IMPLEMENTAÇÃO DO MÉTODO RAYLEIGH-RITZ APLICADO A UM MODELO DE DEFLEXÃO DE VIGA USANDO O MATLAB

Yuri Elias Rodrigues[1]

Eliete Biasotto Hauser[2]

Modelagem Matemática

Resumo: Neste trabalho apresentamos os fundamentos teóricos para a construção do Método de Rayleig-Ritz, aplicado em um problema de valor de contorno, que modela matematicamente a deflexão de uma viga. O algoritmo foi implementado computacionalmente com o auxilio do software Matlab para diferentes espaçamentos de malha. Os resultados obtidos foram comparados com a solução obtida pela técnica da transformada de Laplace. Finalizamos apresentando um estudo de custo do algoritmo.

Palavras Chaves: Método de Rayleigh-Ritz, Matlab, Deflexão de Viga

INTRODUÇÃO

Concebido exclusivamente pelo físico-matemático suíço Walter Ritz (1878-1909) (LEISSA, A.W.,2005), o Método de Rayleigh-Ritz (MRR), popularizou-se academicamente com o advento dos computadores, tendo êxito em problemas nas áreas de engenharia mecânica, aerodinâmica, mecânica quântica e etc.

O MMR, assim como outros métodos variacionais, busca determinar os pontos críticos de um funcional, cujo domínio é um espaço vetorial, com imagem em um corpo de escalares. Trata-se de um problema de minimização, no qual determina-se aproximadamente os pontos estacionários do funcional envolvido, encontrando a solução.

Para analisar a eficiência implementamos o algoritmos usando o software de álgebra linear numérica Matlab de duas maneiras que se diferenciam pelo aproveitamento da estrutura da matriz associada ao problema.

[1] Licenciando em Matemática. Pontifícia Universidade Católica do Rio Grande do Sul. yuri.rodrigues@acad.pucrs.br
[2] Doutora em Matemática Aplicada. Pontifícia Universidade Católica do Rio Grande do Sul. eliete@pucrs.br

O MÉTODO DE RAYLEIGH-RITZ

Descrição do Método de Rayleigh-Ritz sobre o modelo de deflexão de viga

A técnica variacional requer a escolha funções de base usadas na construção da solução que satisfazem certas condições. Descrevemos o problema de valor de contorno (PVC), que modela a deflexão de uma viga, $y(x)$, com secção transversal $q(x)$, sujeitas as tensões $p(x)$ e $f(x)$, com extremidades fixas, expresso por

$$\begin{cases} -\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right) + q(x)y = f(x), \qquad 0 \le x \le 1, \\ y(0) = y(1) = 0. \end{cases} \tag{1}$$

A solução exata de (1), é aproximada por $y(x) \cong \phi(x)$, com

$$\phi(x) = \sum_{i=1}^{n} \phi_i\,(x)c_i. \tag{2}$$

Para a construção das funções de base ϕ_i, é necessário partir o domínio da equação (1) selecionando os valores $x_0, x_1, \ldots, x_{n+1}$

$$0 = x_0 < x_1 < \cdots < x_n < x_{n+1} = 1,$$

com $h_{i-1} = x_i - x_{i-1}$ e $i = 1 \ldots n$, temos que

$$\phi_i(x) = \begin{cases} 0, \ se \ \ 0 < x \le x_{i-1} \\ \frac{x - x_{i-1}}{h_{i-1}}, \ se \ \ x_{i-1} < x \le x_i \\ \frac{x_{i+1} - x}{h_i}, \ se \ \ x_i < x \le x_{i+1} \\ 0, \ se \ \ x_{i+1} < x \le 1, \end{cases} \tag{3}$$

ilustradas na Figura 1.

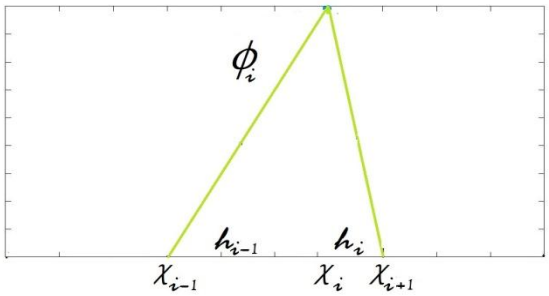


Figura 1: função de base ϕ_i

Como as funções de base ϕ_i são obtidas de (3), em (2) é necessário calcular os coeficientes c_i. Para determinar esses coeficientes, o MRR exige minimizar o funcional associado à equação (1),

$$I(y) = \int_0^1 \{p(x)\left[\frac{dy}{dx}\right]^2 + q(x)[y]^2 - 2f(x)y(x)\}dx. \quad (4)$$

