma = (funcao(a)+funcao(b))/2 # valores extremos da funcao
    for i in range(2, N-1):
        soma += funcao(a+(i-1)*h)
    return h*soma
a = 0.
b = pi
N = 1201 #numero impar de pontos
print(trapezio(a,b,N))
```

Podemos verificar a dependência da aproximação numérica com o argumento do valor do número de subintervalos N. Reutilizamos o programa anterior e implementamos o programa *trapezioerro.py* que chama a função trapézio em um laço for variando o número de intervalos N.

```
#trapezio_erro.py
'''Determina o erro no etodo do trapezio para integração
Programador: Elinei Santos
Data da ultima revisão:03/09/2017'''
from numpy import *
def funcao(x):
    return (3./2)*pow(sin(x),3.0)
def trapezio(a, b, N):
    h = (b-a)/(N-1) #passo da integração
    soma = (funcao(a)+funcao(b))/2 # valores extremos da funcao
    for i in range(2, N-1):
        soma += funcao(a+(i-1)*h)
    return h*soma
a = 0.
b = pi
for N in 5, 11, 51, 101, 501, 1001: #Numeros impar de pontos
    aprox = trapezio(a, b, N)
    print('N = %3d, valor_aprox = %18.15f, erro = %9.2E '%\
          (N, aprox, 2-aprox))
```

O programa *trapezio_erro.py* gera o seguinte resultado:

```
N =   5, valor_aprox =  1.594617520548519, erro =  4.05E-01
N =  11, valor_aprox =  1.986343825027498, erro =  1.37E-02
N =  51, valor_aprox =  1.999977057928317, erro =  2.29E-05
N = 101, valor_aprox =  1.999998563942519, erro =  1.44E-06
N = 501, valor_aprox =  1.999999997701192, erro =  2.30E-09
N = 1001, valor_aprox =  1.999999999856323, erro =  1.44E-10
```

podemos notar que a aproximação melhora e o erro diminui à medida que o valor de N aumenta.

Exemplo: Uma partícula de massa m = 0,5kg iniciando seu movimento com uma velocidade inicial de 2m/s e submetida a uma força dada por:

$$F(t) = F_0 e^{-\gamma t} cos(\omega t + \theta)$$

em que $F_0$=1.0 N é a amplitude da força, $\theta$ =π/6 é o ângulo de fase, $\omega$ = 1,2rad/s é a frequência angular e $\Upsilon$ = 0,2 é um fator de amortecimento. A velocidade da partícula em um intervalo de tempo pode ser determinada a partir da integral:

$$v = v_0 + \frac{F_0}{m}\int_0^t e^{-\gamma t} cos(\omega t + \theta) dt$$

Utilizando o método do trapézio para resolver integrais numericamente determine 20 valores da velocidade entre os tempos t = 0 e t = 20 s. Faça um gráfico t x v.

Solução: uma vez implementado o método numérico, só temos que utilizá-lo para calcular a integral. Vamos dividir o tempo de 20s em 40 valores começando de t = 0 e acrescentando 0,5s cada vez que realizarmos a integração. Ou seja, criamos um processo de recorrência onde cada nova velocidade depende do cálculo numérico da velocidade anterior:

$$v_{i+1} = v_i + \frac{F_0}{m}\int_{t_i}^{t_{i+1}} e^{-\gamma t} cos(\omega t + \theta) dt$$

o intervalo de integração será feito em $t_{i+1}$-$t_i$=0,5s com N = 11 passos. Utilizamos poucos passos, pois o intervalo de tempo é pequeno para cada integração. O programa *velocidade_trapezio.py* implementa esta ideia. No gráfico mostrado na Figura 5.3 mostramos o comportamento da velocidade calculada numericamente em cada tempo.

