UNISINOS

Métodos Numéricos
Prof. Julio Cesar Lombaldo Fernandes

Apostila de Apoio ao Ensino

UNISINOS

Junho de 2017.

Sumário

1 Prefácio

Este material tem o intúito de auxiliar o aluno na disciplina de Métodos Numéricos da Universidade do Vale do Rio dos Sinos (UNISINOS) na compreensão do conteúdo programático tendo em vista sua grande extensão. Este é um curso oferecido a estudantes de matemática, física, engenharias e outros. O Material é baseado na compreensão de problemas, sua resolução e implementação em computador. Como pré-requesito entendem-se as disciplinas de cálculo, álgebra linear e equações diferenciais. Este material está divido em teoria básica de Métodos numéricos bem como mostrada no sumário, contando com exercícios que apoiam o aluno na fixação da toeria proposta em cada capítulo. Constam aqui também alguns trechos de linguagem de programação desenvolvida em MATLAB e sempre com a `'fonte verbatim'`, mas ressaltamos que o foco continua sendo a teoria. Para implementações computacionais mais avançadas e técnicas de Métodos numéricos mais complexas recomendamos a pesquisa no livro *Numerical Recipes*.

2 Tipos de Arredondamentos e Erros

Infelizmente nem todos números são ou podem ser representados de forma exata nos computadores, o que nos levam aos possíveis erros de arredondamento. A resolução de problemas numéricos utilizando computadores estão sujeitos a esses erros. Alguns erros podem ser gerados de diferentes formas. Para o estudo dos erros devemos levar em conta algumas definições, como por exemplo, qual o critério de arredondamento de um número e os erros absoluto e relativo.

2.1 Tipos de Arredondamentos

Chamamos de arredondamento quando representamos um número com uma quantidade de dígitos menor que a de dígitos significativos (DIGSE). Tomaremos como exemplo o número

$$0,00340600.10^{-3}$$

que tem exatamento 4 dígitos significativos, pois podemos reescrevê-lo como

$$0,3406.10^{-5}$$

Podemos arredondar esse número sempre de duas maneiras: por Truncamento ou por Aproximação. Para tanto, devemos tomar como base o número de dígitos significativos (DIGSE) que desejamos utilizar nesse arredondamento.

Por Truncamento:

Se desejamos arrendondar utilizando DIGSE= 3, devemos simplismente ignorar os dígitos restantes após o terceiro significativo. Então o arredondamento de

$$0,3406.10^{-5}$$

fica

$$0,340.10^{-5}$$

Por Aproximação:

Se desejamos arrendondar utilizando DIGSE= 3, devemos olhar para o dígito seguinte a esse, ou seja, o quarto d_4. Se esse dígito for entre 0 e 4 não mudamos o terceiro dígito d_3, porém se for entre 5 e 9 adicionamos um ao terceiro dígito. Resumidamente:

Para n dígitos significativos, temos que

$$\text{Se} \quad 0 \leq d_{n+1} \leq 4, \quad d_3 \quad \text{permanece e ignoramos os posteriores} \tag{2.1}$$

$$\text{Se} \quad 5 \leq d_{n+1} \leq 9, \quad d_3 \quad \text{soma-se um a } d_3, \text{ isto é, } d_3 + 1 \tag{2.2}$$

o arredondamento por aproximação de

$$0,3406.10^{-5}$$

fica

$$0,341.10^{-5}$$

Exemplo: Represente os números $x_1 = 0,437$, $x_2 = 0,144$, $x_3 = -0,495$ e $x_4 = 0,314159265.10^{-1}$ com dois dígitos significativos por truncamento e arredondamento.

Por Truncamento:

$$x_1 = 0,43, x_2 = 0,14, x_3 = -0,49, x_4 = 0,31.10^{-1}$$

Por Aproximação:

$$x_1 = 0,44, x_2 = 0,14, x_3 = -0,50, x_4 = 0,31.10^{-1}$$

2.2 Tipos de Erros

Vamos considerar basicamente dois tipos de erros: absoluto e relativo. Estas definições nos auxiliam no estudo da tolerância na aplicação de algum método numérico, bem como na sua convergência, que veremos mais adiante.

O erro absoluto, tem como definição:

$$EA(x) = |x - \bar{x}| \tag{2.3}$$

onde x tomamos como o valor exato e $\bar{x}$ a aproximação para o mesmo.

O erro absoluto, visto sua definição, tem como principal objetivo ver o quanto distam os valores de reais e aproximados.

E o erro relativo, tem como definição:

$$ER(x) = \frac{|x - \bar{x}|}{|x|} = \frac{EA(x)}{|x|} \tag{2.4}$$

Nota-se que o erro relativo é uma medida admensional. Este conceito torna o erro relativo uma proporção com relação ao valor real.

Exemplo 1: Sejam $x = 123456,789$ e sua aproximação $\bar{x} = 123000$. O erro absoluto é

$$|x - \bar{x}| = |123456,789 - 123000| = 456,789$$

e o erro relativo é

$$\frac{|x - \bar{x}|}{|x|} = \frac{|456,789|}{|123456,789|} = 0,00369999$$

ou ainda $0,36\%$.

Exemplo 2: Sejam $x = 1,23456789$ e $\bar{x} = 1,13$. O erro absoluto é

$$|x - \bar{x}| = 1,23456789 - 1,13| = 0,10456789$$

isto é, um erro aparentemente pequeno, porém o erro relativo é

$$\frac{|x - \bar{x}|}{|x|} = \frac{|0,10456789|}{|1,23456789|} = 0,08469999$$

ou ainda $8,46\%$.

Uma maneira útil de analizarmos o quão boa pode ser a aproximação é ver o número de dígitos significativos CORRETOS da aproximação feita. Definimos da seguinte maneira:

Definição: Dizemos que um número $\bar{x}$ que representa a aproximação de x tem s dígitos significativos corretos se

$$\frac{|x - \bar{x}|}{|x|} < 5.10^{-s} \tag{2.5}$$

Podemos verificar analisando alguns exemplos:

Exemplo 1: A aproximação de $x = 0,333333$ por $x = 0,333$ tem 3 dígitos significativos corretos, pois

$$\frac{x - \bar{x}}{x} = \frac{0,333333 - 0,333}{0,333333} = 0,000999 < 5.10^{-3}$$

Exemplo 2:

Tomando agora duas aproximações $\bar{x}_1 = 0,666$ e $\bar{x}_2 = 0,667$ de $x = 0,666888$. Os erros relativos são

$$\frac{x - \bar{x}_1}{x} = \frac{0,666888 - 0,666}{0,666888} = 0,00133... < 5.10^{-3}$$

$$\frac{x - \bar{x}_2}{x} = \frac{0,666888 - 0,667}{0,666888} = 0,00167... < 5.10^{-4}$$

Aqui podemos ver que $\bar{x}_1$ possui 3 dígitos significativos corretos e $\bar{x}_2$ possui 4 dígitos significativos. Isto leva a conclusão que $\bar{x}_2$ aproxima melhor o valor de x do que $\bar{x}_1$ pois está mais próximo de x.

Cabe salientar que não existe uma única definição para o cálculo de DISGE corretos, uma das mais usadas é a expressão

$$DIGSE(x, \bar{x}) = s = int \left| \log_{10} \frac{|x - \bar{x}|}{|x|} \right| \tag{2.6}$$

2.3 Condicionamento de um Problema

O condicionamento de um problema, o qual nos reportaremos dessa maneira sempre que tivermos uma expressão a ser calculada, pode ser obtida mediante uma manipulação algébrica. Consideraremos um conjuntos de valores de entrada denotados por x e um conjuntos de valores de saída denotados por y. Em termos matemáticos, a resolução de um problema é realizada pelo mapeamento $y = f(x)$. Muitas vezes os dados de entrada do problema não são conhecidos com total exatidão. O conceito de condicionamento está relacionado à forma como os erros nos dados de entrada influenciam os dados de saída. Denotaremos por x, os dados de entrada com precisão absoluta e por x^* ou $\bar{x}$ os dados com erro. Definiremos também a solução y^* ou $\bar{y}$, do problema com dados de entrada x^*, ou seja, $y^* = f(x^*)$. Queremos saber se os erros cometidos na entrada $\Delta x = x - x^*$ influenciaram na saída do problema $\Delta y = y - y^*$. No caso mais simples, temos que $x \in \mathbb{R}$ e $y \in \mathbb{R}$. Assumindo que f seja diferenciável, a partir da série de Taylor $f(x + \Delta x) = f(x) + f'(x)\Delta x$ obtemos (subtraindo $f(x)$ dos dois lados) $\Delta y = f(x + \Delta x) - f(x) = f'(x)\Delta x$. Para relacionarmos os erros relativos, dividimos o lado esquerdo por y, o lado direito por $f(x) = y$ e obtemos

$$\frac{\Delta y}{y} = \frac{x f'(x)}{f(x)} \frac{\Delta x}{x} \tag{2.7}$$

Definição: Seja f uma função diferenciável. O número de condicionamento de um problema é definido como

$$\kappa_f(x) := \left| \frac{x f'(x)}{f(x)} \right| \tag{2.8}$$

e fornece uma estimativa de quanto os erros relativos na entrada $\left|\frac{\Delta x}{x}\right|$ serão amplificados na saída $\left|\frac{\Delta y}{y}\right|$.

Quando f depende de várias variáveis, podemos de maneira similar expressar o erro, porém absoluto, como:

$$\delta_f = |f(x_1, x_2, ..., x_n) - f(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)| \approx \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i}(x_1, x_2, ..., x_n) \right| \delta_{x_i} \tag{2.9}$$

Exemplo 1: Considerando o problema de calcular $\sqrt{x}$ em $x = 2$. Se usarmos $\bar{x} = 1,999$, quanto será o erro relativo na saída? O erro relativo na entrada é

$$\left| \frac{\Delta x}{x} \right| = \left| \frac{2 - 1,999}{2} \right| = 0,0005 \tag{2.10}$$

O número de condicionamento nesse caso é

$$\kappa_f(x) = \left| \frac{x . \frac{1}{2\sqrt{x}}}{\sqrt{x}} \right| = \frac{1}{2} \tag{2.11}$$

Ou seja, os erros na entrada serão diminuídos pela metade.

De fato, usando $y = \sqrt{2} = 1,4142136...$ e $\bar{y} = \sqrt{1,999} = 1,41386...$, obtemos

$$\frac{\Delta y}{y} = 0,000250031... \tag{2.12}$$

Exemplo 2: Seja $f(x) = xe^x$. Calcule o erro absoluto ao computar $f(x)$ sabendo que $x = 2 \pm 0,05$.

Se $x = 2$ com erro absoluto de $\delta_x = 0,05$. Neste caso, calculamos δ_f , isto é, o erro absoluto ao calcular $f(x)$, por:

$$\delta_f = |f'(x)|\delta_x$$

Como $f'(x) = (1 + x)e^x$, temos que $\delta_f = |(1 + x)e^x|\delta_x = |3e^2|0,05 = 1,1084$. Portanto, o erro absoluto ao calcular $f(x)$ quando $x = 2 \pm 0,05$ é de $1,084$.

2.4 Exercícios

1. Arredonde os seguintes números abaixo pelos critérios de truncamento e aproximação, respectivamente: (DIGSE=dígitos significativos)
   (a) 2,39939 com DIGSE 3.
   (b) π com DIGSE 5.
   (c) $\sqrt{3}$ com DIGSE 3.
   (d) $\sqrt{7}$ com DIGSE 4.
2. Represente os números 3276; 42,55 e 0,00003331 com três dígitos significativos por truncamento e aproximação.
3. Com quantos DIGSE corretos são feitas as seguintes aprocimações:
   (a) $x = 0,4444$ e $\bar{x} = 0,443$.
   (b) $x = \pi$ e $\bar{x} = 3,1413$.
   (c) $x = 0,034303.10^{-2}$ e $\bar{x} = 0,0344.10^{-2}$.
   (d) $x = 2/9$ e $\bar{x} = 0,223$.
   (e) $x = 1/7$ e $\bar{x} = 1,43.10^{-1}$
4. Ao calcularmos o valor de x^2 em $x = 4$ usando $\bar{x} = 3,92$ qual será o erro relativo na saída?
5. Se devemos calcular a expressão $f(x) = x^3 - x^2$. Calcule o erro absoluto ao efetuarmos essa operação com $x = 2 \pm 0,03$.
6. Calcule o erro absoluto ao calcularmos $f(x, y) = x^2 - 2y^2 + 2$ com $x = 2 \pm 0,01$ e $y = 3 \pm 0,05$.
7. Ao calcularmos o valor de $\ln x$ em $x = 2$ usando $\bar{x} = 1,98$ qual será o erro relativo na saída?
8. Considere um triângulo retângulo onde a hipotenusa e um dos catetos são conhecidos a menos de um erro: hipotenusa $a = 3 \pm 0,01$ metros e cateto $b = 2 \pm 0,01$ metros. Calcule o erro absoluto ao calcular a área dessa triângulo.
9. Consirando que devemos calcular as raízes de $x^2 - 5x + 6$ qual seria o erro absoluto de fizessemos o cálculo com 5,9 ao invés de 6 no termo independente?
10. A função de duas variáveis que indica as curvas de nível de uma superfície é dada por $f(x, y) = x^2 + y^2$ se usarmos os valores de $x = 2,8$ e $y = 4,01$ ao invés de $x = 3$ e $y = 4$ respectivamente, qual será o erro absoluto reotornado para essa curva de nível?

3 Solução de Equações em uma variável

Vamos tratar de buscar soluções aproximadas para equações de uma variável real, isto é, resolver aproximadamente $f(x) = 0$. Primeiramente, vamos realizar um estudo básico da existência dessas raízes em um intervalo inicial de estudo. Os métodos que desenvolveremos aqui se resumem a Bisseção, Newton, Secante e Ponto Fixo.

3.1 Existência de Raízes em um Intervalo $[a, b]$

Para analisar a existência de raízes no intervalo de estudo, vamos utilizar o teorema de Bolzano que nos fornece condições suficientes para a existência de raízes de uma função em $[a, b]$.

Teorema: *(Teorema de Bolzano) Se* $f : [a, b] \to \mathbb{R}$, $y = f(x)$ *é uma função contínua tal que* $f(a) \cdot f(b) < 0$, *então existe* $x^* \in (a, b)$ *tal que* $f(x^*) = 0$.

Demonstração: Esse fato segue imediatamente do teorema do valor intermediário que estabelece que dada uma função contínua $f : [a, b] \to \mathbb{R}$, $y = f(x)$, tal que $f(a) < f(b)$ (ou $f(b) < f(a)$), então para qualquer $d \in (f(a), f(b))$ (ou $k \in (f(b), f(a))$) existe $x^* \in (a, b)$ tal que $f(x^*) = k$. Assim, se $f(a).f(b) < 0$, então $f(a) < 0 < f(b)$ (ou $f(b) < 0 < f(a)$). Assim, tomando $k = 0$, temos que existe $x^* \in (a, b)$ tal que $f(x^*) = k = 0$ □.

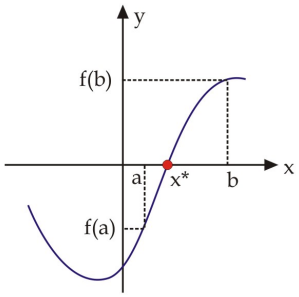


Figura 3.1: Existência de uma raiz

Podemos verificar pela figura 3.1 que existe uma raiz no intervalo com essas condições, isto é, sempre que $f(a)$ e $f(b)$ tiverem sinais contrários, teremos ao menos uma raiz no intervalo $[a, b]$.

Exemplo 1: Podemos mostrar que existe pelo menos uma raiz no intevalo $[2, 4]$ para a equação $x^2 - \ln x - 4 = 0$, pois $f(2) = -0,693$ e $f(4) = 13,38$ e portanto $f(2).f(4) = -9,23 < 0$ e satisfaz o teorema de Bolzano.