Para tanto, substituímos a equação (2) na equação (4) obtendo:

$$\int_0^1 \{p(x)\left[\sum_{i=1}^n \frac{d\phi_i(x)}{dx} c_i\right]^2 + q(x)[\sum_{i=1}^n \phi_i\,(x)c_i]^2 - 2f(x)\sum_{i=1}^n \phi_i\,(x)c_i\}dx \quad (5)$$

O valor mínimo do funcional ocorre quando sua derivada parcial em relação á c_i é igual a 0, ou seja

$$\frac{\partial I}{\partial c_j} = 0. \quad (6)$$

Então,

$$\int_0^1 f(x)\phi_j(x)dx = \sum_{i=1}^n \int_0^1 \{p(x)\frac{d\phi_i(x)}{dx}\frac{d\phi_j(x)}{dx} + q(x)\phi_i(x)\phi_j(x)\}dx. \quad (7)$$

A equação (7) pode ser representada matricialmente na forma

$$A.C = B, \quad (8)$$

em que A é uma matriz tridiagonal de ordem n × n, e seus elementos não nulos são expressos por

$$a_{i,j} = \int_0^1 \left\{p(x)\frac{d\phi_i(x)}{dx}\frac{d\phi_j(x)}{dx} + q(x)\phi_i(x)\phi_j(x)\right\}dx. \quad (9)$$

Dos elementos da matriz, os valores não nulos ocorrem quando o produto das funções de base é não nulo, desta forma temos que

$$a_{i,i} = \int_0^1 \left\{p(x)\left[\phi'_i(x)\right]^2 + q(x)\left[\phi'_i(x)\right]^2\right\}dx$$

$$= \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)\,dx + \left(\frac{-1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} p(x)\,dx$$

$$+ \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} (x - x_{i-1})^2 q(x)\,dx + \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)^2 q(x)\,dx$$

para $i = 1 \ldots n$,

$$a_{i,i+1} = \int_0^1 \{p(x)\phi'_i(x)\phi'_{i+1}(x) + q(x)\phi'_i(x)\phi'_{i+1}(x)\}dx$$

$$= -\left(\frac{1}{h_i}\right)^2 \int_{x_{i-1}}^{x_i} p(x)\,dx + \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)(x - x_i) q(x)\,dx$$

para $i = 1 \ldots n - 1$,

$$a_{i,i-1} = \int_0^1 \{p(x)\phi'_i(x)\phi'_{i-1}(x) + q(x)\phi'_i(x)\phi'_{i-1}(x)\}dx$$

$$= -\left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)\,dx + \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_i}^{x_{i-1}} (x_i - x)(x - x_{i-1}) q(x)\,dx,$$

para $i = 2 \ldots n$.

De forma semelhante para os elementos do vetor B, podemos colocar escrever a integral apenas em termo dos seus valores não nulos.

$$b_i = \int_0^1 f(x)\phi_i(x)dx. \tag{10}$$

Com elementos não nulos:

$$b_i = \frac{1}{h_{i-1}} \int_{x_{i-1}}^{x_i} (x - x_{i-1}) f(x)\, dx + \frac{1}{h_i} \int_{x_i}^{x_{i+1}} (x_{i+1} - x) f(x)\, dx,$$

para $i = 1 \ldots n$.

Integrais constituintes dos elementos do Sistema de Equações Lineares

Por conta dos elementos nulos produzidos pelas funções de base nos valores do fora dos intervalos $[x_{i-1}, x_{i+1}]$, com $i = 1 \ldots n$, em (9) e (10), os elementos diferentes de zero ocorrem apenas quando j é igual a i, $i - 1$ ou $i + 1$. Logo a_{ij} e b_i são descritos apenas para valores não nulos, expressos em seis tipos de integrais:

(11)

$$Q_{1,i} = \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)(x - x_i) q(x)\, dx$$

$$Q_{2,i} = \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} (x - x_{i-1})^2 q(x)\, dx$$

$$Q_{3,i} = \left(\frac{1}{h_i}\right)^2 \int_{x_i}^{x_{i+1}} (x_{i+1} - x)^2 q(x)\, dx$$

$$Q_{4,i} = \left(\frac{1}{h_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} p(x)\, dx$$

$$Q_{5,i} = \frac{1}{h_{i-1}} \int_{x_{i-1}}^{x_i} (x - x_{i-1}) f(x)\, dx$$

$$Q_{6,i} = \frac{1}{h_i} \int_{x_i}^{x_{i+1}} (x_{i+1} - x) f(x)\, dx.$$

O cálculo das integrais são uma das partes do algoritmo mais custosas, uma vez que exigem uma resolução numérica e produzem os elementos do sistema de equações lineares. Podem inclusive trazer dificuldades para a resolução do sistema, dependendo de quão bem condicionada a matriz envolvida.