```
#velocidade_trapezio.py
‘’’Metodo do trapezio para integração
Programador: Elinei Santos
Data da ultima revisão:03/09/2017’’’
from numpy import *
import matplotlib.pyplot as plt
def funcao(t):
    return ((1./0.5)*exp(-0.2*t)*cos(1.2*t+pi/6))
def trapezio(a, b, N):
    h = (b-a)/(N-1) #passo da integração
    soma = (funcao(a)+funcao(b))/2 # valores extremos da funcao
    for i in range(2, N-1):
        soma += funcao(a+(i-1)*h)
    return h*soma
a = 0.
b = pi
N = 11 #numero impar de pontos
van = 2.0
tan = 0.
for i in arange(40):
    t = tan +0.5
    v = van + trapezio(tan,t,N)
    print(v)
    van = v
    tan = t
    print(t)
    plt.scatter(t,v, color=’r’,s=10)
```

```
plt.xlabel('tempo(s)', fontsize=15)
plt.ylabel('Velocidade (m/s)', fontsize=15)
plt.show()
```

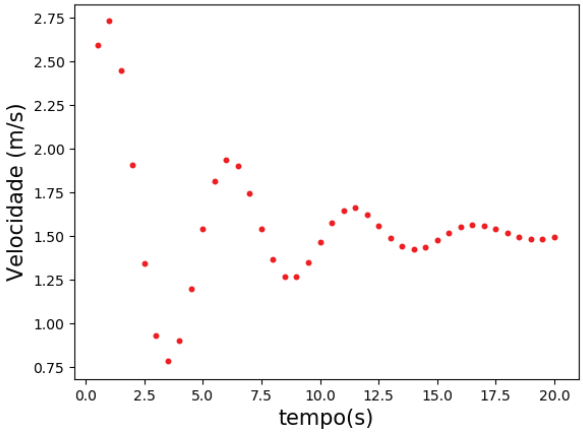


**Figura 5.3 –** Velocidade em função do tempo para uma partícula com força dependente do tempo. Cálculo da integral pelo método do trapézio.

### 5.1.3 Cálculo do período do pêndulo simples para grandes amplitudes

Quando linearizamos a equação do pêndulo simples, o que implica que sua análise só vale para pequenas oscilações, mostramos que o mesmo tem movimento periódico com períodos dado por:

$$T_0 = 2\pi\sqrt{\frac{l}{g}},$$

onde l é o comprimento do pêndulo e g a aceleração da gravidade.

Queremos, no entanto, determinar o período do pêndulo simples considerando grandes oscilações. Neste caso, não podemos linearizar a equação de movimento e lançamos mão do método da conservação da energia mecânica (Energia cinética K mais a energia pontencial V), dada por:

$$E = K + V = \frac{1}{2}I\dot{\theta}^2 - mglcos\theta = \frac{1}{2}ml^2\dot{\theta}^2 - mglcos\theta,$$

onde m é a massa do pêndulo. Vamos considerar que iniciamos o movimento do pêndulo a partir de um ângulo inicial máximo $\theta_m$. Sendo a energia mecânica conservada, temos então

$$\frac{1}{2}ml^2\dot{\theta}^2 - mglcos\theta = -mglcos\ \theta_m,$$

onde $\theta_m$ é o ângulo máximo. Explicitando o valor da velocidade angular w, temos

$$\omega^2 = \frac{2g}{1}(cos\theta - cos\theta_m),$$

Sendo $\omega = d\theta/dt$, temos

$$\left(\frac{d\theta}{dt}\right)^2 = \frac{2g}{1}(cos\theta - cos\theta_m).$$

Isolando o diferencial do tempo,

$$dt = \frac{d\theta}{\sqrt{\frac{2g}{1}(cos\theta - cos\theta_m)}}.$$

Para determinar o período do movimento, devemos integrar a equação em dt. Por simetria, podemos integrar de 0 a $\theta_m$ e multiplicar por quatro, pois, num período completo o pêndulo oscila de $\theta = \theta_m$ a $\theta = -\theta_m$ e retorna a $\theta = \theta_m$.

### Período do pêndulo simples

$$T = 4\sqrt{\frac{1}{2g}}\int_0^{\theta_m}\frac{d\theta}{\sqrt{cos\theta - cos\theta_m}}.$$

Utilizamos a identidade trigonométrica: $cos2q = 1 - 2sen^2q$, podemos reescrever a equação, obtendo

$$T = 2\sqrt{\frac{1}{g}}\int_0^{\theta_m}\frac{d\theta}{\sqrt{sen^2(\theta_m/2) - sen^2(\theta/2)}}.$$

Fazendo a mudança de variável:

$$senz = \frac{sen(\theta/2)}{sen(\theta_{\mathrm{m}}/2)}, \quad \mathrm{x} = sen(\theta_{\mathrm{m}}/2); \quad sen\left(\frac{\theta}{2}\right) = xsenz$$

Temos então:

$$\cos z dz = \frac{\cos(\theta/2)}{2sen(\theta_m/2)} d\theta = \frac{\sqrt{1-x^2 sen^2 z}}{2x} d\theta.$$

Inserindo essas variáveis na equação do período, obtemos:

$$T = 2\sqrt{\frac{l}{g}} \int_0^{\theta_m} \frac{d\theta}{\sqrt{sen^2(\theta_m/2) - sen^2(\theta/2)}} = 2\sqrt{\frac{l}{g}} \int_0^{\theta_m} \frac{2x\cos z dz}{\sqrt{1-x^2 sen^2 z}} \frac{1}{\sqrt{x^2 - x^2 sen^2 z}}$$

Simplificando, temos:

$$T = 4\sqrt{\frac{1}{g}} \int_0^{\theta_{\mathrm{m}}} \frac{dz}{\sqrt{1-x^2 sen^2 z}}.$$

Para uma oscilação do pêndulo fazendo $180^0$ ($\theta_m = \pi/2$) podemos reescrever o período em função da integral elíptica de primeira espécie, dada por:

$$K(x) = \int_0^{\pi/2} \frac{dz}{\sqrt{1-x^2 sen^2 z}} \tag{5.1}$$

o período do pêndulo pode ser escrito como

$$T = 4\sqrt{\frac{l}{g}} K(x) \quad \text{ou} \quad T = \frac{2T_0}{\pi} K(x)$$

Temos que integrar numericamente a Integral 5.1, onde x = sen($\theta_m$/2). O programa `período_trapezio.py` determina o período para um ângulo máximo de $90^0$ graus utilizando o método do trapézio.