Podemos também verificar isso no MATLAB, como segue:

```
>> x=(2:0.1:4);
>> y=x.^2-log(x)-4;
```

```
>> plot(x,y)
>> grid
```

E desta forma verificamos também pelo gráfico da função como vemos na figura 3.2

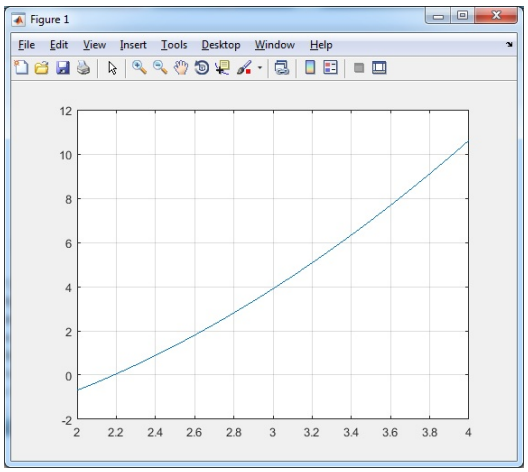


Figura 3.2: Gráfico de $f(x)$.

De maneira geral, é usual que procuremos isolar cada raiz em um único intervalo. Assim poderiamos garantir a existência e a unicidade da raiz dentro de um dado intervalo $[a,b]$. Para tanto, consideramos a seguinte proposição:

Proposição: *Se* $f : [a,b] \to \mathbb{R}$ *é uma função diferenciável,* $f(a).f(b) < 0$ *e* $f'(x) > 0$ *(ou* $f'(x) < 0$*) para todo* $x \in (a,b)$*, então existe um único* $x^* \in (a,b)$ *tal que* $f(x^*) = 0$.

Exemplo 2: Tomando a mesma função do exemplo anterior, a saber $f(x) = x^2 - \ln x - 4$, pode-se garantir que em $[2,4]$ existe apenas uma raiz. Pois $f'(x) = 2x - \frac{1}{x}$ a qual é positivo para todo $x > 1/8$ logo $f'(x) > 0$ em todo intervalo $[2,4]$. Assim, temos que existe apenas uma raiz nesse intervalo.

3.2 Método da Bisseção

Este método chamado de bisseção, trata-se de um método numérico para aproximar a raiz de uma função de uma variável em um intervalo $[a, b]$ tal que devemos garantir antes de sua aplicação a existência de uma única raiz no intervalo inicial. Este método explora o fato de que uma função contínua $f : [a, b] \to \mathbb{R}$ com $f(a).f(b) < 0$ tem um zero no intervalo (a, b) e que $f'(x) > 0$ (ou $f'(x) < 0$) nesse mesmo intervalo. A ideia para aproximar a raiz nesse caso, é seccionar o intervalo ao meio, e como primeira aproximação, tomar o ponto médio do intervalo $[a, b]$, da seguinte maneira:

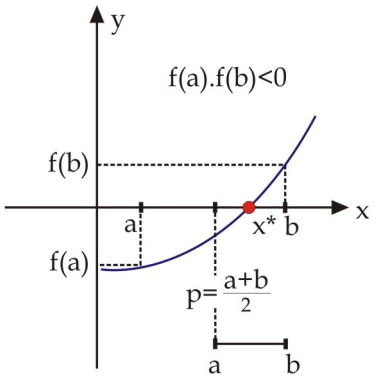


Figura 3.3: Secção do intervalo em $p = \frac{a+b}{2}$

Essa secção se repete sucessivas vezes até encontrarmos uma aproximação para a raiz com a precisão desejada, um esquema para a repetição pode ser entendido como na figura a seguir,

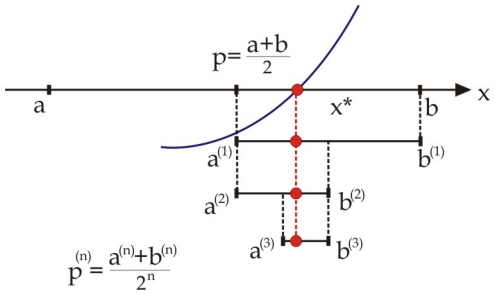


Figura 3.4: Secções sucessivas: Método da Bisseção.

Analizando os comprimentos dos sucessivos intervalos temos que $|p^{(n)} - a^{(n)}|$ ou $|p^{(n)} - b^{(n)}|$ tende a zero e vemos também que

$$|p^{(n)} - a^{(n)}| < |b^{(n)} - a^{(n)}| = \frac{|b-a|}{2^n} < TOL \tag{3.1}$$

onde TOL será a tolerância exigida com precisão deste método numérico. Sendo assim, podemos ainda isolar n tendo em vista a última equação e descobrir sem realizar iterações o número n para dado intervalo e tolerância, isto é, descobrir o número de iterações sem realizá-las.

$$\begin{aligned}
\frac{|b-a|}{2^n} &< TOL \\
\frac{|b-a|}{TOL} &< 2^n \\
\log\left(\frac{|b-a|}{TOL}\right) &< \log 2^n \\
\frac{\log\left(\frac{|b-a|}{TOL}\right)}{\log 2} &< n
\end{aligned}$$

Assim, podemos dizer que o número de iterações necessárias será descoberto mediante a relação:

$$\frac{\log\left(\frac{|b-a|}{TOL}\right)}{\log 2} < n \tag{3.2}$$

Exemplo 1: No caso queremos aproximar a raiz positiva de $f(x) = x^2 - 2$ utilizando o método da bisseção, temos que primeiramente garantir um intervalo que contenha somente essa raiz. Nesse caso, esse intervalo pode ser $[1,2]$, isto é, $a = 1$ e $b = 2$ ($f(1).f(2) < 0$ e $f'(x) > 0 \forall x \in [1,2]$). Ao iniciarmos o método da Bisseção faremos calculos sucessivos conforme a tabela 3.1.

Tabela 3.1: Esquema Método da Bisseção

n	$a^{(n)}$	$b^{(n)}$	$p^{(n)}$	$f(a^{(n)})f(p^{(n)})$	$f(p^{(n)})f(b^{(n)})$	$\lvert b^{(n)} - a^{(n)}\rvert$
0	1	2	1.5	−	+	1
1	1	1.5	1.25	+	−	0.5
2	1.25	1.5	1.375	+	−	0.25
3	1.375	1.5	1.4375	+	−	0.125
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Teremos que $p^{(n)}$ será a aproximação desejada após n iterações. Se quisermos fazer realizar as iterações com uma precisão de 10^{-4} ($TOL = 10^{-4}$) e com o mesmo intervalo inicial, fazemos

$$\frac{\log\left(\frac{|2-1|}{10^{-4}}\right)}{\log 2} = 13,28 < n \tag{3.3}$$

assim, necessitamos de $n = 14$ iterações. Nesse caso $p^{(14)} = 1,414214$.

No caso desse e de qualquer exemplo usando o método da bisseção, podemos utilizar um código em MATLAB, um dos tantos exemplos de códigos segue abaixo:

Código em MATLAB:

```
----------------------------------
function raiz=bissec(def,a,b,tol)
f=inline(def);
c=1;
if f(a)*f(b)<0
x=a;
while abs(f(x))>tol
x=(a+b)/2;
if f(a)*f(x)<0
b=x;
else a=x;
end
c=c+1;
end
raiz=x;
c
else raiz='não tem troca de sinal. Portanto não tem raiz no intervalo'
end
end
----------------------------------
```

No caso do último exemplo, precisariamos apenas difinir o intervalo inicial de iteração e chamar o código no prompt de comando do MATLAB da seguinte forma:

```
>> bissec('x^2-2',1,2,0.0001)
```

onde o programa devolve

```
c = 14
ans = 1.414214
```

3.3 Método de Newton

Vamos apresentar nessa seção o método de Newton (também conhecido como Newton-Raphson) que calcula raízes de funções de uma variável real. Sabemos que x^* é uma raiz de $f(x)$ que por sua vez é continuamente diferenciável, isto é, $f(x^*) = 0$. A ideia do método de Newton foi atravéz de um valor inicial x_0 (um chute inicial, como chamaremos também) calcular a reta tangente a função $f(x)$ nesse ponto. Por ser uma reta, obviamente ele cruza o eixo x em um dado ponto (a menos de reta horizontal). Este ponto será a primeira aproximação para a raiz e o processo é repetido conforme o esquema do figura 3.5

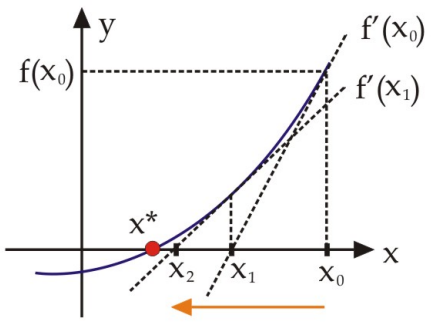


Figura 3.5: Aproximação via reta tangente: Método da Newton.

Toda equação de reta pode ser descrita por $(y - y_0) = a(x - x_0)$ em um dado ponto (x_0, y_0) onde a é a inclinação ou taxa de variação desta reta. No caso do método de Newton, temos que $a = f'(x_0)$ e aplicando a equação ao ponto de cruzamento no eixo x, isto é, $(x_1, 0)$ temos que

$$(y - y_0) = a(x - x_0) \Rightarrow (0 - f(x_0)) = f'(x_0)(x_1 - x_0) \tag{3.4}$$

o que resulta em

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

onde x_1 é a próxima aproximação para a raiz. Se aplicarmos de maneira iterativa, teremos que

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.5}$$

o que a medida que fazemos mais iterações ficamos mais próximos da raiz. Nota-se que esse método deve levar em conta o cálculo da derivada da função $f(x)$ que nem sempre será trivial.

Algumas restrições devem ser levadas em conta no método de Newton, como por exemplo, a escolha do valor inicial para as iterações, ou chute inicial.

Note que pela definição de iterações, não podemos ter x_k tal que $f'(x_k) = 0$ nas iterações.

Exemplo 1: Considerando a equação $x^2 - 2 - \ln(x) = 0$ e o problema de aproximar suas raízes. Utilizando o MATLAB para o conhecimento e localização de suas raízes, usamos os comandos: (criamos um vetor x sem valores nulos e negativos, pois pela presença de $\ln(x)$ a função não está definida para estes valores)

```
>>x=(0.01:0.01:4);
>>y=x.^2-2-log(x);
>>plot(x,y,'r-')
```

o que resulta na gráfico da figura 3.6.

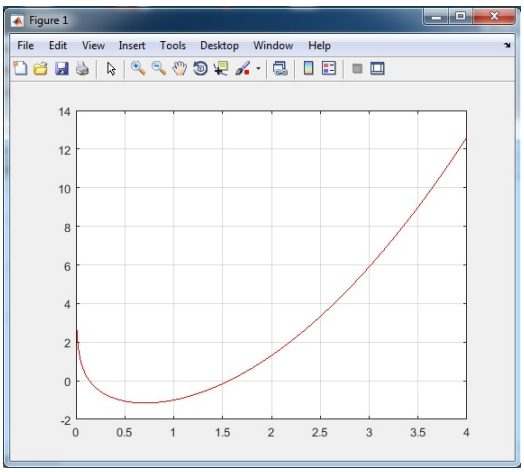


Figura 3.6: Gráfico de $f(x)$.

Pelo gráfico vemos que uma das raízes se encontra no intervalo $[0, 0.5]$ e a outra no intervalo $[1, 2]$. Para realizar as iterações via método de Newton, vamos utilizar o código em MATLAB como abaixo:

Código em MATLAB:

```
---------------------------------
function y=newton(x0,tol)
c=1;
function s=f(x)
s=x^2-2-log(x);
end
function s=df(x)
s=2*x-1/x;
end
while abs(f(x0))>tol
a=x0-f(x0)/df(x0);
x0=a;
```

```
c=c+1;
end
c
a
end
-----------------------------------
```

Ao chamarmos o código com chute inicial igual a $x_0 = 2$ e tolerância $TOL = 0.01$, por exemplo, como segue:

```
>> newton(2,0.01)
```

onde o programa devolve

```
c = 3
ans = 1.5662
```

em outras palavras aproximamos a maior raiz com 3 iterações para a tolerância imposta. Para aproximarmos a raiz mais próxima de zero, tomamos um chute inical diferente, por exemplo $x_0 = 0.3$ e mesma tolerância, chamamos então

```
>> newton(0.3,0.01)
```

onde o programa devolve

```
c = 5
ans = 0.1317
```

Assim aproximamos a menor raiz com 5 iterações para a tolerância imposta. Note que usamos o mesmo critério de convergência do método da bisseção.

Exemplo 2: Desejamos encontrar os valores da abcissa x referentes aos pontos de interseção das funções $f(x) = x^2$ e $g(x) = \ln(x+1)$ com tolerância igual a 10^{-4}. Utilizando o MATLAB para o conhecimento e localização de suas raízes. Criamos um vetor x com valores maiores que -1, pois pela presença de $\ln(x+1)$ a função não está definida para $x \leq -1$. Podemos previamente analisar o gráfico das duas funções e verificar a localização das interseções, para isso fazemos:

```
>>x=(-0.99:0.1:3);
>>y1=x.^2;
>>y2=log(x+1);
>>plot(x,y1,x,y2)
```

o que resulta nos gráficos da figura 3.7.

Onde claramente vemos que a raiz positiva está no intervalo $[0.5, 1]$. Uma raiz é zero, pois podemos de maneira trivial perceber isso ao fazermos $x = 0$, isto é, teremos $0^2 = \ln(0+1) = 0$.

Buscar o ponto de interseção entre duas curvas $f(x)$ e $g(x)$ é buscar os valores de x tais que $f(x) = g(x)$, mas podemos reescrever $f(x) - g(x) = 0$ ou ainda $h(x) = 0$ se $h(x) = f(x) - g(x)$. Logo o problema se transforma em buscar as raízes de $h(x)$. Neste exemplo, temos que

$$h(x) = x^2 - \ln(x+1)$$

Podemos aqui aplicar o método de Newton. Para tanto, precisamos da derivad de $h(x)$ que é

$$h'(x) = 2x - \frac{1}{x+1}$$

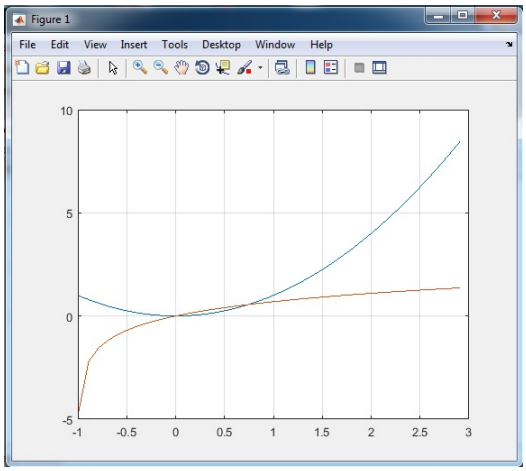


Figura 3.7: Gráfico de $f(x)$ e $g(x)$

Assim, após cadastrar dentro do código ambas, podemos chamá-lo com chute inicial $x_0 = 1$ e $TOL = 10^{-4}$. Fazemos:

```
>> newton(1,0.0001)
```

onde o programa devolve

```
c = 4
ans = 0.7469
```

desta maneira, via método de Newton, foram necessáriamos apenas 4 iterações para que com tolerância igual a 10^{-4} pudessemos aproximar a raiz em questão.

3.4 Método da Secante

A ideia básica do Método das secantes é redefinir a derivada que aparece na aproximação via método de Newton como uma secante e não mais propriamente como a derivada em x_0. Desta forma não dependeriamos da derivada, mas precisariamos de mais pontos para iniciar as iterações, isto é, x_0 e x_1 para aproximar x_2 e assim sucessivamente como mostra a figura 3.4.

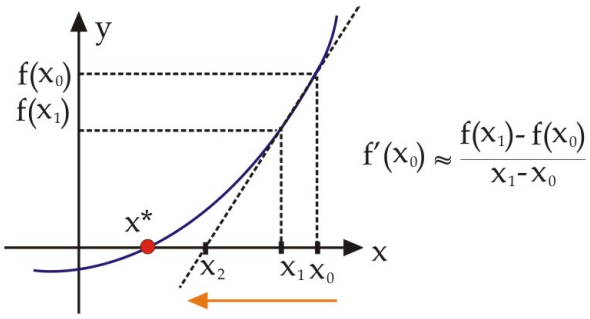


Figura 3.8: Aproximação via reta secante: Método da Secante.