Em relação as funções de base, existem variações, ao invés do uso de funções triangulares, poderia ser utilizados polinômios ou outra função, desde que esta satisfaça as exigências do método, e preferivelmente forme uma matriz com uma estrutura que forneça bons resultados.

PROBLEMA DE VALOR DE CONTORNO

Modelo de deflexão de viga e solução exata

Para avaliar a solução aproximada de (1) escolhemos uma equação diferencial que modela a deflexão de uma viga em forma de chapa, apoiada sobre seus extremos, (12), a qual obtemos a solução exata. O MRR foi aplicado ao seguinte PVC,

$$\begin{cases} -\frac{d}{dx}\left(\frac{dy}{dx}\right)+\frac{\pi^2}{4}y=\frac{\pi^2}{16}\cos\left(\frac{x\pi^2}{4}\right), & 0\le x\le 1, \\ y(0)=y(1)=0. \end{cases} \tag{12}$$

Sua solução exata foi obtida através do método da transformada de Laplace, induzindo que o PVC seja representado como uma equação algébrica, e então aplicar a transformada inversa, encontrando a solução. A gráfico esta ilustrado nas figuras 2 e 3.

(13)

$$y''(x)+\frac{\pi^2}{4}y(x)=\frac{\pi^2}{16}\cos\left(\frac{x\pi^2}{4}\right)$$

$$\mathcal{L}\{y''(x)\}+\frac{\pi^2}{4}\mathcal{L}\{y(x)\}=\frac{\pi^2}{16}\mathcal{L}\{\cos\left(\frac{x\pi^2}{4}\right)\}$$

$$s^2Y(s)-y(0)-y'(0)+\frac{\pi^2}{4}Y(s)-\frac{\pi^2}{4}y(0)=\frac{\pi^2}{16}\frac{s}{\pi^2/16+s^2}$$

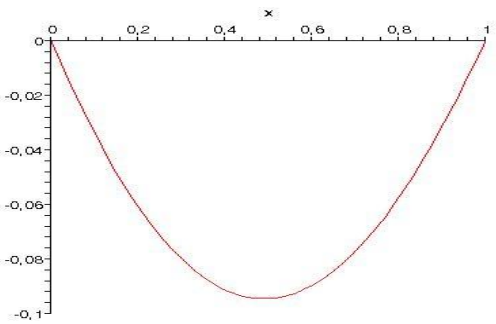
$$Y(s)=\frac{y'(0)}{s^2+\pi^2/4}+\frac{s\pi^2}{16}\frac{1}{\left(s^2+\frac{s}{s^2+\pi^2/16}\right)(s^2+\pi^2/4)}$$

Ao utilizarmos o artifício das frações parciais separamos a equação com dois termos para que possamos aplicar de uma forma mais simples a transformada inversa.

$$Y(s)=\frac{y'(0)}{(s^2+\pi^2/4)}+\frac{1}{3}\frac{s}{(s^2+\pi^2/16)}-\frac{1}{3}\frac{s}{(s^2+\pi^2/4)}$$

$$\mathcal{L}^{-1}\{Y(s)\}=\mathcal{L}^{-1}\{\frac{y'(0)}{(s^2+\pi^2/4)}\}+\mathcal{L}^{-1}\{\frac{1}{3}\frac{s}{(s^2+\pi^2/16)}\}-\mathcal{L}^{-1}\{\frac{1}{3}\frac{s}{(s^2+\pi^2/4)}\}$$

$$y(x)=-\frac{\sqrt{2}}{6}sin\left(\frac{x\pi}{2}\right)+\frac{1}{3}cos\left(\frac{x\pi}{4}\right)-\frac{1}{3}cos\left(\frac{x\pi}{2}\right).$$



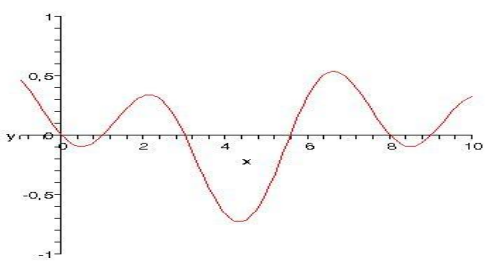


Figura 2: gráfico da solução no intervalo [0,1]

lução em um inido no PVC

ALGORITMO

O planejamento de um algoritmo tem como finalidade otimizar o tempo e espaço para obter uma solução confiável e rápido. Para avaliar o MRR, utilizamos das propriedades de pré-alocação dos espaços a serem utilizados ao longo do algoritmo e controlamos o tempo em cada uma das etapas decisivas. Colocamos no algoritmo o controle de detalhamento visual, em que é possível melhorar a qualidade da solução, em termos de quantidade de amostras. O algoritmo criado apresenta o erro absoluto no mesmo gráfico que as soluções.