```
#periodo_trapezio.py
'''Calculo do periodo do pendulo com integracao numerica
da integral eliptica peo método do trapézio
Programador: Elinei Santos
```

```
Data da ultima revisão:03/09/2017'''
from numpy import *
a = 0.
b = pi/2.
l = 1.0
g = 9.8
N = 1701 #numero impar de pontos
def eliptica(z,te =b/2.):
    x = sin(te)
    return 1./sqrt(1-(x*sin(z))**2)
def trapezio(a, b, N):
    h = (b-a)/(N-1) #passo da integração
    soma = (eliptica(a)+eliptica(b))/2 # valores extremos da funcao
    for i in range(2, N-1):
        soma += eliptica(a+(i-1)*h)
    return h*soma
periodo =4.* sqrt(l/g)*trapezio(a,b,N)
print('O periodo do pendulo e = %.4f s\n' %periodo)
```

Para um pêndulo de comprimento $\mathbf{L}$ = 1,0 m, temos que o programa fornece:

O periodo do pendulo e = 2.3674 s

## Método de Simpson composto

Vamos implementar o método de Simpson, também denominado método de Simpson 1/3 composto. Consideramos os subintervalos igualmente espaçados e o número de subintervalos dentro do domínio de integração [a, b] deve ser um número par. O método é reescrito a seguir:

$$I(\mathrm{f}) = \int_a^b f(x)dx \approx \frac{h}{3}\left[ f(a) + 4\sum_{\mathrm{i}=1}^{n/2} f(a+(2\mathrm{i}-1)h) + 2\sum_{\mathrm{i}=1}^{n/2-1} f(a+2ih) + f(b) \right]$$

Este método é implementado em Python no programa `integra_Simpson.py` a seguir.

**Exemplo:** A fórmula de Debye para a determinação do calor específico de um sólido é dada por:

$$C_{\mathrm{V}} = 9Nk_B\left(\frac{T}{\theta_D}\right)^3 \int_0^{\theta_{\mathrm{D}}/\mathrm{T}} \frac{x^4 e^{\mathrm{x}}}{(e^{\mathrm{x}}-1)^2} dx,$$

onde N é o número de átomos, $k_B$ é a constante de Boltzmann, $\theta_D$ é a temperatura de Debye. Faça um programa em Python que imprima uma tabela com 20 valores de temperatura T começando em $T=0^0$ aumentando de vinte em vinte graus e os valores de $C_V/3Nk_B$ para o cobre ($\theta_D$ = 309 K). Em seguida, faça um gráfico de $Cv/3Nk_B$ por $T/\theta_D$ para estes 20 valores calculados. Utilize o método de Simpson para o cálculo numérico da integral. O programa *integra_Simpson.py* a seguir implementa o método de Simpson para resolver este problema. Na Figura 5.4 mostramos o resultado gerado pelo programa.

Nota: Vale destacar nos dois programas a seguir nos comandos for, por exemplo: for i range(1, int(n/2)+1), nós fazemos a conversão explícita para inteiro da divisão de n/2, esta é uma diferença crucial entre Python 3 e Python 2, pois na versão atual Python 3 a divisão de dois inteiros resulta sempre num número de ponto flutuante. Assim, se não fizéssemos esta conversão explícita o Python acusaria erro, uma vez que o comando range exige um intervalo de números inteiros, não números em ponto flutuante que resultaria de n/2.