Assim, matematicamente temos uma simples mudança, faremos

$$f'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{3.6}$$

o que tornam as iterações da forma:

$$x_{n+1} = x_n - \frac{f(x_n)}{\frac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}}} \tag{3.7}$$

Exemplo 1: Desejamos encontrar os valores da abcissa x referentes aos pontos de interseção das funções $f(x) = x^2$ e $g(x) = \ln(x+1)$ com tolerância igual a 10^{-3}. Trata-se de um exemplo já visto anteriormente e portanto já conhecemos a localização de suas raízes.

Vamos realizar iterações usando o código da secante em MATLAB.

Código em MATLAB:

```
---------------------------------
function y=secante(x0,x1,tol)
c=1;
function s=f(x)
s=x^2-log(x+1);
end
while abs(f(x0))>tol
a=x0-f(x0)*(x1-x0)/(f(x1)-f(x0));
x0=x1;
```

```
x1=a;
c=c+1;
end
c
a
end
---------------------------------
```

Neste caso, podemos escolher valores inciais $x_0 = 1$ e $x_1 = 2$ e chamamos

```
>>secante(1,2,0.001)
```

onde o programa devolve

```
c = 6
ans = 0.7469
```

desta maneira, via método da Secante, foram necessáriamos 6 iterações para que com tolerância igual a 10^{-3} pudessemos aproximar a raiz em questão.

Por se tratar de uma aproximação da derivada em equivalência ao método de Newton, muitas vezes o método da Secante terá uma convergência mais lenta em relação a Newton.

3.5 Método da Iteração Ponto Fixo

Este método foi criado a partir de uma modificação albébrica na equação $f(x) = 0$, a qual buscamos solução em todo esse capítulo. Esta modificação trata-se de uma explicitação da variável x fazendo $f(x) = 0$ tornar-se $g(x) - x = 0$ e consequentemente $g(x) = x$. Assim, buscar as soluções de $f(x) = 0$ torna-se equivalente a buscar soluções (interseções) de $g(x) = x$. Um ponto $x = x^*$ tal que $g(x^*) = x^*$ é chamado de ponto fixo da função $g(x)$. Geometricamente, um ponto fixo de uma função é um ponto de interseção entre a reta $y = x$ com o gráfico da função $g(x)$, isto é, temos a seguinte abordagem geométrica,

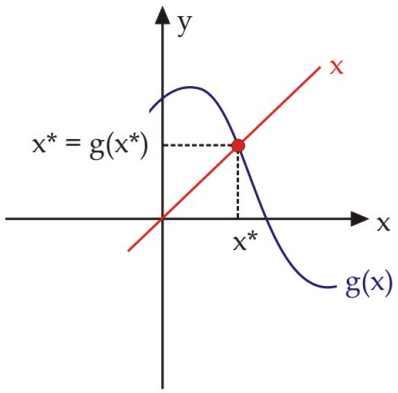


Figura 3.9: $g(x) = x$: Método Iteração Ponto Fixo.

Dada uma função $g(x)$, a iteração do ponto fixo consiste em computar a seguinte sequência recursiva:

$$x_{n+1} = g(x_n), \quad n \geq 0 \tag{3.8}$$

onde x_0 é uma aproximação inicial do ponto fixo. E sua interpretação geométrica pode ser descrita como na figura 3.10.

Exemplo 1: Resolver $e^x = x + 1$ é equivalente a resolver $f(x) = 0$ se com $f(x) = e^x - x - 1$. Nesse caso, também é equivalente a resolver $g(x) = x$, com $g(x) = e^x - 1$.

Assim, teremos que:

$$e^x = x + 1, \quad e^x - x - 1 = 0, \quad ex - 1 = x$$

Porém, devemos analisar com cuidado alguns pontos referentes ao método. Por exemplo:

- O método vai convergir sempre?
- Em caso de convergir, o limite $x^* = \lim_{n\to\infty} x_n$ é um ponto fixo?

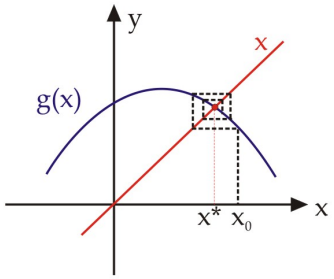


Figura 3.10: $g(x) = x$: Método Iteração Ponto Fixo (convergência geométrica).

No caso da segunda pergunta é fácil de verificar que

$$x^* = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty g(x_n)} = g(\lim_{n\to\infty} x_n) = g(x^*) \tag{3.9}$$

Porém a primeira perguntar requer algum cuidado.

Exemplo 2: Queremos aproximar a raiz de $f(x) = xe^x - 10$, isto é, as possíveis soluções de $xe^x - 10 = 0$. Logo $xe^x = 10$, ou ainda $x = 10e^{-x}$ que é equivalente a aproximar a interseção de $x = g(x)$ se $g(x) = 10e^{-x}$. Ao iniciarmos as iterações via método do ponto fixo com $x_0 = 2$, teremos que

$$\begin{aligned}
x_1 &= 10e^{-x_0} = 10e^{-2} = 1,3534 \\
x_2 &= 10e^{-x_1} = 10e^{-1,3534} = 2,5837 \\
x_3 &= 10e^{-x_2} = 10e^{-2,5837} = 0,7549 \\
x_4 &= 10e^{-x_3} = 10e^{-0,7549} = 4,7005 \\
x_5 &= 10e^{-x_4} = 10e^{-4,7005} = 0,0909 \\
\vdots &= \vdots
\end{aligned}$$

que claramente diverge. Então não é para qualquer $g(x)$ que o método da Iteração Ponto Fixo funciona. Para garantirmos que este método funciona, devemos usar o seguinte teorema, provido da definição de contração:

Definição: *Uma contração é uma função $g : [a,b] \to [a,b]$ tal que:*

$$|g(x) - g(y)| < \beta|x - y|, \quad 0 < \beta < 1$$

Teorema do Ponto Fixo: *Se $g : [a,b] \to [a,b]$ é uma contração, então existe um único ponto $x^* \in [a,b]$ tal que $g(x^*) = x^*$, isto é, x^* é ponto fixo de $g(x)$. Além disso, a sequência $\{x_n\}_{n=1}^{\infty}$ dada por:*

$$x_{n+1} = g(x_n) \tag{3.10}$$

converge para x^* para qualquer $x_0 \in [a,b]$.

A definição de contração pode ser entendida como $|g'(x)| < 1$ para todo $x \in [a,b]$, pois

$$|g(x) - g(y)| < \beta|x - y| \Rightarrow \underbrace{\frac{|g(x) - g(y)|}{|x - y|}}_{|g'(x)|} < \beta, \quad 0 < \beta < 1$$

Uma interpretação geométrica pode ser dada para esses casos (convergência e divergência) do método da iteração ponto fixo, como por exemplo na figura a seguir:

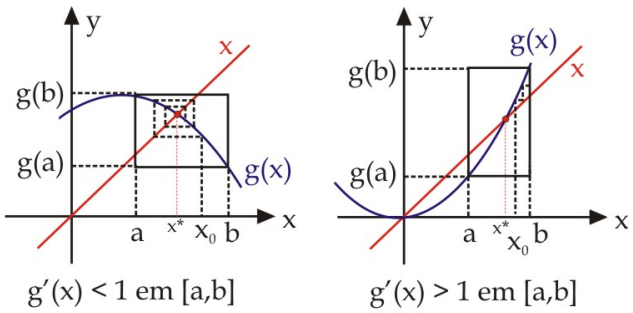


Figura 3.11: $g(x) = x$: Método Iteração Ponto Fixo $g'(x) < 1$ e $g'(x) > 1$ em $[a,b]$.

Também é fácil ver que na figura da esquerda, temos que

$$|g(b) - g(a)| < |b - a|$$

e já na figura da direita temos que

$$|g(b) - g(a)| > |b - a|$$

Exemplo 2: Vamos revisitar o exemplo anterior em que desejavamos aproximar a raiz de $f(x) = xe^x - 10$. Ao separarmos de maneira diferente para obtermos uma $g(x)$ podemos ter, por exemplo, $xe^x = 10 \Rightarrow e^x = \frac{10}{x} \Rightarrow \ln e^x = \ln\left(\frac{10}{x}\right)$ e por fim $x = \ln\left(\frac{10}{x}\right)$. Assim, temos $g(x) = \ln\left(\frac{10}{x}\right)$. Como

$$g'(x) = \frac{1}{\frac{10}{x}} \cdot \left(\frac{10}{x}\right)'$$

ou ainda

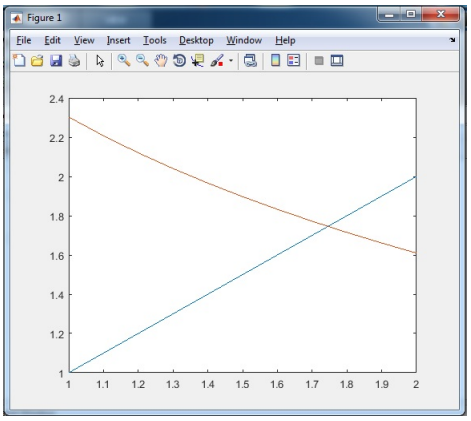
$$g'(x) = \frac{1}{\frac{10}{x}} \cdot \frac{-10}{x^2} \tag{3.11}$$

e então

$$g'(x) = \frac{x}{10} \cdot \frac{-10}{x^2} = -\frac{1}{x} \tag{3.12}$$

A interseção de $g(x)$ e x (ou raiz de $f(x)$) pode ser estimada utilizando o Matlab com os comandos

```
x=(1:0.1:2);
y1=x;
y2=log(10./x);
plot(x,y1,x,y2)
```



Claramente essa interseção encontra-se entre 1.5 e 2, o qual tomaremos por a e b, repectivamente. Agora, precisamos verificar se a função $g(x)$ satisfaz as condições do teorema do ponto fixo no intervalo $[a,b]$. Como

$$g'(x) = -\frac{1}{x^2} < 1, \forall x \in [1.5, 2]$$

Ao escolhermos um chute inicial x_0 tal que $x_0 \in [1.5,2]$ o método deverá convergir.

Usaremos um código em Matlab para calcularmos as iterações do método, como segue:

Código em MATLAB:

```
---------------------------------
function y=pfixo(x0,tol)
c=1;
function g=f(x);
g=log(10/x);
end
while abs(f(x0)-x0)>tol
a=f(x0);
x0=a;
c=c+1;
end
```

```
c
a
end
--------------------------------
```

Ao executarmos o código com chute inicial $x_0 = 2$, por exemplo, teremos como resultado:

```
>> pfixo(2,0.01)
c = 8
a = 1.7406
```

Isto é, a raiz é aproximadamente igual a 1,7406 com tolerância de 10^{-2} e foram necessarias 8 iterações para esse problema.

3.6 Exercícios

Método da Bisseção:

1. Determine via método da bisseção as soluções de $x^2-3x+1=0$ realizando 5 iterações para cada uma delas.

2. Determine via método da bisseção as soluções de $3-x^2-e^x=0$ realizando 5 iterações para cada uma delas.

3. Encontre, usando o método da bisseção os valores de x aos quais os gráficos de $f(x)=x^2-4$ e $\ln x$ se interceptam.

4. Explique porque não é possível determinar via método da bisseção as raízes de $f(x)=x^2-4x+4$.

5. Resolva os seguintes itens usando MATLAB:

   (a) Esboçe os gráficos de $f(x)=x$ e $g(x)=2\sin x$.

   (b) Utilize o método da bisseção para determinar uma aproximação com precisão de 10^{-3} do primeiro valor positivo de x tal que $f(x)=g(x)$.

   (c) Sem realizar iterações, diga quantas seriam necessárias fazer para determinar com precisão em 10^{-5} e intervalo de comprimento $0,5$ esse mesmo valor do item (b).

6. Considere a equação $\sqrt{x}=\cos(x)$. Use o método da bisseção com intervalo inicial $[a,b]=[0,1]$ e $p^{(0)}=(a+b)/2$ para calcular a aproximação $p^{(4)}$ da solução desta equação.

7. Trace o gráfico e isole as três primeiras raízes positivas da função:

$$f(x)=5\sin x^2-e^{x/10} \tag{3.13}$$

   em intervalos de comprimento 0,1. Então, use o método da bisseção para obter aproximações dos zeros desta função com precisão de 10^{-5}.

8. O polinômio $p(x)=-4+8x-5x^2+x^3$ tem raízes $x_1=1$ e $x_2=x_3=2$ no intervalo $[1/2,3]$.

   (a) Se o método da bisseção for usando com o intervalo inicial $[1/2,3]$, para qual raiz as iterações convergem?

   (b) É possível usar o método da bisseção para a raiz $x=2$? Justifique sua resposta.

9. O polinômio $f(x)=x^4-4x^2+4$ possui raízes duplas em $\sqrt{2}$ e $-\sqrt{2}$. O método da bisseção pode ser aplicados a $f(x)$? Explique.

Método de Newton:

1. Encontre, usando o método de Newton os valores de x aos quais os gráficos de $f(x) = x^2 - 4$ e $\ln x$ se interceptam com DIGSE 3.

2. Considere o problema de calcular as soluções positivas da equação:

$$\tan(x) = 2x^2$$

   (a) Use o método gráfico para isolar as duas primeiras raízes positivas em pequenos intervalos. Use a teoria para argumentar quanto à existência e unicidade das raízes dentro intervalos escolhidos.

   (b) Calcule cada uma das raízes pelo método de Newton com 5 dígitos significativos e discuta a convergência comparando com o item (b).

3. Considere a equação $e^{-x^2} = x$. Trace o gráfico com auxílio do computador e verifique que ela possui uma raiz positiva. Encontre uma aproximação para esta razão pelo gráfico e use este valor para inicializar o método de Newton e obtenha uma aproximação para a raiz com 4 dígitos significativos.

4. Verifique com auxilio do MATLAB o número de interseções de $f(x) = \sin(x)$ e $g(x) = \log x$. Utilize o método de Newton para calcular o menor valor positivo que satisfaz essa equação, se possível.

5. Considere o método de Newton aplicado para encontrar a raiz de $f(x) = x^3 - 2x + 2$. O que acontece quando $x(0) = 0$? Escolha um valor adequado para inicializar o método e obter a única raiz real desta equação.

6. Encontre as raízes do polinômio $f(x) = x^4 - 4x^2 + 4$ através do método de Newton.

Método da Secante:

1. Dê uma interpretação geométrica ao método das secantes. Qual a vantagem do método das secantes sobre o método de Newton?

2. Aplique o método das secantes para resolver a equação

$$e^{-x^2} = 2x$$

3. Existem infinitos valores positivos tais que

$$e^{-x} = \sin x$$

   (a) Use o método gráfico para verificar aproximadamente onde se encontram as interseções, bem como decida valores iniciais para iniciar suas iterações.

   (b) Aproxime via método da Secante o menor valor positivo que satisfaz essa equação com tolerância igual a 10^{-3}.

4. Verifique com auxilio do MATLAB o número de interseções de $f(x) = \cos(x)$ e $g(x) = \log x$. Utilize o método da secante para calcular os valores que satisfazem essa equação.

5. Encontre as raízes do polinômio $f(x) = x^4 - 4x^2 + 4$ através do método da Secante com tolerância de 10^{-3}.

Método da Iteração Ponto Fixo:

1. Determine uma ou mais separações algébricas para as funções abaixo com o intuito de resolvê-las via método da Iteração Ponto Fixo:
   (a) $f(x) = x^3 - x + 1$
   (b) $f(x) = 3^{-x} - x + 2$
   (c) $f(x) = x^4 - x^2 - \ln(x)$
   (d) $f(x) = xe^x - 5$
   (e) $f(x) = x^3 - \cos(x)$

2. Resolver a equação $e^x = x + 2$ é equivalente a calcular os pontos fixos da função $g(x) = e^x - 2$. Use a iteração do ponto fixo com $x_0 = -1,8$ para obter uma aproximação de uma das soluções da equação dada com tolerância igual 10^{-7}.