Pseudoalgoritmo do Método de Rayleigh-Ritz utilizando a função de base triangular ϕ

Inserir: inteiro $n \geq 1$; pontos $x_0 = 0 < x_1 < \cdots < x_n < 1 = x_{n+1}$
Saída: coeficientes $c_1 \ldots c_n$ e aproximação para o PVC (1)
Passo 1 Definindo as diferenças entre os nodos
Para $i = 0 \ldots n$ faça $h_i = x_{i+1} - x_i$
Passo 2 Construção das funções
Para $i = 0 \ldots n$ construir as funções ϕ_i descritas em (3)
Passo 3 Cálculo das integrais
Para $i = 1,2 \ldots, n-1$ calcule o conjunto de integrais (11),
$\mathrm{Q_{1,i}}, \mathrm{Q_{2,i}}, \mathrm{Q_{3,i}}, \mathrm{Q_{4,i}}, \mathrm{Q_{5,i}}, \mathrm{Q_{6,i}}$ e $\mathrm{Q_{2,n}}, \mathrm{Q_{3,n}}, \mathrm{Q_{4,n}}, \mathrm{Q_{4,n+1}}, \mathrm{Q_{5,n}}, \mathrm{Q_{6,n}}$
Passo 4 Construção do SEL associado ao problema
para $i = 1 \ldots n$, faça $a_{i,i} = \mathrm{Q_{4,i}} + \mathrm{Q_{4,i+1}} + \mathrm{Q_{2,i}} + \mathrm{Q_{3,i}}$
para $i = 1 \ldots n-1$, faça $a_{i,i+1} = -\mathrm{Q_{4,i+1}} + \mathrm{Q_{1,i}}$
para $i = 2 \ldots n$, faça $a_{i,i-1} = -\mathrm{Q_{4,i}} + \mathrm{Q_{1,i-1}}$
para $i = 1 \ldots n$ faça $b_i = \mathrm{Q_{5,i}} + \mathrm{Q_{6,i}}$
Passo 5 Resolução do SEL exibido em (8) e obtenção dos coeficientes utilizados na aproximação da solução.

$$\mathrm{A.C} = \mathrm{B}$$

Passo 6 Construção da solução como em (2)

$$\phi(x) = \sum_{i=1}^{n} \phi_i(x) c_i$$

Implementação do Algoritmo

Na função MRR a seguir implementada no Matlab, temos que x é o vetor de nodos, os parâmetros p, q e f, são as funções $p(x)$, $q(x)$ e $f(x)$ na equação (1), a e b são os limites em (1), onde é estabelecido o contorno do problema, e N é o número de pontos para detalhamento do gráfico. Este primeiro (MRR) algoritmo não leva em conta a configuração tridiagonal que é explorada no algoritmo MRRtrid.

```
function MRR(x,p,q,f,a,b,N)
```

Declaração da váriavel simbólica

```
ponto = sym('ponto');
```

Início do contador de tempo

```
tic;n=length(x)-2;
```

Vetor para o detalhamento do gráfico e pré-alocação de memória do vetor para detalhamento

```
vec=zeros(1,N+1);
for i=0:N;
    vec(1,i+1)=a+i*(b-a)/N;
end
```

Partição do domínio

```
h=zeros(1,n+1);
for i=1:n+1;
    h(1,i)=x(:,i+1)-x(i);
end
```