```
#integra_Simpson.py
'''Programa que determina o calor especifico de um solido
pela formula de Debye. O calculo da integral é feito pelo
método de Simpson composto
Programador: Elinei Santos
Ultima revisão: 06/09/2017'''
from numpy import *
import matplotlib.pyplot as plt
def g(x):
```

```
    return x**4*exp(x)/(exp(x)-1.)**2
def integra_Simpson(f, a, b, n= 300):
    h = (b-a)/float(n)
    soma1 = 0
    for i in range(1, int(n/2)+1):
        soma1 += f(a + (2*i-1)*h)
    soma2 = 0
    for i in range(1, int(n/2)):
        soma2 += f(a + 2*i*h)
    somageral = (b-a)/(3.*n)*(f(a)+f(b)+4.*soma1+2.*soma2)
    return somageral
a =0.001 #valor nao nulo para nao gerar indeterminacao
b =309
Td =309.
T=1
for k in range(1,21):
    b =309./T
    Cv = 3.*((T/309.)**3)*integra_Simpson(g,a,b)
    print('T = %.2f K Cv/3Nk = %.3f ' %(T/Td,Cv))
    plt.scatter(T/Td,Cv, color='black')
    T +=20.
plt.xlabel(r'$T/T_\theta$')
plt.ylabel(r'$C_V/3Nk_B$')
plt.show()
```

A Tabela e o gráfico (Figura 5.4) são mostrados na sequência.

```
T = 0.00 K Cv/3Nk = 0.000
T = 0.07 K Cv/3Nk = 0.024
T = 0.13 K Cv/3Nk = 0.160
T = 0.20 K Cv/3Nk = 0.361
T = 0.26 K Cv/3Nk = 0.531
T = 0.33 K Cv/3Nk = 0.653
T = 0.39 K Cv/3Nk = 0.737
T = 0.46 K Cv/3Nk = 0.796
T = 0.52 K Cv/3Nk = 0.838
```