3. Mostre usando Matlab que a equação:

$$\cos(x) = x$$

   possui uma única solução no intervalo $[0,1]$. Use a iteração do ponto fixo e encontre uma aproximação para esta solução com 4 DIGSE.

4. Encontre a solução de cada equação com tolerância 10^-6.
   (a) $e^x = x + 2$ no intervalo $(-2,0)$.
   (b) $x^3 + 5x^2 - 12 = 0$ no intervalo $(1,2)$.
   (c) $\sqrt{x} = \cos(x)$ no intervalo $(0,1)$.

5. Encontre numericamente as três primeiras raízes positivas da equação dada por:

$$\cos(x) = \frac{x}{10 + x^2}$$

   com tolerância 10^{-5}.

4 Sistemas Lineares

A solução de sistemas lineares muitas vezes é associada a problemas reais de engenharia. Vamos estudar nesse capítulo técnicas numéricas usadas para solucionar esses sistemas. Trataremos também do conceito de condicionamento de um sistema linear e quando será possível ou não empregar tais técnicas. Os métodos a serem empregados são métodos iterativos e basicamente estudaremos aqui dois deles: Método de Jacobi e Gauss-Seidel.

Vamos inicialmente considerar o sistema linear:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

e que matricialmente podemos descrevê-lo como

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

A solução desse sistema ($Ax = b$) pode ser obtida de diversas formas analíticas, mas nem sempre de maneira trivial. Uma das técnicas a ser usada é a da eliminação gaussiana ou ainda a obtenção da sua inversa para que possamos ter $x = A^{-1}b$. Tais técnicas analíticas são empregadas costumeiramente quando temos uma matriz A com coeficientes inteiros ou com pequena variação entre si, mas em casos em que essas duas coisas não ocorrem podemos usar técinicas iterativas.

4.1 Método de Jacobi

O Método de Jacobi visa isolar na i-ézima linha a variável x_i tendo assim uma nova equação para cada linha desse sistema. Considerando novamente o sistema

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

Podemos fazer para cada linha

$$\begin{aligned}
x_1 &= -\frac{a_{12}}{a_{11}}x_2 - \ldots - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \\
x_2 &= -\frac{a_{21}}{a_{22}}x_1 - \ldots - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \\
&\vdots \\
x_n &= -\frac{a_{n1}}{a_{nn}}x_1 - \ldots - \frac{a_{n,n-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}}
\end{aligned}$$

o que em notação matricial se traduz em

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x_{n+1}} = \underbrace{\begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \cdots & -\frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & \cdots & -\frac{a_{n,n-1}}{a_{nn}} & 0 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x_n} + \underbrace{\begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{bmatrix}}_{c}$$

Assim, vamos realizar iterações para aproximar a solução de $Ax = b$ através de

$$x_{n+1} = Mx_n + c \tag{4.1}$$

As iterações serão da forma

$$x_0 = \text{Chute Inicial} \tag{4.2}$$

$$x_{j,(n+1)} = \frac{b_i}{a_{ii}} - \sum_{i=1, i\neq j}^{n} \frac{a_{ij}}{a_{ii}} x_{i,(n)}, \qquad j \in \{1, 2, \ldots, n\} \tag{4.3}$$

Segue abaixo o *Código em Matlab para o Método de Jacobi:*

```
-----------------------------------
function [X,delta,Z] = jacobi(A,b,X0,eps,max)
n=length(b);
Xant = X0;
X=X0;
Z = X0';
for k=1:max,
for j = 1:n,
Sum = b(j) - A(j,[1:j-1,j+1:n])*Xant([1:j-1,j+1:n]);
X(j) = Sum/A(j,j);
end
Z = [Z;X'];
```

```
delta = norm(abs(X-Xant),1);
if (delta<eps) break, end
Xant = X;
end
Z
end
----------------------------------
```

Onde as entradas são a matriz `A`, o vetor `b`, o chute inicial `X0`, a tolerância `eps` e o número máximo de iterações `max`.

Porém a convergência do método só é garantida se as entradas na matriz M somadas por linhas devem ser menores que o elemento da diagonal, isto é, se

$$\sum_{i=1, i\neq j}^{n} |a_{ij}| < a_{jj}, j \in \{1, 2, ..., n\} \tag{4.4}$$

Isto é, se a matriz A for diagonal dominante.

Exemplo: Considerando as matrizes A e B,

$$A = \begin{bmatrix} 4 & -1 & 1 \\ 2 & 7 & 2 \\ 2 & -1 & 9 \end{bmatrix} \tag{4.5}$$

$$B = \begin{bmatrix} 1 & -5 & 2 \\ 2 & 2 & 2 \\ 2 & -1 & 1 \end{bmatrix} \tag{4.6}$$

nesse caso A é diagonal dominante e B não.

Segue um código para a verificação se uma matriz é diagonal dominante:

```
----------------------------------
function r = diagdominante(A,n)
for i = 1:n
j = 1:n;
j(i) = [];
B = abs(A(i,j));
Check(i) = abs(A(i,i)) - sum(B);
if Check(i) < 0
fprintf('A matriz não é diagonal dominante na linha %2i\n\n',i)
else
fprintf('A matriz é diagonal dominante na linha %2i\n\n',i)
end
end
end
----------------------------------
```

Onde as entradas são a matriz `A` e a ordem `n` da mesma.

Exemplo: Considere o sistema linear abaixo:

$$\begin{aligned} 1.2x_1 + -0.4x_2 + 0.1x_n &= 0.4 \\ 0.1x_1 + 0.7x_2 + 0.21x_n &= 0.2 \\ 0.2x_1 - 0.1x_2 + 1.76x_3 &= 0.2 \end{aligned}$$

o qual podemos escrever matricialmente $Ax = b$ onde

$$A = \begin{bmatrix} 1.2 & -0.4 & 0.1 \\ 0.1 & 0.7 & 0.21 \\ 0.2 & -0.1 & 1.76 \end{bmatrix}$$

$$b = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \end{bmatrix}$$

O qual podemos verificar se é diagonal dominante através do código em Matlab, após cadastrar a matriz A e chamando

```
>> diagdominante(A,3)
```

e o programa devolve

```
A matriz é diagonal dominante na linha  1
A matriz é diagonal dominante na linha  3
A matriz é diagonal dominante na linha  3
```

Isto é, podemos resolver esse sistema via Jacobi.

Fazendo com $x_0 = (0,0,0)^T$ e tolerância igual a 10^{-4} e 20 iterações, chamamos o programa `jacobi.m` fazendo

```
>> jacobi(A,b,X0,0.0001,20)
```

onde teremos como resposta

```
Z =
0         0         0
0.3333    0.2857    0.1136
0.4191    0.2040    0.0920
0.3937    0.1982    0.0776
0.3929    0.2062    0.0802
0.3954    0.2055    0.0807
0.3951    0.2050    0.0804
0.3950    0.2052    0.0804
0.3950    0.2052    0.0804
ans =
0.3950
0.2052
0.0804
```

Ou seja, foram necessárias 8 iterações pelo método de Jacobi para aproximar com a tolerância desejada.

Assim, a solução aproximada é dada por

$$x_8 = \begin{bmatrix} 0.3950 \\ 0.2052 \\ 0.0804 \end{bmatrix}$$

4.2 Método de Gauss-Seidel

O Método de Gauss-Seidel visa solucionar também um sistema linear de forma iterativa tal qual o Método de Jacobi, mas atualizando dentro de cada iteração as variáveis já calculadas. Assim, iniciamos com o mesmo sistema de forma geral

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ \vdots &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n &= b_n \end{aligned}$$

Podemos fazer para cada linha

$$\begin{aligned} x_1 &= -\frac{a_{12}}{a_{11}}x_2 - ... - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \\ x_2 &= -\frac{a_{21}}{a_{22}}x_1 - ... - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \\ &\vdots \\ x_n &= -\frac{a_{n1}}{a_{nn}}x_1 - ... - \frac{a_{n,n-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}} \end{aligned}$$

Mas ao contrário de Jacobi, atualizaremos a cada nova linha os valores recentemente calculados. Por exemplo, na primeira iteração ao calcularmos a aproximação de x_2 usaremos já atualizada x_1 assim como no cálculo de x_3, usaremos os já atualizados x_1 e x_2. Então de maneira geral, temos que

$$x_0 = \text{Chute Inicial} \tag{4.7}$$

$$x_{j,(n)} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_{j,(n+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_{j,(n)}, \qquad j \in \{1, 2, ..., n\} \tag{4.8}$$

Matricialmente, se temos $Ax = b$, faremos $A = L + D + U$ (triangular inferior+diagonal+triangular superior)e teremos que

$$
\begin{aligned}
Ax &= b \\
(L+D+U)x &= b \\
Lx+Dx+Ux &= b \\
Dx &= -(L+U)x+b \\
x &= -D^{-1}(L+U)x+D^{-1}b \\
x &= -D^{-1}Lx-D^{-1}Ux+D^{-1}b \\
x-D^{-1}Lx &= -D^{-1}Ux+D^{-1}b \\
(I-D^{-1}L)x &= -D^{-1}Ux+D^{-1}b \\
x &= -(I-D^{-1}L)D^{-1}Ux+(I-D^{-1}L)D^{-1}b
\end{aligned}
$$

e por fim teriamos que

$$x_{n+1} = -(I-D^{-1}L)D^{-1}Ux_n+(I-D^{-1}L)D^{-1}b \tag{4.9}$$

Segue abaixo o *Código em Matlab para o Método de Gauss-Seidel:*

```
---------------------------------
function [X,delta,Z] = gseidel(A,b,X0,eps,max)
n = length(b);
Xant = X0;
X=X0;
Z = X0';
for k=1:max,
for j = 1:n,
if j==1
Sum = b(1) - A(1,2:n)*Xant(2:n);
elseif j==n
Sum = b(n) - A(n,1:n-1)*X(1:n-1);
else
Sum = b(j)-A(j,1:j-1)*X(1:j-1)-A(j,j+1:n)*Xant(j+1:n);
end
X(j) = Sum/A(j,j);
end
Z = [Z;X'];
delta = norm(abs(X-Xant),1);
if (delta<eps) break, end
Xant = X;
end
Z
end
---------------------------------
```

Onde as entradas são a matriz `A`, o vetor `b`, o chute inicial `X0`, a tolerância `eps` e o número máximo de iterações `max`.

Exemplo: Se tomarmos o mesmo exemplo da seção anterior

$$\begin{aligned} 1.2x_1 + -0.4x_2 + 0.1x_n &= 0.4 \\ 0.1x_1 + 0.7x_2 + 0.21x_n &= 0.2 \\ 0.2x_1 - 0.1x_2 + 1.76x_3 &= 0.2 \end{aligned}$$

o qual podemos escrever matricialmente $Ax = b$ onde

$$A = \begin{bmatrix} 1.2 & -0.4 & 0.1 \\ 0.1 & 0.7 & 0.21 \\ 0.2 & -0.1 & 1.76 \end{bmatrix}$$

$$b = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \end{bmatrix}$$

que é diagonal dominante conforme já vimos anteriormente, isto é, podemos resolver esse sistema via Jacobi e também via Gauss-Seidel.

Fazendo com $x_0 = (0, 0, 0)^T$ e tolerância igual a 10^{-4} e 20 iterações, chamamos o programa `gseidel.m` fazendo

```
>> gseidel(A,b,X0,0.0001,20)
```

onde teremos como resposta

```
Z =
0         0         0
0.3333    0.2381    0.0893
0.4053    0.2010    0.0790
0.3938    0.2058    0.0806
0.3952    0.2051    0.0804
0.3950    0.2052    0.0804
0.3950    0.2052    0.0804
ans =
0.3950
0.2052
0.0804
```

Ou seja, foram necessárias 6 iterações pelo método de Gauss-Seidel para aproximar com a tolerância desejada.

Assim, a solução aproximada é dada por

$$x_6 = \begin{bmatrix} 0.3950 \\ 0.2052 \\ 0.0804 \end{bmatrix}$$

4.3 Normas Matriciais e Condicionamento de Sistemas

Sistemas matriciais são de grande utilidade na resolução de problemas de engenharia, porém cada caso é deduzido conforme sua aplicação e muitas vezes esses sistemas podem não ser bem condicionados. Estudaremos nessa seção, como identificar e proceder com sistemas ditos mal condicionados. Iniciamos nossa abordagem apresentando um sistema linear simples de duas variáveis, dado por:

$$\begin{aligned} x + y &= 2 \\ 1001x + 1000y &= 2001 \end{aligned}$$

cuja solução é $x = 1$ e $y = 1$. Este sistema pode ser escrito matricialmente como $Ax = b$ onde

$$A = \begin{bmatrix} 1 & 1 \\ 1001 & 1000 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 2001 \end{bmatrix}$$

No caso desse sistema podemos identificar que a solução é dada por duas retas que se intersecionam no ponto (1,1) no sistemas de coordenadas cartesianas do plano xy. Se procedermos uma pequena alteração na matriz A e tornarmos a mesma

$$A = \begin{bmatrix} 1.01 & 1 \\ 1001 & 1000 \end{bmatrix}$$

A solução do sistema passa a ser $x = -0.11$ e $y = 2.02$ o que configura uma mudança brusca em termos percentuais. Mas porque quando alterção em 1% apenas uma coordenada do sistema suas respostas alteram em torno de 90%? Trata-se claramente do um sistema mal condicionado, onde pequenas alterações em A ou em b podemos ocasionar mudanças de grande porte na solução.

Para quantificarmos isso, devemos entender previamente as propriedades das normas matriciais. Dizemos que a norma de uma matriz A é dada por $\| A \|$ e usaremos três tipos delas para cálculo, são elas

- $\| A \|_2 = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2 \right)^{1/2}$
- $\| A \|_1 = \max_i \{ \sum_{j=1}^{n} |a_{ij}| \}$
- $\| A \|_\infty = \max_j \{ \sum_{i=1}^{n} |a_{ij}| \}$

a norma 2 é mais conhecida e também chamada de norma euclidiana, pois também é a mesma definição de módulo de um vetor. Já a norma 1, por sua

definição, é simplesmente a coluna de maior soma em módulo e a norma infinito é a linha de maior soma em módulo. Tomamos como exemplo a matriz A definida abaixo:

$$A = \begin{bmatrix} 2 & 1 & 4 \\ -1 & 1 & 5 \\ 3 & 4 & -2 \end{bmatrix}$$

Nesse caso, temos que

- $\| A \|_2 = \left(2^2 + 1^2 + 4^2 + (-1)^2 + 1^2 + 5^2 + 2^2 + 4^2 + (-2)^2\right)^{1/2} = 6,7298$
- $\| A \|_1 = |4| + |5| + |-2| = 11$
- $\| A \|_\infty = |3| + |4| + |-2| = 9$

e podemos sempre usar o Matlab para o cálculo de normas matriciais, com os comandos:

```
>> norm(A)
>> norm(A,1)
>> norm(A,'inf')
```

respectivamente para norma 2,1 e infinito.