Cálculo, pré-alocação de espaço para as integrais e conversão numérica

```
Q1=zeros(1,n-
1);Q2=zeros(1,n);Q3=zeros(1,n);Q4=
zeros(1,n+1);Q5=zeros(1,n);Q6=zero
s(1,n);
for i=1:n-1;

Q1(:,i)=(1/h(:,i+1))^2*int((x(:,i+
2)-ponto)*(ponto-
x(:,i+1))*q,x(:,i+1),x(:,i+2));

Q2(:,i)=(1/h(:,i))^2*int((ponto-
x(:,i))^2*q,x(:,i),x(:,i+1));

Q3(:,i)=(1/h(:,i+1))^2*int((ponto-
x(:,i+1))^2*q,x(:,i+1),x(:,i+2));

Q4(:,i)=(1/h(:,i))^2*int(p*ponto^0
,x(:,i),x(:,i+1));
    Q5(:,i)=(1/h(1,i))*int((ponto-
x(1,i))*f,x(1,i),x(1,i+1));

Q6(:,i)=(1/h(1,i+1))*int((x(1,i+2)
-ponto)*f,x(1,i+1),x(1,i+2));
end
i=i+1;

Q2(:,i)=(1/h(:,i))^2*int((ponto-
x(:,i))^2*q,x(:,i),x(:,i+1));

Q3(:,i)=(1/h(:,i+1))^2*int((ponto-
x(:,i+1))^2*q,x(:,i+1),x(:,i+2));

Q4(:,i)=(1/h(:,i))^2*int(p*ponto^0
,x(:,i),x(:,i+1));
    Q5(:,i)=(1/h(1,i))*int((ponto-
x(1,i))*f,x(1,i),x(1,i+1));

Q6(:,i)=(1/h(1,i+1))*int((x(1,i+2)
-ponto)*f,x(1,i+1),x(1,i+2));
i=i+1;
Q4(:,i)=(1/h(:,i))^2*int(p*ponto^0
,x(:,i),x(:,i+1));

Q1=double(Q1);Q2=double(Q2);Q3=dou
ble(Q3);Q4=double(Q4);Q5=double(Q5
);Q6=double(Q6);
```

Transformação de valor simbólico para valor numérico

```
Q1=double(Q1);Q2=double(Q2);Q3=dou
ble(Q3);Q4=double(Q4);Q5=double(Q5
);Q6=double(Q6);
```

Resolução e construção do sistema sem levar em conta a configuração tridiagonal e pré-alocação de espaço para o sistema de equações lineares

```
A=zeros(n);b=zeros(1,n);c=zeros(1,
n); phi=zeros(1,n);
for i=1:n;
A(i,i)=Q4(:,i)+Q4(:,i+1)+Q2(:,i)+Q
3(:,i);
end
for i=1:n-1
    A(i,i+1)=-Q4(:,i+1)+Q1(:,i);
end
for i=2:n
    A(i,i-1)=-Q4(:,i)+Q1(:,i-1);
end
A(n,n)=Q4(:,n)+Q4(:,n+1)+Q2(:,n)+Q
3(:,n);
for i=1:n;
   be(i,:)=Q5(:,i)+Q6(:,i);
end

c=A\be;
```

Funções de base, pré alocação de espaço para o vetor erro, aproximação e solução exata

```
aprox=zeros(1,N);exa=zeros(1,N);
ERROM=zeros(1,N);
for j=1:N+1;
    X=vec(1,j);
    for i=1:n;
      if and(a<X,X<=x(:,i))
          phi(1,i)=0;
      end
      if and(x(:,i)<X,X<=x(:,i+1))
```

```
            phi(1,i)=(X-
x(:,i))/h(:,i);
       end
       if
and(x(:,i+1)<X,X<=x(:,i+2))
          phi(1,i)=(x(:,i+2)-
X)/h(:,i+1);
       end
       if and(x(:,i+2)<X,X<=b)
          phi(1,i)=0;
       end
       if X==0
          phi(1,i)=0;
       end
       if X==1
          phi(1,i)=0;
       end
    end
```

Solução aproximada em cada ponto de detalhamento

```
aprox(1,j)=phi*c;
```

Solução exata em cada ponto de detalhamento

```
exa(1,j)=double(-(sqrt(2)/6)
*sin(X*pi/2)+(1/3)*cos(X*pi/4)-
1/3*cos(X*pi/2));
```

Erro absoluto em cada ponto de detalhamento

```
ERROM(1,j)=abs((exa(1,j)-
aprox(1,j)));
End
```

Cálculo do erro máximo sobre o detalhamento

```
maxE=ERROM(1,N);
for i=1:N-1;

    if ERROM(1,i)>ERROM(1,i+1);
        maxE=ERROM(1,i);
    end
end
```

Visualização do gráfico

```
plot(vec,0,vec,aprox,vec,ERROM,vec
,exa);
```

Fim do tempo

```
toc
end
```

O módulo responsável por calcular os valores das integrais é o processo mais longo, pois envolve um grande número de operações entre variáveis simbólicas e sua conversão. No algoritmo MRRtrid, usaremos a mesma estrutura do algoritmo anterior, com exceção do módulo responsável pela construção e solução do SEL, que será substituído pelo módulo a seguir, este módulo tende a ser melhor quanto maior for a quantidade de informação.