```
T = 0.59 K Cv/3Nk = 0.868
T = 0.65 K Cv/3Nk = 0.891
T = 0.72 K Cv/3Nk = 0.909
T = 0.78 K Cv/3Nk = 0.922
T = 0.84 K Cv/3Nk = 0.933
T = 0.91 K Cv/3Nk = 0.942
T = 0.97 K Cv/3Nk = 0.949
T = 1.04 K Cv/3Nk = 0.955
T = 1.10 K Cv/3Nk = 0.960
T = 1.17 K Cv/3Nk = 0.964
T = 1.23 K Cv/3Nk = 0.968
```

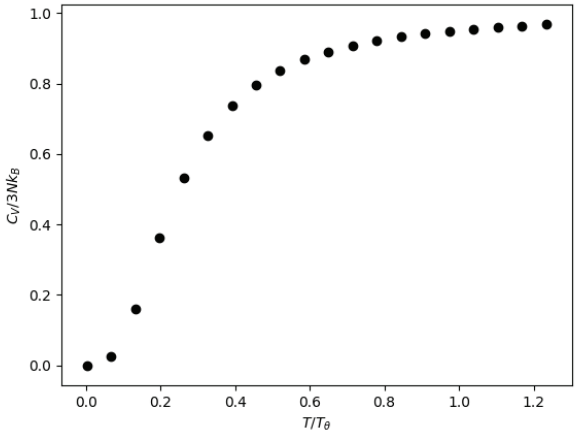


**Figura 5.4 –** Calor específico do cobre em função da temperatura ($T/T_{\boxtimes}$). Integral calculada pelo método de Simpson composto.

Exemplo: A espiral de Euler, espiral de Cornu ou clotoide é encontrada na teoria de difração de Fresnel e é gerada a partir da curva paramétrica formada a partir do cálculo das seguintes integrais:

$$S(w) = \int_0^{\mathrm{w}} sen\left(\frac{\pi x^2}{2}\right) dx$$

$$C(w) = \int_0^{\mathrm{w}} cos\left(\frac{\pi x^2}{2}\right) dx$$

Vamos utilizar o método de Simpson para calcular estas integrais e gerar a espiral de C(w) versus S(w) no intervalo de -10 < w < 10. O programa integra_fresnel.py calcula estas integrais e gera o gráfico da espiral mostrado na Figura 5.5

```
#integra_fresnel.py
'''Programa que determina a espiral de Euler ou de Cornu
O calculo da integral é feito pelo
método de Simpson composto
Programador: Elinei Santos
Ultima revisão: 06/09/2017'''
from numpy import *
import matplotlib.pyplot as plt
def c(x):
    return cos(0.5*pi*x**2)
def s(x):
    return sin(0.5*pi*x**2)
def integra_Simpson(f, a, b, n= 400): #n - numero para de pontos
    h = (b-a)/float(n)
    soma1 = 0
    for i in range(1, int(n/2)+1):
        soma1 += f(a + (2*i-1)*h)
    soma2 = 0
    for i in range(1, int(n/2)):
        soma2 += f(a + 2*i*h)
    somageral = (b-a)/(3.*n)*(f(a)+f(b)+4.*soma1+2.*soma2)
    return somageral
a =0.0 #valor inicial da integral
b =-10.
N = 1000
dx = fabs(2*b)/N
cx = []
sy = []
for k in range(1,N):
    C =integra_Simpson(c,a,b)#usa o valor default n=400 do metodo
```