Algumas das propriedades importantes das normas são dadas por: (válidas para qualquer uma das três normas)

- $\| \alpha A \| = |\alpha| \, \| A \|$
- $\| AB \| \leq \| A \| \| B \|$
- $\| A + B \| \leq \| A \| + \| B \|$

Assim, podemos voltar ao exemplo inicial do sistema $Ax = b$, tomaremos inicialmente δ_x como a alteração ocasionada no vetor solução e δ_b um pequena alteração ocasionada no vetor b (a análise é análoga para a matriz A). Teremos que

$$A(x + \delta_x) = b + \delta_b \tag{4.10}$$

e consequentemente por $Ax = b$, teremos que $A\delta_x = \delta_b$. Gostariamos de estudar a influência de pequenas alterações sobre a solução. Vamos então olhar a para o erro relativo da solução que é dado por $\frac{\|\delta_x\|}{\|x\|}$ e podemos escrever inicialmente que

$$\| \delta x \| \leq \| A^{-1} \| \| \delta_b \| \tag{4.11}$$

pois $\delta_x = A^{-1}\delta_b$. Também por outro lado, temos que $Ax = b$ implicando que $\| A \| \| x \| \geq \| b \|$ e portanto $\| x \| \geq \frac{\|b\|}{\|A\|}$. Dividindo a última equação por $\| x \|$ teremos que

$$\frac{\| \delta x \|}{\| x \|} \leq \frac{\| A^{-1} \|\| \delta_b \|}{\| x \|} \tag{4.12}$$

e usando que $\| x \| \geq \frac{\|b\|}{\|A\|}$, teremos que

$$\frac{\| \delta x \|}{\| x \|} \leq \frac{\| A^{-1} \|\| A \|\| \delta_b \|}{\| b \|} \tag{4.13}$$

Assim,

$$\underbrace{\frac{\| \delta x \|}{\| x \|}}_{ER(x)} \leq \underbrace{\| A^{-1} \|\| A \|}_{\kappa(A)} \underbrace{\frac{\| \delta_b \|}{\| b \|}}_{ER(b)} \tag{4.14}$$

logo, o número de condicionamento da matriz A é dado por

$$\kappa(A) = \| A \|\| A^1 \| \tag{4.15}$$

De maneira que podemos concluir que o erro relativo no vetor solução x pode variar até $\kappa(A) \times ER(b)$ (número de condicionamento de A vezes o erro relativo do vetor b). Em outras palavras, uma pequena perturbação no vetor b pode acarretar uma pertubação muito maior na solução se o número de condicionamento da matriz A for alto.

Exemplo:

Considere o sistema $Ax = b$ dado por

$$A = \begin{bmatrix} 2 & 1 & 4 \\ -1 & 1 & 5 \\ 3 & 4 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

cuja solução é $x = (0.38, 0.54, 0.16)^T$

porém ao perturbarmos b em 2% na primeira coordenada, teremos que

$$b = \begin{bmatrix} 2.04 \\ 1 \\ 3 \end{bmatrix}$$

teremos como solução $x = (0.40, 0.53, 0.17)^T$. De fato, parece tratar-se de um sistema bem condicionado. Verificando esse fato temos que ao cadastrar no Matlab a matriz A e usar o comando

```
>> cond(A)
```

e temos como resposta

```
>> ans = 4.1385
```

Como $\delta_b = (0.04, 0, 0)^T$ e consequentemente $\| \delta_b \|_2 = 0.04$ e $\| b \|_2 = 3.7417$, teremos que $\frac{\|\delta_b\|}{\|b\|} = \frac{0.04}{3.7417} = 0.0107$, ou seja, teremos erro em x de no máximo $\kappa(A).\frac{\|\delta_b\|}{\|b\|} = 4.1385 \times 0.0107 = 0.0442$, ou seja, de 4,42%.

4.4 Exercícios

Método de Jacobi e Gauss-Seidel:

1. Em cada caso, verifique se as matrizes são diagonais dominantes:

   (a)
   $$\begin{bmatrix} 2 & 2 & 6 \\ 1 & 4 & 2 \\ 9 & -1 & 4 \end{bmatrix}$$

   (b)
   $$\begin{bmatrix} 3 & 1 & 0 & 1 \\ -1 & 6 & 2 & 2 \\ 2 & 1 & 19 & -2 \\ 3/2 & -1 & 1 & 4 \end{bmatrix}$$

   (c)
   $$\begin{aligned} 0.723x - 0.263y + 0.089z &= 2.4 \\ 1.721x - 2.263y + 0.491z &= 2.1 \\ 0.632x - 0.263y + 9.019z &= -7.4 \end{aligned}$$

2. Considere o sistema

   $$\begin{aligned} 0.723x - 0.263y + 0.089z &= 2.4 \\ 1.721x - 2.263y + 0.491z &= 2.1 \\ 0.632x - 0.263y + 9.019z &= -7.4 \end{aligned}$$

   Resolva no Matlab usando o comando `>>A\b` e após resolva (comparando) por Jacobi e Gauss-Seidel com tolerância 10^{-4} e diga em cada caso quantas iterações foram necessárias.

3. Diga se é possível resolver o sistema abaixo com Jacobi ou Gauss-Seidel e porque (em caso afirmativo ou não).

   $$\begin{aligned} 1.705x - 2.163y + 0.421z &= 2.101 \\ 0.832x - 0.263y + 11.019z &= -7.1 \\ 0.623x - 0.163y + 0.089z &= 1.4 \end{aligned}$$

4. Faça uma permutação de linhas no sistema abaixo e resolva pelos métodos de Jacobi e Gauss-Seidel com tolerância de 10^{-3}:

$$
\begin{aligned}
x_1 + 10x_2 + 3x_3 &= 27 \\
4x_1 + x_3 &= 6 \\
2x_1 + x_2 + 4x_3 &= 12
\end{aligned}
$$

5. Considere o seguinte sistema de equações lineares:

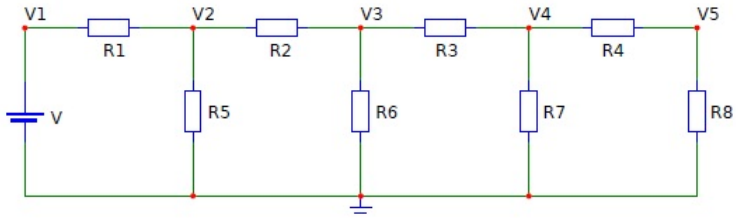
$$
\begin{aligned}
x_1 - x_2 &= 0 \\
-x_{j-1} + 5x_j - x_{j+1} &= \cos\left(\frac{j}{10}\right), j = 2, ..., 10 \\
x_{11} &= x_{10}/2
\end{aligned}
$$

   Construa a iteração para encontrar a solução deste problema pelos métodos de Gauss-Seidel e Jacobi com tolerância igual a 10^{-5}.

6. O circuito linear da figura pode ser modelado pelo sistema dado a seguir. Escreva esse sistema na forma matricial sendo as tensões V_1, V_2, V_2, V_4 e V_5 as cinco incógnitas. Resolva esse problema quando $V = 127$ e

   (a) $R1 = R2 = R3 = R4 = 2$ e $R5 = R6 = R7 = 100$ e $R8 = 50$.

   (b) $R1 = R2 = R3 = R4 = 2$ e $R5 = 50$ e $R6 = R7 = R8 = 100$.



$$
\begin{aligned}
V_1 &= V \\
\frac{V_1 - V_2}{R_1} + \frac{V_3 - V_2}{R_2} - \frac{V_2}{R_5} &= 0 \\
\frac{V_2 - V_3}{R_2} + \frac{V_4 - V_3}{R_3} - \frac{V_3}{R_6} &= 0 \\
\frac{V_3 - V_4}{R_3} + \frac{V_5 - V_4}{R_4} - \frac{V_4}{R_7} &= 0 \\
\frac{V_4 - V_5}{R_4} - \frac{V_5}{R_8} &= 0
\end{aligned}
$$

   Complete a tabela abaixo representado a solução com DIGSE 4:

Caso	V_1	V_2	V_3	V_4	V_5
(a)					
(b)					

Condicionamento de Sistemas:

1. Em cada caso, calcule o número de condicionamento relativo ao sistema $Ax = b$ e explique o que o mesmo significa. Logo após, usando o Matlab, resolva o sistema A e b dados e posteriormente resolva com pequenas variações em A e b e compare.

   (a)
   $$A = \begin{bmatrix} 2 & 2.2 & 6 \\ 1 & 4 & 2 \\ 7.1 & 0 & 401 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ 1 \\ 4.2 \end{bmatrix}$$

   (b)
   $$A = \begin{bmatrix} 2.01 & 2.23 & -1.89 \\ 1.23 & 8.91 & 0 \\ 3 & 1 & 2 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ -1 \\ -0.45 \end{bmatrix}$$

   (c)
   $$A = \begin{bmatrix} 200 & 21 & 19 \\ -0.01 & 23 & 0.01 \\ 30 & 1 & 159 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 20 \\ -1 \end{bmatrix}$$

2. Calcule o número de condicionamento do sitema para os valores de λ pedidos nas três normas possíveis:

   $$\begin{aligned} 71x + 41y &= 10 \\ \lambda x + 30y &= 4 \end{aligned}$$

   (a) Quando $\lambda = 51$.

   (b) Quando $\lambda = 52$.

   (c) Quando $\lambda = 51,5$.

3. Usando a norma 1, calcule o número de condicionamento da matriz

   $$A = \begin{bmatrix} 1 & 2 \\ 2+\varepsilon & 4 \end{bmatrix}$$

   em função de ε e considere $0 < \varepsilon < 1$, faça o limite quando $\varepsilon \to 0$ e interprete o resultado.

4. Considere os sitemas

$$\begin{aligned} 100000x - 9999.99y &= -10 \\ -9999.99x + 1000.1y &= 1 \end{aligned}$$

$$\begin{aligned} 100000x - 9999.99y &= -9.999 \\ -9999.99x + 1000.1y &= 1.01 \end{aligned}$$

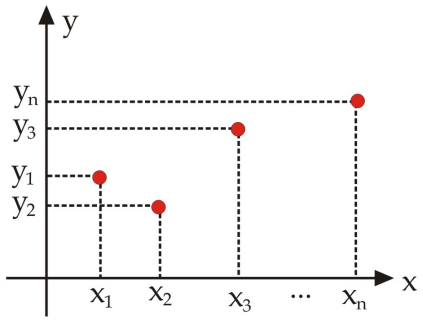
Encontre a solução de cada um e discuta usando o conceito do número de condicionamento.

5 Interpolação

Muitas vezes de posse de um conjuntos de dados ou pontos em um plano, nos questionamos se é possível modelar uma curva que interprete o comportamento dessa distribuição de pontos e que passa por eles. Na interpolação estudamos como modelar tais curvas de modo que essas condições são satisfeitas. De maneira mais geral teremos um conjunto de pontos

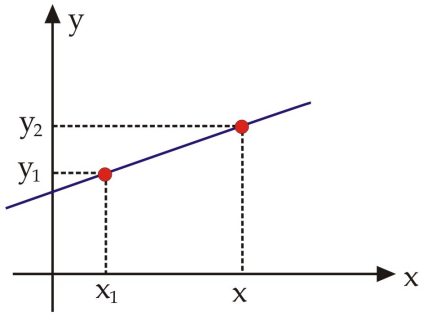
$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

que pode ser distribuido conforme suas posições no plano xy da forma:

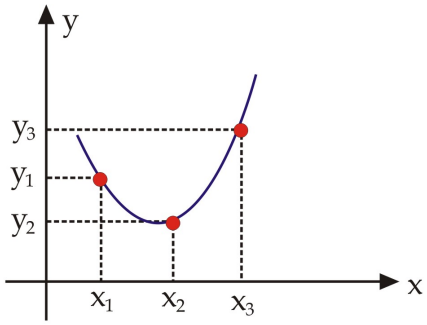


Vamos buscar uma função $y = f(x)$ tal que $f(x_i) = y_i \forall i \in \{1, 2, ..., n\}$. Chamamos tal função f de função interpoladora dos dados.

Iniciamos a discutindo qual o polinômio de menor grau possível que interpola um certo número de pontos. Por exemplo, no caso de termos apenas dois pontos, vamos interpolar por uma reta (polinômio de grau 1), como mostra o desenho.



No caso de três pontos, teremos que interpolar por um polinômio de grau 2, pois existe a chance deles não serem colineares e portanto um polinômio de grau 1 não bastaria para modelar essa função. Assim, teriamos



De maneira geral, temos que n pontos serão interpolados por um polinômio de grau $n-1$. A interpolação polinomial é um caso particular do problema geral de interpolação, no qual a família de funções é constituída de polinômios. O teorema da Aproximação de Weierstrass, segundo o qual, qualquer função contínua definida em um intervalo fechado pode ser aproximada uniformemente por um polinômio tão bem quanto se queira.

Teorema: (Weierstrass) *Seja f uma função contínua definida no intervalo fechado $[a,b]$ e seja δ um número positivo. Então existe um polinômio p, tal que para todo $x \in [a,b]$, $|f(x) - p(x)| < \delta$.*

O teorema de Weirestrass afirma que existe um polinômio tão próximo quanto de queira da função a ser interpolada. Para determinar esse polinômio precisamos de algumas técnicas, uma delas é a resolução de um sistema cujo sua montagem é determinada por $y_i = f(x_i)$ para todo $i \in \{1,2,...,n\}$ e chamamos entre outros nomes, essa técnica, interpolação via matriz de Vandermonde.

5.1 Matriz de Vandermonde

Vamos considerar um exemplo cujo a intenção é interpolar um polinômio que passa por 3 pontos. Sejam os pontos

$$(1,1),(2,0.5),(3,2)$$

Assim teremos que interpolar por um polinômio de grau 2 da forma $p_2(x) = ax^2 + bx + c$ que passe por esses pontos, isto é, tal que

$$\begin{aligned} a.1^2 + b.1 + c &= 1 \\ a.2^2 + b.2 + c &= 0.5 \\ a.3^2 + b.3 + c &= 2 \end{aligned}$$

Tornando assim um sistema linear da forma

$$\begin{bmatrix} 1 & 1 & 1 \\ 2^2 & 2 & 1 \\ 3^3 & 3 & 1 \end{bmatrix}\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ 2 \end{bmatrix}$$

Cuja solução é exatamente $a = 1, b = -3.5, c = 3.5$ tornando o polinômio interpolar

$$p_2(x) = x^2 - 3.5x + 3.5 \tag{5.1}$$

Essa solução para casos maiores pode ser descoberta usando o Matlab com o programa

Código em Matlab para Interpolação via Matriz de Vandermonde:

```
---------------------------------
function s = vandermonde(x,y)
yt = y';
A = vander(x)
s=A\yt
end
---------------------------------
```

A matriz desse sistema é chamada de Matriz de Vandermonde e quando desejamos interpolar um polinômio de grau $n-1$ tem em geral a forma

$$A_v = \begin{bmatrix} x_1^n & x_1^{n-1} & \cdots & 1 \\ x_2^n & x_2^{n-1} & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1}^n & x_{n-1}^{n-1} & \cdots & 1 \end{bmatrix} \tag{5.2}$$

ainda no mesmo exemplo, proceder da seguinte maneira. Primeiramente cadastramos os pontos como vetores, um com as coordenadas de x e outro com as coordenadas de y. No Matlab:

```
>> x=[1 2 3];
>> y=[1 0.5 2 ];
```

Logo após chamamos o programa

```
>> vandermonde(x,y)
```

teremo como resposta

```
ans =
    1.0000
   -3.5000
    3.5000
```

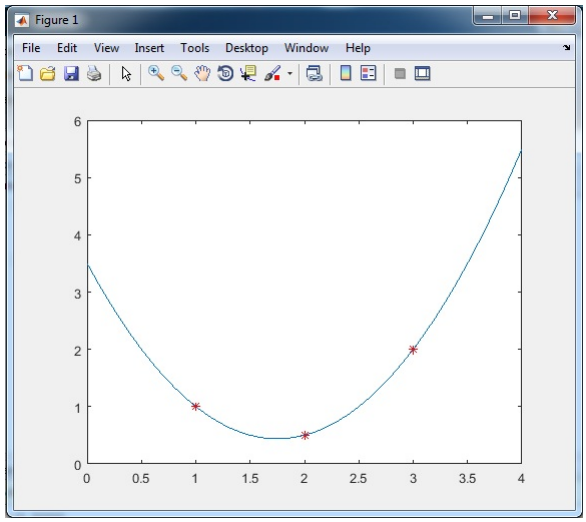
Para verificação podemos plotar o gráfico juntamente com o conjunto de pontos, fazendo antes a definição da função dentro do domínio (nesse caso o domínio é de 1 a 3, faremos uma pouco maior, de 0 a 4) então temos

```
>> x1=(0:0.1:4);
>> y1=x1.^2-3.5*x1+3.5;
```

e posteriormente

```
>> plot(x,y,'*r',x1,y1)
```

Assim teremos que



5.2 Interpolação via Lagrange

Outro tipo de interpolação é a utilização dos polinômios de Lagrange. Dado um conjunto de pontos distintos dois a dois, definimos os polinômios de Lagrange como os polinômios de grau n-1 que satisfazem

$$\ell_k(x_j) = \begin{cases} 1, k = j \\ 0, k \neq j \end{cases}$$

Assim, o polinômio $p(x)$ de grau $n-1$ que interpola os pontos dados tal que $p(x_j) = y_j$, $j = 1,...,n$ é dado por

$$p(x) = y_1\ell_1(x) + y_2\ell_2(x) + ... + y_n\ell_n(x) \tag{5.3}$$

Estes polinômios de Lagrange são da forma:

$$\ell_i(x) = \prod_{j=1, j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)} \tag{5.4}$$

Como exemplo, podemos tomar o problema de encontrar o polinômio da forma $p_3(x) = a_3x^3+a_2x^2+a_1x+a_0$ que passa pelos pontos $\{(0,0),(1,1),(2,4),(3,9)\}$.