Resolução utilizando as propriedades do SEL

```
for i=1:n-1;
alpha(1,i)=Q4(:,i)+Q4(:,i+1)+Q2(:,
i)+Q3(:,i);
  beta(1,i)=Q1(:,i)-Q4(:,i+1);
   be(1,i)=Q5(:,i)+Q6(:,i);
 end
i=n;
alpha(1,i)=Q4(:,i)+Q4(:,i+1)+Q2(:,
i)+Q3(:,i);
be(1,i)=Q5(:,i)+Q6(:,i);
   aaa(1,1)=alpha(1,1);
  zeta(1,1)=beta(1,1)/alpha(1,1);
 z(1,1)=be(1,1)/aaa(1,1);
for i=2:n-1;
  aaa(1,i)=alpha(1,i)-beta(1,i-
1)*zeta(1,i-1);
   zeta(1,i)=beta(1,i)/aaa(1,i);
   z(1,i)=(be(1,i)-beta(1,i-
1)*z(1,i-1))/aaa(1,i);
end
 aaa(1,n)=alpha(1,n)-beta(1,n-
1)*zeta(1,n-1);
 z(1,n)=(be(1,n)-beta(1,n-
1)*z(1,n-1))/aaa(1,n);
    c(1,n)=z(1,n);
   for i=n-1:-1:1;
       c(1,i)=z(1,i)-
zeta(1,i)*c(1,i+1);
   end;
c=c'
```

Resultados

Considerando respectivamente, $n = 2$, $n = 10$ e $n = 100$, na figura 4, representamos graficamente a solução, $\phi(x)$ (em azul), a solução exata $y(x)$ (em vermelho) e o erro absoluto (em verde).

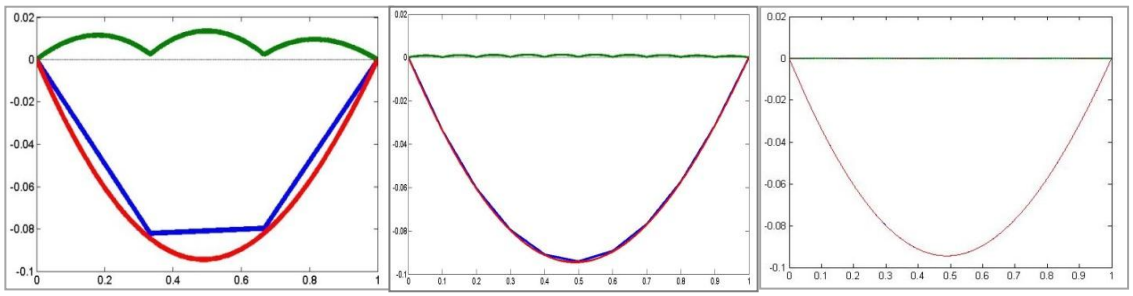


Figura 4: Comparação de soluções

Os erros absolutos, $|y(x) - \phi(x)|$, são apresentados na Tabela 1, e sugerem a convergência de $\phi(x)$ para $y(x)$, à medida que refinamos a malha.

x_i	$n = 2$	$n = 10$	$n = 100$
0	0	0	0
0.1	0.009167024934768	0.089846820120171e-003	0.090216550822225e-005
0.2	0.011368193908847	0.163243032191063e-003	0.163918728198625e-005
0.3	0.005999651817989	0.217150022943119e-003	0.218052744206665e-005
0.4	0.009576069892443	0.249465900733437e-003	0.250505960618097e-005
0.5	0.013502749573392	0.259095093875367e-003	0.260176914695931e-005
0.6	0.009420445730513	0.245981040327498e-003	0.247008226800238e-005
0.7	0.011035633989261	0.211101033628827e-003	0.211981447324761e-005
0.8	0.006749961283407	0.156423525877934e-003	0.157074144573055e-005
0.9	0.008857725388720	0.084829412698481e-003	0.085180767515902e-005
1	0	0	0

Tabela 1: erro absoluto

Na tabela 2 comparamos os valores aproximados obtidos para cada amostra da simulação em relação ao valor exato, podemos notar que quando usamos a malha para n=100, os resultados são exatos em uma margem de quatro casas depois da vírgula, ou se for em metros um décimo de milímetro. Para aumentarmos a precisão do sistema basta que se diminua o espaçamento da malha, em contrapartida temos o aumento do tempo de processamento da solução.

x_i	$n = 2$	$n = 10$	$n = 100$	$y(x)$
0	0	0	0	0
0.1	-0.0246	-0.0337	-0.0338	-0.0338
0.2	-0.0493	-0.0605	-0.0606	-0.0606