```
        S =integra_Simpson(s,a,b)
        cx.append(C)
        sy.append(S)
        #plt.scatter(S,C,color='black',s=2) #para pontos individuais
calculados
        b += dx
plt.plot(cx,sy,'r')
plt.xlabel('C(x)', fontsize=15)
plt.ylabel('S(x)', fontsize=15)
plt.show()
```

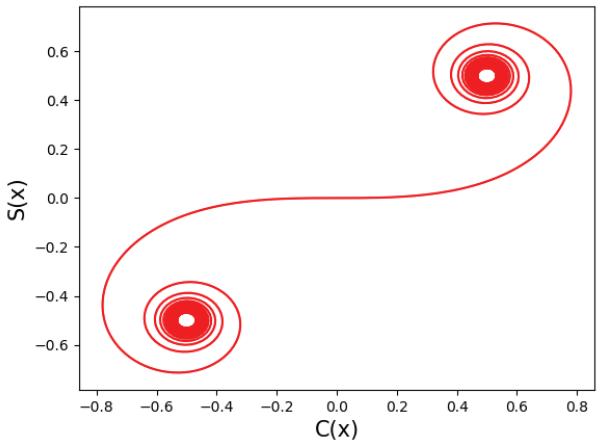


**Figura 5.5 –** Clotoide ou espiral de Euler. Cálculo das integrais pelo método de Simpson.

## 5.2 Integral utilizando o método de Monte Carlo

### 5.2.1 Método de Monte Carlo para uma dimensão

Considere uma região retangular de altura h e largura (b-a), conforme a Figura 5.6 a seguir. Esta região é separada por uma função contínua f(x). Vamos supor que geremos N pares de números aleatórios ($x_i$, $y_i$), onde os valores de $x_i$ devem estar no intervalo [a, b] ou seja: $a \leq x_i \leq b$ e $y_i$ deve estar no intervalo [0,h] ( $0 \leq y_i \leq h$). A fração n dos pontos ($x_i$, $y_i$) que satisfazem a relação: $y_i \leq f(x_i)$ é uma estimativa da área abaixo de curva f(x) (ou seja a integral de f(x)) em relação a região da área retangular. Em outras palavras,

a área estimada abaixo de f(x) é dada por:

$$E_n = \frac{n}{N} A,$$

Onde N é o número de pontos aleatórios total, n é o número de pontos que estão debaixo da curva f(x) e A é a área da região retangular (h(b-a)). Um outro exemplo: imagine que isto fosse uma figura de tiro ao alvo que você atirasse N balas e contasse quantas balas n atingem a figura abaixo da curva f(x). A área abaixo da curva é aproximadamente a fração de balas (n/N) que atingem a figura abaixo de f(x) multiplicada pela área da figura A. Esta é a ideia central do método de Monte Carlo.

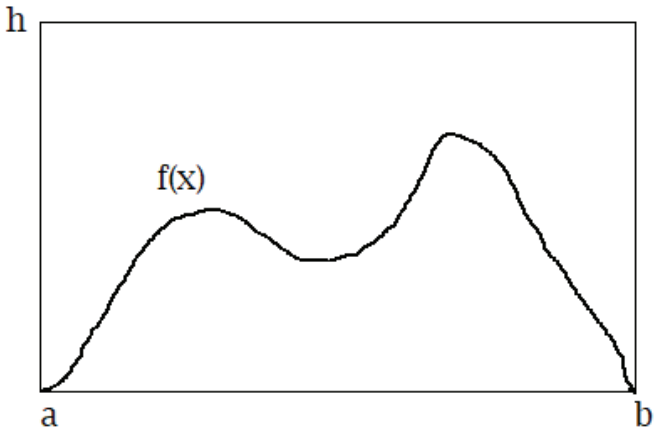


**Figura 5.6 –** Integral de f(x) é igual a área embaixo da curva f(x).

Formalmente o método de Monte Carlo para integração numérica é baseado no Teorema do valor médio do cálculo elementar:

$$I = \int_a^b f(x)dx = (b-a)\langle f \rangle$$

O teorema diz que a integral de uma função contínua f(x) entre o intervalo [a, b] (igual a área abaixo da curva) é igual ao intervalo (b-a) multiplicado pelo valor médio da função neste intervalo. O método de Monte Carlo escolhe a sequência de $x_i$ de N números uniformemente aleatórios ao invés de intervalos regulares. Assim, nós determinamos a média amostral no intervalo para esses pontos:

$$\langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

Obtemos assim a regra para a integração numérica do método:

$$I = \int_a^b f(x)dx = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) = (b-a)\langle f \rangle.$$

Onde $x_i$ são números aleatórios uniformemente distribuídos no intervalo: $a \leq x_i \leq b$ e N é o número total de tentativas. Vale lembrar que para funções em uma dimensão os métodos anteriores são mais precisos e mais rápidos. No método de Monte Carlo para uma dimensão o erro diminui com $N^{-1/2}$, não é melhor que os métodos anteriores. No entanto para dimensões maiores, o método de Monte Carlo é muito mais preciso que os outros métodos.

Exemplo: Utilizando o método de integração de Monte Carlo elaborar um programa em Python que calcule a integral:

$$I = 4\int_0^1 \sqrt{1-x^2}\,dx.$$

Esta integral tem o valor exato igual ao valor de π. Calcule o valor absoluto da diferença entre o valor da integral e o valor de π, ou seja, o erro. Faça o cálculo para 6 valores de n, começando com n = 10 e indo até $10^6$. Em seguida faça um gráfico log x log do erro em função do valor de n.

Solução: Uma possível implementação deste programa é listado a seguir no programa: *integ_montecarlo1D.py*. Na sequência, mostramos na Figura 5.7 o gráfico do erro em função do valor de n, é possível notar que a dependência do erro cai com $n^{-1/2}$.