Escrevemos então no Matlab que

```
>> x=[0 1 2 3];
>> y=[0 1 4 9];
```

e utilizaremos o código em Matlab que monta esses polinômios de Lagrange e devolve os mesmos já na sua forma interpoladora (como na equação 5.3).

Código em Matlab para Interpolação via Lagrange:

```
----------------------------------
function C=lagrange(x,y)
n1=length(x);
n=n1-1;
L=zeros(n1,n1);
for k=1:n+1;
v=1;
for j=1:n+1;
if k~=j
v=conv(v,poly(x(j)))/(x(k)-x(j));
end
end
L(k,:)=v;
end
C=y*L
----------------------------------
```

Assim, ao chamarmos o código,

```
>> lagrange(x,y)
```

teremos como resposta

```
ans =
```

```
-0.0483    1.0950    0.1533         0
```

que são os coeficientes de polinômio de grau 3 que interpola os pontos dados, isto é, teremos que

$$p_3(x) = -0.0483x^3 + 1.0950x^2 + 0.1533x \tag{5.5}$$

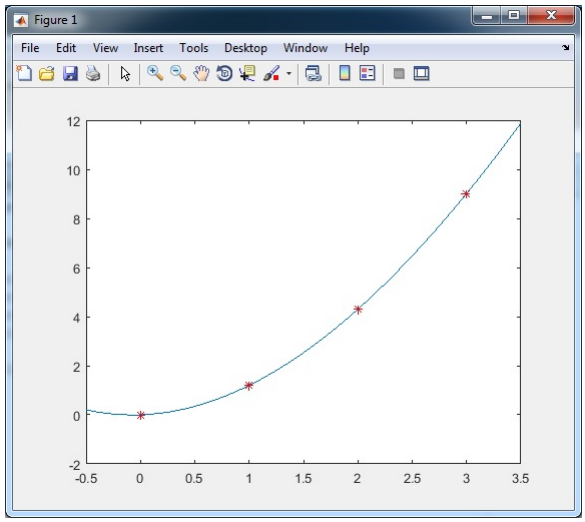
podemos além disso analisar o comportamento dessa função juntamente com o conjunto de pontos. Nesse caso fariamos que

```
>> x1=(-0.5:0.1:3.5);
>> y1=-0.0483*x1.^3+1.0950* x1.^2 +0.1533*x1;
```

aqui fica evidente que escolhemos um domínio um pouco maior que o dado (de -0.5 a 3.5 ao invés de 0 a 3) para analisar os pontos dados no plano de maneira mais clara. Posteriormente, fazemos

```
>> plot(x,y,'*r',x1,y1)
```

Assim teremos que



5.3 Exercícios

Interpolação por Vandermonde:

1. Encontre o polinômio interpolador utilizando a matriz de Vandermonde para o conjunto de pontos $\{(-2,-47),(0,-3),(1,4),(2,41)\}$. Então, faça um gráfico com os pontos e o polinômio interpolador encontrado.

2. Encontre o polinômio de grau 4 que interpola os pontos $\{(-1,-4),(0,-3.1),(1,1.4),(2,4.2),(3,3.1)\}$. Então, faça um gráfico com os pontos e o polinômio interpolador encontrado. Aproxime o valor desse polinômio em $x = 2.3$ e $x = 3.4$ e verifica os pontos no gráfico obtido.

3. Encontre o polinômio interpolador utilizando a matriz de Vandermonde para o conjunto de pontos $\{(-2,-1),(0,-3),(2,5)\}$. Então, faça um gráfico com os pontos e o polinômio interpolador encontrado.

4. Um professor corrigiu a nota da uma prova de peso 10,00 de seis de seus alunos e comparou com o tempo que cada um levou para realizar a prova em minutos. Obtendo o seguinte resultado:

	A_1	A_2	A_3	A_4	A_5	A_6
tempo(min)	32	49	65	105	132	163
Nota	3.2	4.6	7.3	8.1	7.0	6.2

   Monte um polinômio interpolador que modelo as notas pelo tempo gasto e diga o que se espera de um aluno que leve 124 minutos para fazer a prova.

5. Explique a razão de cada item baseado na teoria:

   (a) Existem infinitas parábolas que interpolam dois pontos dados $\{(x_1,y_1),(x_2,y_2)\}$, com $x1 \neq x2$.

   (b) Não existe reta que interpola os pontos (1, 1), (2, 2,1), (3, 3).

   (c) Não existe parábola de equação $y = a_0 + a_1x + a_2x^2$ que interpola dois pontos dados $\{(x_1,y_1),(x_1,y_2)\}$, com $y_1 \neq y_2$. Mas, existem infinitas parábolas de equação $x = a_0 + a_1y + a_2y^2$ que interpolam estes pontos.

   (d) Um modelo de interpolação que foi montado para valores de x entre 0 e 10 foi usado para aproximar um ponto em que a coordenada x era 11.3. O resultado foi muito maior do que o esperado.

Interpolação por Lagrange:

1. Encontre o polinômio interpolador de Lagrange para o conjunto de pontos $\{(-2,-1),(0,-1),(1,5),(2,7)\}$. Então, faça um gráfico com os pontos e o polinômio interpolador encontrado.

2. Encontre o polinômio de grau 4 que interpola os pontos $\{(-1,-2),(0,-1),(1,1),(2,0.4),(3,6)\}$. Então, faça um gráfico com os pontos e o polinômio interpolador encontrado. Aproxime o valor desse polinômio em $x = 2.1$ e $x = 3.9$ e verifica os pontos no gráfico obtido.

3. Encontre o polinômio interpolador de Lagrange para o conjunto de pontos $\{(-2,-1),(0,-3),(2,5)\}$. Compare o resultado com o exercício 3 de Vandermonde e faça um gráfico com os pontos e o polinômio interpolador encontrado.

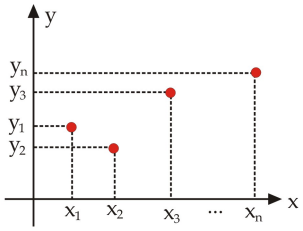
4. Um experimento realizado obteve dados referentes a evolução de uma taxa de juros ao longo de seis meses. Obtendo assim o seguinte padrão:

	M_1	M_2	M_3	M_4	M_5	M_6
tempo(meses)	1	2	3	4	5	6
taxa juros (%)	1.22	x	1.24	1.29	1.19	1.25

   Monte um polinômio interpolador via Lagrange que modele esse problema e aproxime a taxa de juros no segundo mês.

6 Ajuste de Curvas

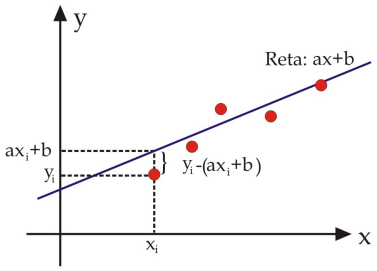
Vamos aqui neste capítulo, discutir como podemos ajustar uma determinada curva a um certo conjuntos de dados. Abordaremos principalmente o método dos mínimos quadrados. Em outras palavras estamos nos perguntando como é possível uma vez dado um conjunto de N pontos da forma $\{(x_i, y_i)\}$ com $i \in \{1, 2, ..., N\}$ encontrar uma função $f(x)$ que melhor se ajuste a esses pontos. Cabe aqui salientar que, diferentemente da interpolação, a função $f(x)$ que estamos buscando não precisa necessariamente passar por todos estes pontos do conjunto dado.



6.1 O Métodos dos Mínimos Quadrados

O método dos mínimos quadrados busca ajustar uma função que tem forma similar ao conjuntos de pontos dados. Por exemplo, no desenho anterior vimos que uma RETA melhor se aproxima desses pontos. Esse método busca medir a distância de cada ponto a função em questão. Encontrando a função cuja a soma dos quadrados das distâncias é mínima, o que da nome ao método supracitado. Vamos realizar a dedução do método supondo que a função a qual desejamos ajustar é uma reta, mas a dedução para outras funções é análoga.

Consideramos então a figura



Desejamos que as somas dos quadrados das distâncias seja mínima, então faremos

$$\frac{\partial}{\partial b}\left(\sum_{i=1}^{N}(y_i-(ax_i+b))^2\right)=0 \tag{6.1}$$

$$\frac{\partial}{\partial a}\left(\sum_{i=1}^{N}(y_i-(ax_i+b))^2\right)=0 \tag{6.2}$$

o que nos leva a

$$\left(\sum_{i=1}^{N}2(y_i-(ax_i+b))\right)=0 \tag{6.3}$$

$$\left(\sum_{i=1}^{N}2(y_i-(ax_i+b))(-x_i)\right)=0 \tag{6.4}$$

e ainda que

$$\sum_{i=1}^{N}(y_i-(ax_i+b))=0 \tag{6.5}$$

$$\sum_{i=1}^{N}(-x_iy_i+ax_i^2+bx_i)=0 \tag{6.6}$$

e portanto

$$\sum_{i=1}^{N}y_i-a\sum_{i=1}^{N}x_i-\sum_{i=1}^{N}b=0 \tag{6.7}$$

$$-\sum_{i=1}^{N}x_iy_i+a\sum_{i=1}^{N}x_i^2+b\sum_{i=1}^{N}x_i=0 \tag{6.8}$$

o que nos leva ao sistema

$$\begin{bmatrix} N & \sum_{i=1}^{N}x_i \\ \sum_{i=1}^{N}x_i & \sum_{i=1}^{N}x_i^2 \end{bmatrix}\begin{bmatrix} b \\ a \end{bmatrix}=\begin{bmatrix} \sum_{i=1}^{N}y_i \\ \sum_{i=1}^{N}y_ix_i \end{bmatrix} \tag{6.9}$$

Assim, podemos ver que a solução é dada atráves desse sistema que descobre os valores a e b, isto é, a reta $y=ax+b$.

Exemplo:

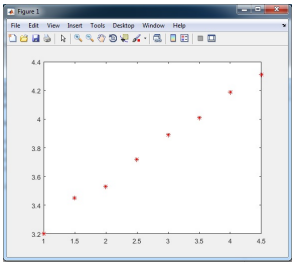
Digamos que queremos verificar se podemos ajustar um certo conjuntos de pontos por uma reta. O conjunto de pontos é dado por:

$$\{(1,3.2),(1.5,3.45),(2,3.53),(2.5,3.72),(3,3.89),(3.5,4.01),(4,4.19),(4.5,4.31)\}$$

podemos verificar previamente no Matlab qual o comportamento desses conjunto de pontos. Para isso, cadastramos 2 vetores horizontais que das coordenadas x e y respectivamente. Isto é,

```
>> x=[1 1.5 2 2.5 3 3.5 4 4.5]
>> y=[3.2 3.45 3.53 3.72 3.89 4.01 4.19 4.31]
>>plot(x,y,'*r')
```

obtendo:



Aqui vemos claramente que uma curva possível para ajuste, seria uma reta. Vamos então buscar via mínimos quadrados essa reta. Uma vez já cadastrados os vetores com as respectivas coordenadas de x e y vamos chamar o código que realiza o cálculo de a e b via mínimos quadrados.

Código em Matlab para Ajuste via Mínimos Quadrados:

```
---------------------------------
function a=minqreta(x,y)
n1=length(x);
m1=0;m2=0;b1=0;b2=0;
for i=1:n1;
m1=m1+x(i)^2;
m2=m2+x(i);
b1=b1+x(i)*y(i);
b2=b2+y(i);
end
M=[n1,m2;m2,m1];
b=[b2;b1];
a=M\b
---------------------------------
```

Chamando então o código, para o nosso exemplo, teremos que

```
>> minqreta(x,y)
```

nos devolve

```
ans =
2.9311
0.3114
```

Sendo assim, a reta que melhor se ajusta a esses pontos é dada por

$$y = 0.3114x + 2.9311 \tag{6.10}$$

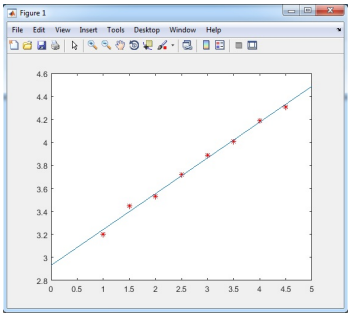
Verificando no próprio Matlab, temos que

```
>> x1=(0:0.1:5);
>> y1=0.3114*x1+2.9311;
```

aqui fica evidente que escolhemos um domínio um pouco maior que o dado (de 0 a 5 ao invés de 1 a 4.5) para analisar os pontos dados no plano de maneira mais clara. Posteriormente, fazemos

```
>> plot(x,y,'*r',x1,y1)
```

Assim teremos que



6.1.1 Ajuste por outras Curvas

Se desejarmos aproximar problemas por uma parábola também é possível e a dedução é análoga. No caso de ajuste a uma polinômio de segundo grau, teremos que o sistema a ser resolvido é

$$\begin{bmatrix} N & \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i^3 \\ \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i^3 & \sum_{i=1}^{N} x_i^4 \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} y_i x_i \\ \sum_{i=1}^{N} y_i x_i^2 \end{bmatrix} \tag{6.11}$$

cujo código para resolução em Matlab é dado por

Código em Matlab para Ajuste via Mínimos Quadrados (Parábola):

```
---------------------------------
function a=minqparabola(x,y)
n1=length(x);
m1=0;m2=0;m3=0;m4=0;b1=0;b2=0;b3=0;
 for i=1:n1;
m2=m2+x(i)^2;
m1=m1+x(i);
m3=m3+x(i)^3;
m4=m4+x(i)^4;
b1=b1+y(i);
b2=b2+x(i)*y(i);
b3=b3+y(i)*x(i)^2;
end
M=[n1,m1,m2;m1,m2,m3;m2,m3,m4];
```

```
b=[b1;b2;b3];
a=M\b
end
--------------------------------
```

Se desejarmos aproximar por uma hipérbole do tipo

$$y = \frac{1}{ax+b} \tag{6.12}$$

Basta que façamos a mudança de variável do tipo $Y = \frac{1}{y}$ tornando a equação anterior em

$$Y = ax + b \tag{6.13}$$

assim, usaremos o mesmo sistema para obtenção da reta de ajuste, tornando-o da forma

$$\begin{bmatrix} N & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \frac{1}{y_i} \\ \sum_{i=1}^{N} \frac{1}{y_i} x_i \end{bmatrix} \tag{6.14}$$

Se desejarmos aproximar por uma curva exponencial do tipo

$$y = ke^{ax} \tag{6.15}$$

fazendo

$$\begin{aligned} \ln(y) &= \ln ke^{ax} \\ \ln(y) &= \ln(k) + \ln(e^{ax}) \\ \underbrace{\ln(y)}_{Y} &= \underbrace{\ln(k)}_{K} + ax \end{aligned}$$

Logo

$$Y = K + ax \tag{6.16}$$

Logo o sistema fica

$$\begin{bmatrix} N & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{bmatrix} \begin{bmatrix} K \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \ln y_i \\ \sum_{i=1}^{N} \ln y_i x_i \end{bmatrix} \tag{6.17}$$

com $k = e^K$.

6.2 Exercícios

Ajuste de curvas:

1. Sejam dados o conjunto de pontos $\{(0.23,-0.54),(-0.30,-0.54),(0.04,-0.57)\}$. Encontre a função $f(x) = ax+b$ que melhor se ajusta no sentido de mínimos quadrados aos pontos dados. Faça, então, um gráfico com os pontos e o esboço da função ajustada.