0.3	-0.0739	-0.0797	-0.0799	-0.0799
0.4	-0.0816	-0.0909	-0.0912	-0.0912
0.5	-0.0809	-0.0941	-0.0944	-0.0944
0.6	-0.0802	-0.0894	-0.0896	-0.0896
0.7	-0.0882	-0.0769	-0.0771	-0.0771
0.8	-0.0642	-0.0573	-0.0575	-0.0575
0.9	-0.0403	-0.0314	-0.0315	-0.0315
1	0	0	0	0

Tabela 2: aproximação e valor

Erro máximo observado

De acordo com o gráfico podemos colocar a seguinte conjectura: o erro cresce à medida que se afasta dos nodo gura 4, que foi construído sobre 2 nodos igualmente espaçados, mais 2, se levarmos em consideração os extremos exigidos pelo contorno do problema, podemos observar esta característica. Desta forma enunciamos que o erro máximo ocorre no ponto médio entre os nodos mais distante. Ou seja,

$$\varepsilon = max_{i=0\ldots n}(\frac{x_{i+1}+x_i}{2}),$$

$$Erromax = |y(\varepsilon) - \phi(\varepsilon)|.$$

ANÁLISE DO ALGORITMO

A complexidade de um algoritmo fornece ao utilizador as informações sobre qual eficiência esperar de determinada tarefa, em nosso problema estas são apenas variações do tamanho de nodos para encontrar a solução do problema. A complexidade computacional pode englobar vários parâmetros, tais como, detalhes da arquitetura da máquina, a linguagem de programação usada, o armazenamento utilizado, o código, tempo decorrido do processo e etc. Seccionamos o algoritmo em módulos para que possamos selecionar quais partes são significativas no tempo de processamento, em experiência, foi constatado que o maior tempo esta na resolução das integrais, que possuem funções de conversão de valor simbólico para numérico (*double*) e as funções integrais (*int*). Poderíamos utilizar neste ponto uma rotina que trate apenas numericamente a questão, selecionando assim a precisão que desejarmos e evitando soluções com custo temporal em excesso dado pela conversão e a função integral, porém deixando de lado a qualidade das soluções.

Módulo do cálculo das integrais

Para descrevermos a complexidade do algoritmo que compreende duas funções que não possuímos a descrição, *int* e *double*, faremos sua análise medindo fisicamente o tempo,

sem deixar de fazer algumas considerações sobre a estrutura do código. Durante a execução para evitarmos gastos com o requerimento de mais espaço para um vetor de trabalho, por exemplo, pré-alocamos a memória que será necessária para a operação.

Na tabela 3, comparamos os números de operações e funções (*int* e *double*), com a quantidade de tempo decorrido afim de destacar a influência no processo. Podemos observar o número de operações quando temos n nodos, considerando X como sendo aproximadamente o número de operações decorrentes da conversão numérica e resolução das integrais. Lembrando que existe o tempo de acesso envolvido, quanto maior a quantidade de informações armazenadas, maior é a latência da solicitação.

n	10	20	30	40	50
Tempo (s)	0,878283	30,44654	50,46964	76,87493	108,3752
Operações	545+60X	1095+120X	1645+180X	2195+240X	2745+300X
n	**60**	**70**	**80**	**90**	**100**
Tempo	143,3865	185,3115	243,8531	313,1185	400,1913
Operações	3295+360X	3845+420X	4395+480X	4945+540X	5495+600X

Tabela 3: Tempo e número de operações

Para avaliar a função é necessários saber o valor de X, e qual o tempo de acesso dependendo do tamanho da informação manipulada para obter uma determinada solução. Temos que o número de operações é

$$T(n) = 55n - 5 + 6nX,$$

como se trata de uma função linear, era esperado que a função fosse uma segmento de reta, mas por termos o custo de acesso envolvido acabamos por aumentar esse tempo a medida que n aumenta.