```
#integ_montecarlo1D.py
'''Metodo de Monte Carlo para o calculo de integrais
em uma dimensao
Prog: Elinei Santos
Data da ultima revisao: 11/09/2017'''
import random
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt, pi,fabs
```

```
def g(x):
    return sqrt(1.-x**2)
def MCarlo(f, a, b, n):
    soma = 0.
    for i in  xrange(n):
        xi = random.uniform(a,b)
        soma += f(xi)
    Integ = (float(b-a)/n)*soma
    return Integ
nc = np.zeros(1000)
E = np.zeros(1000)
n=1
for i in range(6):
    n *= 10
    I = 4.*MCarlo(g,0.,1., n)
    nc[i] = n
    E[i] = fabs(I - pi)
    print('n = %d Integ=%.5f' %(n,I))
plt.loglog(nc,E,'ko')
plt.xlabel('n', fontsize=15)
plt.ylabel('Erro')
plt.show()
```

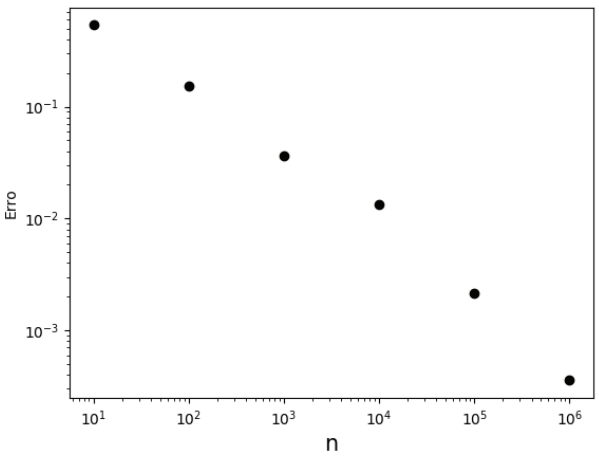


**Figura 5.7 –** Erro versus n da integral numérica calculada com o método de monte carlo. O erro cai com $n^{-1/2}$.

O valor da integral de acordo com o número de tentativas é mostrado a seguir, onde podemos verificar que o valor numérico da integral se aproxima do valor de pi.

```
n = 10 Integ=2.60535
n = 100 Integ=3.29473
n = 1000 Integ=3.10542
n = 10000 Integ=3.12811
n = 100000 Integ=3.13947
n = 1000000 Integ=3.14195
```

### 5.2.2 Método de Monte Carlo para duas dimensões

Podemos generalizar o método muito facilmente para duas ou mais dimensões, basta coletarmos pontos aleatórios no espaço adequado. No caso bidimensional, por exemplo, temos:

$$\int_a^b fx \int_c^d f(x,y)dy \approx (b-a)(d-c)\frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i) = (b-a)(d-c)\langle f \rangle.$$

Onde $(x_i, y_i)$ são números aleatórios dentro da área de integração.

## 5.2.3 Exercícios

1. Estime a integral pelo método do retângulo, trapézio, Simpson e Monte Carlo.

$$\int_0^1 e^{-x^2} dx$$

2. Implemente o método de Monte Carlos para duas variáveis. Determine analiticamente e numericamente o valor da integral:

$$\int_{-1}^{1}\int_{0}^{2} (x^2 - 2y^2 + xy^3)dxdy$$

3. Utilizando o método de Simpson composto determine o valor da integral:

$$I = \int_0^{30} 200\left(\frac{y}{5+y}\right) e^{-2y/30} dy$$

4. Determine as seguintes integrais numericamente:

$$I = \int_0^{\pi} 2sen^2 dx \quad \text{e} \quad G = \int_0^{\pi} xsen^2 dx$$

5. Usando os métodos: a) do trapézio e b) usando o método de Simpson.