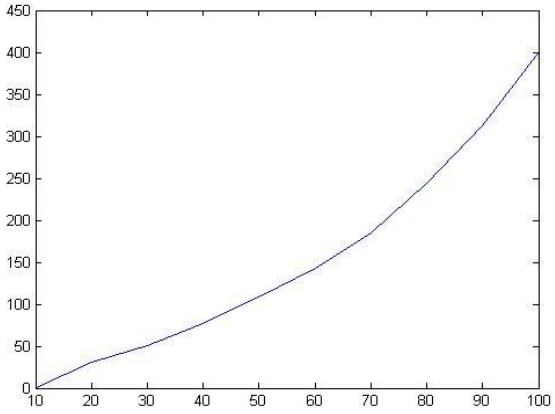


Figura 4: Tempo de processamento

Módulo da resolução do sistema de equações lineares

Aproveitando a configuração da matriz associada, tridiagonal definida positiva, utilizamos uma técnica de resolução que é apropriada a esta estrutura de banda. Em relação ao módulo anterior, temos a vantagem saber perfeitamente o número de operações, podendo fazer inclusive uma análise mais relevante. Temos que o número de operações no algoritmo MRRtrid é descrito por

$$T(n) = 13n - 7.$$

Na tabela 4 podemos verificar as relações entre a duração do processamento e a quantidade de operações realizadas. Em comparação ao módulo anterior fica claro que o tempo mais significativo fica a cargo do cálculo das integrais.

n	10	20	30	40	50
Tempo (s)	0,000196	0,000356	0,000527	0,000695	0,000889
Operações	123	253	383	513	643
n	60	70	80	90	100
Tempo	0,001295	0,001249	0,001489	0,001673	0,001865
Operações	773	903	1033	1163	1293

Tabela 4: Tempo e número de operações, MRRtrid

Existem formas não adotadas neste trabalho para contornar o cálculo das integrais, caso contrário este método não seria tão popular no período inicial da computação, em que a força bruta não era ser a estratégia a seguir. Na figura 5, é representado o tempo do algoritmo MRR(em verde) e MRRtrid(em azul).

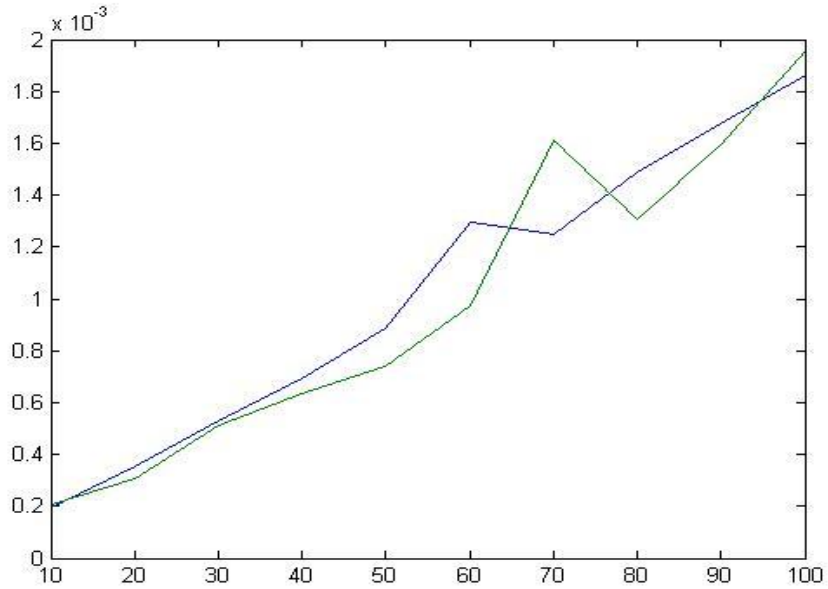


Figura 5: Tempo de resolução do sistema de equações

CONCLUSÃO

Visto as potencialidades do método, tanto quanto à possibilidade para criar variações alterando as funções de base, ou a forma como é feita a discretização do intervalo, podemos criar métodos diferentes e estudar suas características destas variações, a fim de produzir ferramentas úteis para sua finalidade que é a aplicação. Como sequencia deste trabalho vamos aumentar a dimensão do problema para 2D e avaliar as situações sobre a luz da análise de algoritmos.

REFERÊNCIAS

LEISSA , A.W. **The historical bases of the Rayleigh and Ritz methods**. [s.l.], Journal of Sound and Vibration 287, 2005, pp. 961 – 978.

ASSAN, A. E, **Análise Numérica**. São Paulo, SP: Ed. Thomson. 2003.

BURDEN, L., FAIRES, D., **Numerical analysis 9º edition**. [s.l.] Cengage Learning, 2011, pp. 673-679.

JÚDICE, J., **Sistemas de Equações Lineares**, Coimbra, Portugal, 1996, Departamento de Matemática da Universidade de Coimbra, Capítulo 5.

SCHULTZ, M., **Spline Analyses**, [s.l.], Prentice Hall, 1973, pp. 88-89.

MATHWORKS, **User's Guide - The Student Edition of Matlab, The Ultimate Computing Environment for Technical Education**, [s.l.], Prentice Hall, (1995).