2. Seja dado o conjunto de pontos $\{(-1,94,1,02),(-1,44,0,59),(0,93,-0,28),(1,39,-1,04)\}$. Encontre a função $f(x) = ax+b$ que melhor se ajusta no sentido de mínimos quadrados aos pontos dados. Então, responda cada item:

   (a) Encontre o valor de $f(1)$.

   (b) Encontre o valor de $f(0.97)$.

   (c) Encontre o valor de $f(2)$ e diga se é um valor esperado para esse caso.

3. Considere o conjunto de dados $\{(1,0.55),(2,0.14),(3,-0.27),(4,0.11),(5,0.23),(6,0.67)\}$ verique graficamente os pontos e ajuste por uma curva que seja mais aproximada na sua opinião. Logo após faça o gráfico da curva ajustada com os dados fornecidos.

4. Um experimento realizado em Engenharia Mecânica mediu em um sistema massa mola num dado intervalo de tempo para um bloco sujeito a uma força F e de massa fixa m. A variação da força exercida nesse bloco para diferentes valores da constante k foi medida nesse experimento. Obtendo assim os seguintes resultados em forma de tabela:

k	(t_1,F_1)	(t_2,F_2)	(t_3,F_3)	(t_4,F_4)	(t_5,F_5)	(t_6,F_6)
0.34	(1, 0.23)	(2, 0.31)	(3, 0.34)	(4, 0.56)	(5, 0.83)	(6, 0.91)
0.82	(1, 0.33)	(2, 0.58)	(3, 0.98)	(4, 1.51)	(5, 3.45)	(6, 15.37)
0.91	(1, 0.91)	(2, 0.42)	(3, 0.08)	(4, 0.36)	(5, 0.73)	(6, 1.13)

   onde os valores de t representam o tempo em segundos e os valores de F representam a força obtida em N. Então

   (a) Para o valor de $k = 0.34$ ajuste a uma curva que mais se aproxima aos dados fornecidos e verifique o resultado graficamente.

   (b) Para o valore de $k = 0.82$ ajuste a uma curva que mais se aproxima aos dados fornecidos e verifique o resultado graficamente.

   (c) Para o valore de $k = 0.91$ ajuste a uma curva que mais se aproxima aos dados fornecidos e verifique o resultado graficamente.

7 Integração Numérica

Muitas vezes uma integração que modela um problema real não pode ser resolvida analiticamente. Nestes casos devemos usar técnicas numéricas para aproximar a integral em questão. Existem diversos métodos de integração numérica que aproximam a integral

$$\int_a^b f(x)dx \tag{7.1}$$

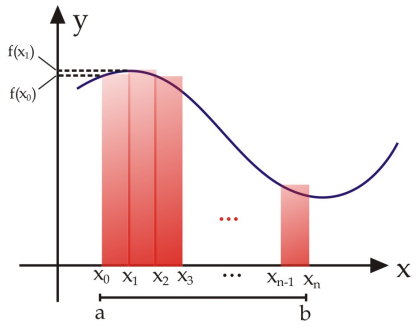
Abordando inicialmente o problema pelo seu conceito mais geral, o de área, pois a equação acima nada mais é do que a área abaixo da curva $f(x)$ até o eixo x entre os limites a e b. Faremos uma primeira abordagem através de uma subdivisão em áreas menores, de maneira a dividir igualmente em n intervalos o intervalo inicial $[a,b]$ a partir de um conjunto ordenado de pontos $a = x_0 < x_1 < ... < x_n = b$. Em cada intervalo i, o subintervalo será aproximado por $f(x_i)\Delta x_i$ (com $\Delta x_i = |x_i - x_{i-1}|$) e a integral será aproximada por

$$I \approx \sum_{i=0}^{n-1} f(x_i)\Delta x_i \tag{7.2}$$

O tamanho de cada intervalo será igual e chamaremos de h que pode ser descrito por $h = \frac{b-a}{n}$

7.1 Integração por Riemann

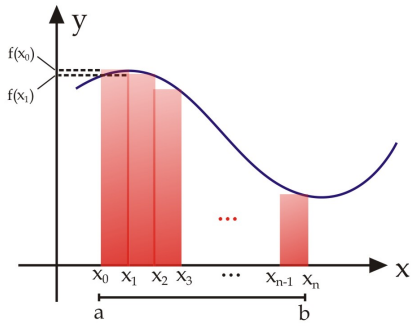
A aproximação indicada é chamada de aproximação de Riemann pela esquerda, pois usa como altura dos retângulos o valor da função à esquerda do intervalo como mostra a figura



ou podemos também indicar essas mesmas alturas tomando os valores da função à direita

$$I \approx \sum_{i=1}^{n} f(x_i)\Delta x_i \tag{7.3}$$

ficando dessa maneira com uma aproximação conforme o esquema da figura abaixo, chamada aproximação de Riemann à direita



Ambos casos podemos ser escritos no Matlab conforme os códigos abaixo:
Código em Matlab para Integração Numérica de Riemann à Esquerda:

```
----------------------------------
function l=riemannesq(def,a,b,n)
f=inline(def);
x=linspace(a,b,n);
y=f(x);
h=(b-a)/(n-1);
l=sum(y(1:(n-1)))*h;
end
----------------------------------
```

Código em Matlab para Integração Numérica de Riemann à Direita:

```
----------------------------------
function r=riemanndir(def,a,b,n)
f=inline(def)
x=linspace(a,b,n);
y=f(x);
h=(b-a)/(n-1);
r=sum(y(2:(n)))*h;
end
----------------------------------
```

Nos códigos descritos acima, n é a entrada do número de pontos dentro do intervalo inicial.

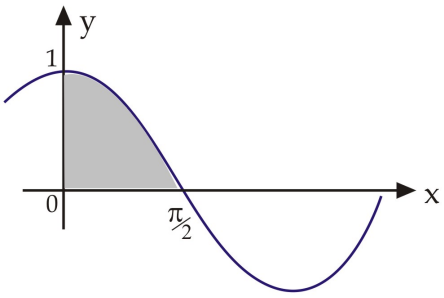
Exemplo:

Vamos integrar a função $\cos(x)$ no intervalo de 0 à $\pi/2$. Como neste caso a função cosseno é fácil de integrar analiticamente, pois

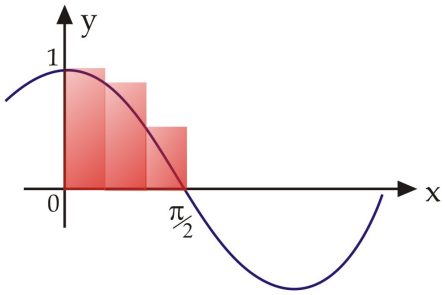
$$\int_0^{\pi/2} \cos(x)dx = \sin(x)]_0^{\pi/2} = 1$$

Assim, sabemos que as aproximações devem estar perto de 1 e quanto mais triângulos usados a aproximação da integral tende a ser mais precisa.

Se analisarmos a área a qual estamos tentando aproximar vemos que é descrita por



E se usarmos, por exemplo, Riemann a esquerda com quatro pontos (três subintervalos), estaremos aproximando conforme a figura abaixo



Ou seja, teremos que

$$h = \frac{\pi/2 - 0}{3} = \frac{\pi}{6}$$

$$\int_0^{\pi/2} \cos(x)dx \approx \frac{\pi}{6}\cos(0) + \frac{\pi}{6}\cos(\pi/6) + \frac{\pi}{6}\cos(2\pi/6) = 1,2388$$

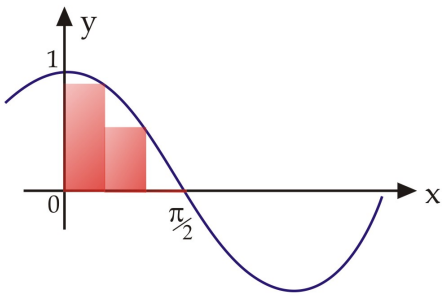
ou ainda, podemos verificar usando o código em Matlab chamado diretamente

```
>> riemannesq('cos(x)',0,pi/2,4)
```

que resulta em

```
>> ans=
1.2388
```

Ao usarmos, por exemplo, Riemann à direita com quatro pontos (três subintervalos), estaremos aproximando conforme a figura abaixo



então, teremos

$$\int_0^{\pi/2} \cos(x)dx \approx \frac{\pi}{6}\cos(\pi/6) + \frac{\pi}{6}\cos(2\pi/6) + \frac{3\pi}{6}\cos(2\pi/6) = 0,7152$$

ou ainda, podemos verificar usando o código em Matlab chamado diretamente

```
>> riemanndir('cos(x)',0,pi/2,4)
```

que resulta em

```
>> ans=
0.7152
```

Em ambos casos, a resposta encontrada como aproximação é absolutamente esperada, pois conforme as figuras, temos que Riemann à esquerda a área é maior do que a área cinza e portanto 1,2388 é natural como resposta. Já no caso de Riemann à direita temos que a área aproximada é claramente menor do que 1 e portanto 0,7152 também é esperado. O que acontece se aumentarmos o número de pontos na aproximação ? (consequentemente o número de intervalos). Em teoria a aproximação para a integral tende a ser melhor aproximada. Vejamos alguns casos de intervalos menores. Ao chamarmos o código para $n = 10$ pontos (ou 9 subintervalos), teremos como resposta

```
>> riemannesq('cos(x)',0,pi/2,10)
```

que resulta em

```
>> ans=
1.0847
```

o que configura uma resposta mais bem aproximada, mas ainda acima de 1, pois os retângulos formados na aproximação terão uma área maior. Se fizermos o mesmo, porém à direita, teremos que

```
>> riemanndir('cos(x)',0,pi/2,10)
```

que resulta em

```
>> ans=
0.9152
```

O fato de aproximar por valores abaixo também faz sentido nesse caso, pois os retângulos estarão sempre abaixo da curva. Agora, aumentando ainda mais para cada caso, veremos que a resposta tende a ser cada vez mais bem aproximada, como mostram as simulações para $n = 100, 200, 500$.

```
>> riemannesq('cos(x)',0,pi/2,100)
```

que resulta em

```
>> ans=
1.0079
>> riemannesq('cos(x)',0,pi/2,200)
```

que resulta em

```
>> ans=
1.0039
>> riemannesq('cos(x)',0,pi/2,500)
```

que resulta em

```
>> ans=
1.0016
>> riemanndir('cos(x)',0,pi/2,100)
```

que resulta em

```
>> ans=
0.9920
>> riemanndir('cos(x)',0,pi/2,200)
```

que resulta em

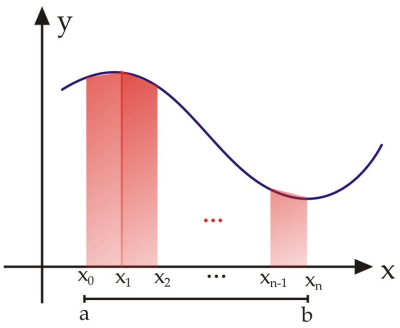
```
>> ans=
0.9960
>> riemanndir('cos(x)',0,pi/2,500)
```

que resulta em

```
>> ans=
0.9984
```

7.2 Integração por Trapézios

O método dos trapézios parte do pressuposto de que podemos usar uma figura geométrica diferente na aproximação das áreas para aproximação da integral $\int_a^b f(x)dx$. Usarmos uma configuração para o número de pontos e intervalos iguais as aproximações por Riemann, de maneira que aproximaremos conforme a figura



Assim, usando a definição da área de uma trapézio para o cálculo de cada figura. Como em um trapézio qualquer sua área é dada por

$$A_T = \frac{(B+b)h}{2}$$

onde B é a base maior, b a base menor e h a altura. Tomando atenção para o primeiro trapézio da figura, temos que sua base maior será $f(x_1)$ e sua base menor será $f(x_0)$ e a altura h coincide com o tamanho do subintervalo, tambe chamado de h. Assim, a área do primeiro trapézio, será de

$$A_1 = \frac{(f(x_0)+f(x_1))h}{2} = h\frac{f(x_0)}{2} + h\frac{f(x_1)}{2}$$

Ao realizarmos o mesmo procedimento com os demais trapézios da figura, teremos que

$$A_2 = h\frac{f(x_1)}{2}h + \frac{f(x_2)}{2}$$

$$A_3 = h\frac{f(x_2)}{2} + h\frac{f(x_2)}{2}$$

$$\vdots$$

$$A_n = h\frac{f(x_{n-1})}{2}h + \frac{f(x_n)}{2}$$

A nossa aproximação para integral será a soma das áreas de todos os trapézios.

$$
\begin{aligned}
I = \int_a^b f(x)dx &\approx h\frac{f(x_0)}{2} + h\frac{f(x_1)}{2} + h\frac{f(x_1)}{2} + h\frac{f(x_2)}{2} + \ldots + h\frac{f(x_{n-1})}{2} + h\frac{f(x_n)}{2} \\
&\approx h\left[\frac{f(x_0)}{2} + \frac{f(x_1)}{2} + \frac{f(x_1)}{2} + \frac{f(x_2)}{2} + \frac{f(x_2)}{2} + \ldots + \frac{f(x_{n-1})}{2} + \frac{f(x_{n-1})}{2} + \frac{f(x_n)}{2}\right] \\
&\approx h\left[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + \frac{f(x_n)}{2}\right] \qquad (7.4)
\end{aligned}
$$

que configura a fórmula geral para a regra dos trapézios. Abaixo segue o código em Matlab para a mesma aproximção

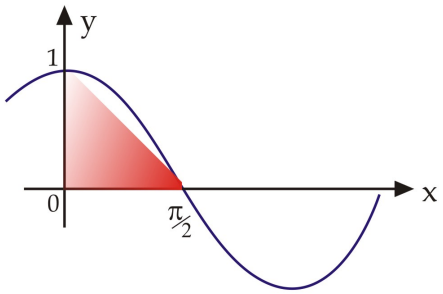
Código em Matlab para Integração Numérica por Trapézios:

```
---------------------------------
function I=trapezio(def,a,b,n)
f=inline(def);
I=0
h=(b-a)/(n-1)
for k=2:n-1
x=a+h*(k-1)
I=I+f(x)
end
I=(2*I+f(a)+f(b))*h/2
end
---------------------------------
```

Exemplo: Tomando o mesmo exemplo da seção anterior, ou seja, calcular a integral

$$\int_0^{\pi/2} \cos(x)dx$$

cujo valor analítico conhecemos como sendo igual a 1. Se usarmos o método dos trapézio nesse caso, com 2 pontos por exemplo (1 único trapézio), temos que



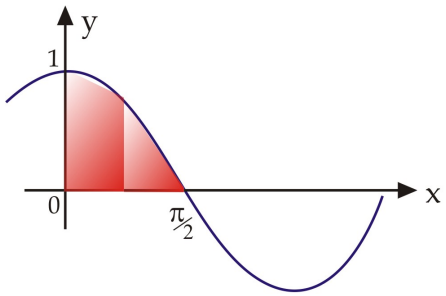
e ao chamarmos o código, teremos que

```
>> trapezio('cos(x)',0,pi/2,2)
```

que resulta em

```
>> ans=
0.7854
```

Se aumentarmos o número de pontos para 3 (2 trapézios) teriamos a configuração



e ao chamarmos o código, teremos que

```
>> trapezio('cos(x)',0,pi/2,3)
```

que resulta em

```
>> ans=
0.9481
```

O que já resulta em uma aproximação bem mais aceitável. Em geral o método do trapézio é mais preciso do que Riemann por usar uma função linear (reta) para aproximar a função entre cada x_i. Sendo assim, ao fazermos para a sequência de ponto $n = 10, 100, 200$ temos que

```
>> trapezio('cos(x)',0,pi/2,10)
```

que resulta em

```
>> ans=
0.997460
```

```
>> trapezio('cos(x)',0,pi/2,100)
```

que resulta em

```
>> ans=
0.9999790
```

```
>> trapezio('cos(x)',0,pi/2,200)
```

que resulta em

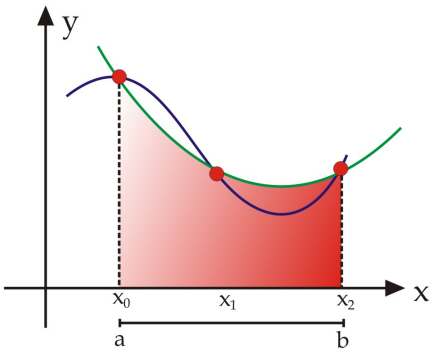
```
>> ans=
0.9999948
```

7.3 Método de Simpson

No método de Simpson aproximaremos $f(x)$ por um polinômio de grau 2 e sendo assim, precisaremos de três pontos do intervalo $[a,b]$. Utilizando $x_0 = a$, $x_1 = \frac{a+b}{2}$ e $x_2 = b$. Nesse caso aproximaremos utilizando a forma de Lagrange vista na iterpolação polinomial, com $f(x_i) = y_i$, então teremos que

$$f(x) \approx p_2(x) = f(x_0)\ell_0(x) + f(x_1)\ell_1(x) + f(x_2)\ell_2(x) \tag{7.5}$$

e aproximaremos a área conforme a figura abaixo:



Então ao integrarmos $f(x)$ podemos substituir por $p_2(x)$ que fica

$$\begin{aligned}\int_a^b f(x)dx \approx \int_a^b p_2(x)dx &= \int_a^b (f(x_0)\ell_0(x) + f(x_1)\ell_1(x) + f(x_2)\ell_2(x))dx \\ &= f(x_0)\int_a^b \ell_0(x)dx + f(x_1)\int_a^b \ell_1(x)dx + f(x_2)\int_a^b \ell_2(x)dx\end{aligned}$$

Ao integrarmos cada uma das funções de segundo grau ℓ_i teremos que a equação acima é

$$\int_a^b f(x)dx \approx \int_a^b p_2(x)dx = f(x_0)\frac{h}{6} + f(x_1)\frac{4h}{6} + f(x_2)\frac{h}{6} \tag{7.6}$$

o que configura a regra geral de Simpson e seu código em Matlab é descrito por

Código em Matlab para Integração Numérica por Simpson:

```
-----------------------------------
function I=simpson(def,a,b,n)
f=inline(def);
I=f(a)+f(b);
h=(b-a)/(n-1)
w=4
```

```
for k=2:n-1
x=a+h*(k-1)
I=I+f(x)*w
w=6-w
end
I=I*h/3
end
---------------------------------
```

Exemplo: Tomando o mesmo exemplo da seção anterior, ou seja, calcular a integral

$$\int_0^{\pi/2} \cos(x)dx$$

cujo valor analítico conhecemos como sendo igual a 1. Se usarmos o método de Simpson, com 3 pontos por exemplo, temos que ao chamar o código, nos retorna

```
>> simpson('cos(x)',0,pi/2,3)
```

que resulta em

```
>> ans=
1.0022
```

que já é uma excelente aproximação, pois usamos apenas três pontos. No método de Simpson podemos usarmos mais pontos se julgarmos necessário, porém como a cada três pontos usaremos uma função de grau 2, devemos ter sempre um número de pontos ímpar. De maneira que

$$\int_a^b f(x)dx \approx \frac{h}{6}\left[f(x_0) + 4f(x_1) + f(x_2)\right] \tag{7.7}$$

aproxima para três pontos e

$$\begin{aligned}\int_a^b f(x)dx &\approx \frac{h}{6}\left[f(x_0) + 4f(x_1) + f(x_2) + f(x_2) + 4f(x_3) + f(x_4)\right] \\ &\approx \frac{h}{6}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + f(x_4)\right] \end{aligned} \tag{7.8}$$

aproxima para cinco pontos. E em geral temos que

$$\int_a^b f(x)dx \approx \frac{h}{6}\left[f(x_0) + 4\sum_{i=0}^{n-1} f(x_{2i+1}) + 2\sum_{i=0}^{n-1} f(x_{2i}) + f(x_n)\right] \tag{7.9}$$

7.4 Exercícios

Integração Numérica:

1. Considerando cada item abaixo, calcule as integrais usando os métodos de Reimann (Esquerda, Direita), Trapézios e Simpson para $n = 1, 3, 11, 51$ em cada caso. Nos itens de (a) até (d) compare com o resultado analítico.

   (a) $\int_2^5 x^2 - 4dx$.

   (b) $\int_1^4 \ln(x)dx$.

   (c) $\int_0^\pi \sin(x)dx$.

   (d) $\int_0^3 9 - x^2 dx$.

   (e) $\int_1^4 \sqrt{x^2+1}dx$.

   (f) $\int_0^{0,25} \tan(x)dx$.

   (g) $\int_{-1}^4 \frac{1}{x+2}dx$.

   (h) $\int_0^1 x^9 dx$.

2. Calcule numericamente o valor de $\int_2^5 e^{4-x^2}dx$ usando os métodos de Riemann à esquerda, trapézio e Simpson. Obtenha os resultados utilizando, em cada caso, o número de pontos indicado:

n	Riemann	Trapézio	Simpson
3			
5			
7			
9			

3. Use a regra de Simpson para aproximar a integral

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

   Depois divida a integral em duas

$$\int_{-1}^0 e^{-x^2}dx$$

   e

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

e aplique a regra de Simpson em cada uma delas. Tome sempre número de pontos ímpares.

4. O valor exato da integral imprópria

$$\int_0^1 x \ln(x) dx = -\frac{1}{4}$$

Aproxime o valor desta integral usando a regra de Simpson para $n = 3$, $n = 5$ e $n = 7$. Como você avalia a qualidade do resultado obtido? Por que isso acontece.

5. Aproxime numéricamente a integral

$$\int_0^3 \frac{1}{x^2} dx$$

usando Riemann à direita e explique por não é possível aproximar usando Riemann à direita e os demais métodos. Seria possível realizar algum tipo de modificação para usarmos os demais métodos?

6. Estamos interessados em avaliar numericamente a seguinte integral:

$$\int_0^1 \ln(x) \sin(x) dx$$

cujo valor com 10 casas decimais corretas é -0.2398117420. Chegue nesse resultado usando um dos métodos apresentados em aula e diga quantos pontos foram necessários para isso e o porque da escolha do método.

7. Considere a integral abaixo

$$\int_\varepsilon^\gamma \frac{1}{x^2} dx$$

Considere as combinações sujeridas na tabela abaixo e calcule a integral acima usando o método de Simpson.

n	$\varepsilon = 0.1$ e $\gamma = 10$	$\varepsilon = 0.01$ e $\gamma = 100$	$\varepsilon = 0.001$ e $\gamma = 100$
5			
15			
29			
99			

Os resultados obtidos são esperados conforme a forma da curva a ser integrada?

8 Solução Numérica de EDO's

Visamos neste capítulo, discutir algumas técnicas numéricas para aproximar a solução de diversos tipos de equações diferenciais. Muitas equações diferenciais podem ser de díficil resolução analítica devido a alta ordem ou também à imposições de condições iniciais ou de contorno complicadas. Discutiremos aqui alguns métodos que auxiliam na obtenção numérica dessas soluções. Os tipos de equações que vamos trabalhar variam muito e iniciaremos com problemas de valor inicial de priemeira ordem.

8.1 Problemas de Valor Inicial de primeira ordem

Vamos aproximar a solução do problema de valor inicial (PVI) dado pela equação diferencial ordinária (EDO) de primeira ordem descrita por

$$PVI = \begin{cases} y' = f(x, y(x)) \\ y(x_0) = y_0 \end{cases} \tag{8.1}$$

onde tratamos $y(x)$ com uma função de x, o ponto (x_0, y_0) é conhecido e a função $f(x,y)$ uma função que depende de $y(x)$ e da variável independente x. Vejamos alguns exemplos que elucidam o problema de valor inicial

Exemplo 1:

$$PVI = \begin{cases} y' = x^2 \\ y(0) = 1 \end{cases} \tag{8.2}$$

onde nesse caso a derivada a função $y(x)$ é x^2 e o único ponto conhecido desse PVI é o $(0,1)$.

Exemplo 2:

$$PVI = \begin{cases} y' = \ln(x) - y \\ y(2) = 7 \end{cases} \tag{8.3}$$

onde nesse caso o PVI é originado da EDO dada por $y' + y - \ln(x) = 0$ e o único ponto conhecido desse PVI é o $(2,7)$.

A grande maioria desses problemas de valor inicial não podem ser resolvidos analpiticamente, isto é, sabemos que a solução existe e é única, mas não podemos determiná-la usando funções conhecidas. Origina dai o fato de que devemos aproximar numéricamente as soluções desses EDO's. Vamos iniciar a obtenção dos nossos métodos utilizando um conjunto de pontos limitados em x ao qual chamaremos de malha.

Devemos ter um cuidado analisando se o PVI em questão é um problema bem posto. Em outras palavras, devemos nos perguntar se:

1. Existe uma solução para o PVI?

2. A solução é única?

3. A solução do PVI é estável? Ou seja, é pouco sensível a pequenas perturbações nas condições iniciais?

Definição: A função $f(x,y)$ é Lipschitz em x se existe uma constante L, tal que $x \in [a,b]$ e $y, g \in \mathbb{R}$,

$$|f(x,y) - f(x,g)| \leq L|y(x) - g(x)| \tag{8.4}$$

Teorema: *Seja $f(x,y)$ contínua em x e Lipschitz em y. Então existe uma única solução para o PVI*

$$PVI = \begin{cases} y' = f(x,y) \\ y(x_0) = y_0 \end{cases} \tag{8.5}$$

.

Uma vez garantida as condições temos que a solução será única e então podemos analisar maneiras diferentes de aproximar numéricamente a EDO deste PVI.

8.2 Método de Euler

Considerando novamente o PVI dado anteriormente

$$PVI = \begin{cases} y' = f(x,y(x)) \\ y(x_0) = y_0 \end{cases} \tag{8.6}$$

com a garantia de que a solução é única, podemos realizar a integração na variável x em um intervalo (x_0, x_1) de comprimento h em ambos lados e teremos que

$$\int_{x_0}^{x_1} y'(x)dx = \int_{x_0}^{x_1} f(x,y(x))dx \tag{8.7}$$

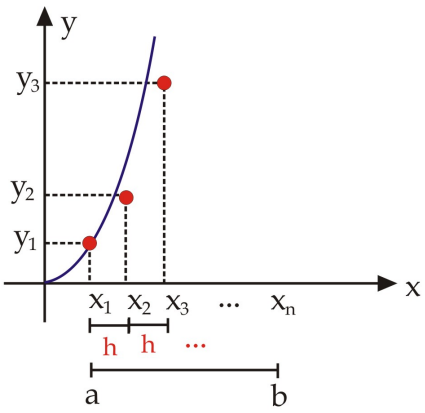
e posteriomente

$$\begin{aligned} \int_{x_0}^{x_1} y'(x)dx &= \int_{x_0}^{x_1} f(x,y(x))dx \\ y(x_1) - y(x_0) &= \int_{x_0}^{x_1} f(x,y(x))dx \\ y(x_1) &= y(x_0) + \int_{x_0}^{x_1} f(x,y(x))dx \end{aligned}$$

e aqui aproximaremos a integral a direita condirendo $f(x,y(x))$ uma função constante em x, isto é, usaremos a aproximação integral de Riemann à esquerda, pois não temos $y(x_1)$ e então temos que

$$
\begin{aligned}
y(x_1) &= y(x_0) + \int_{x_0}^{x_1} f(x, y(x))dx \\
y(x_1) &= y(x_0) + f(x, y(x)) \int_{x_0}^{x_1} dx \\
y(x_1) &= y(x_0) + (x_1 - x_0) f(x, y(x)) \\
y(x_1) &= y(x_0) + hf(x, y(x))
\end{aligned}
$$

assim cada ponto da função será aproximado mediante a escolha de um passo h e a partir do ponto $(x_0, y(x_0))$ como mostra a figura



onde o tamanho do passo é dado por $h = \frac{b-a}{N}$ e N é o número de iterações a serem realizadas. O Método de Euler tem equação geral dada por

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{8.8}$$

onde tomamos $y_i = y(x_i)$ para facilitar a notação. Sabemos também que cada x_i é dado por $x_i = x_o + ih$.

Código em Matlab para Solução Numérica de EDO por Euler:

```
----------------------------------
function [x,y]=euler(def,a,b,y0,N)
f=inline(def);
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x(1)=a;
y(1)=y0;
for i=1:N;
y(i+1)=y(i)+h*feval(f,x(i),y(i));
x(i+1)=x(i)+h;
end
```

```
y
plot(x,y);
end
--------------------------------
```

Exemplo 1: Consideramos o exemplo de aproximarmos a solução da EDO dada pelo PVI abaixo

$$PVI = \begin{cases} y' = 3x^2 \\ y(1) = 1 \end{cases} \tag{8.9}$$

Vamos aproximar o valor de $y(2)$ usando o Método de Euler com $h = 0.1$. Para isso, vamos usar $N = 10$, pois

$$N = \frac{b-a}{h} = \frac{2-1}{0.1} = 10$$

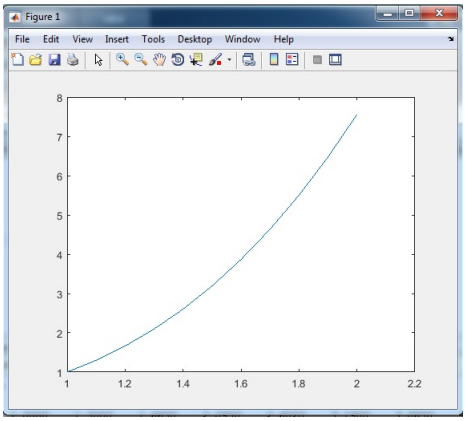
Assim devemos chamar o programa com

```
>> euler('3*x.^2+0*y',1,2,1,10)
```

e teremos como resposta

```
y =
1.0000    1.3000    1.6630    2.0950    2.6020    3.1900
3.8650    4.6330    5.5000    6.4720    7.5550
```

que representam as aproximações de em cada x_i de 1 até 2. A aproximação que desejamos é em $x = 2$ e portanto 7.5550. O programa também mostra o gráfico do comportamento da solução em $(1, 2)$:



Ao tentarmos com valores diferentes de h, como por exemplo $h = 0.05$, temos

```
>> euler('3*x.^2+0*y',1,2,1,20)
```

e teremos como resposta

```
y=
1.0000    1.1500    1.3154    1.4969    1.6952    1.9112
2.1456    2.3991    2.6725    2.9665    3.2819    3.6194
```

```
3.9798    4.3638    4.7721    5.2056    5.6650    6.1510
6.6644    7.2059    7.7763
```

Aqui a aproximação é dada por $y(2) = 7,7763$. Como sabemos que a solução exata para esse caso é $y(x) = x^3$ temos que o valor exato em 2 é $y(2) = 2^3 = 8$, ou seja, as aproximações tendem ao valor exato. Uma observcação importante é de que o comando `'inline'` do Matlab associa o número de variáveis da função como o número variáveis cadastradas. Por exemplo, cadastramos `'3*x.^2+0*y'` ao invés de `'3*x.^2'`, pois o programa considera as funções para iterções uma função de duas variáveis.

8.3 Taylor de Ordem n

O método de Taylor consiste em aproximar a solução da equação diferencial usando a série de Taylor truncada na ordem h^n com n desejado, o que da nome ao método. Ao considerarmos a série de Taylor como segue

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}y''(x) + \frac{h^3}{3!}y'''(x) + ... + \frac{h^n}{n!}y^{(n)}(x) + O(h^{(n+1)}) \quad (8.10)$$

mas como estamos tratando de aproximar o PVI

$$PVI = \begin{cases} y' = f(x, y(x)) \\ y(x_0) = y_0 \end{cases} \quad (8.11)$$

podemos substituir na série de Taylor $y(x)$ por $f(x, y(x))$ ficando com

$$y(x+h) = y(x) + hf(x, y(x)) + \frac{h^2}{2!}f'(x, y(x)) + \frac{h^3}{3!}f''(x, y(x)) + ... + \frac{h^n}{n!}f^{n-1}(x, y(x)) + O(h^{(n+1)}) \quad (8.12)$$

Assim a dedução do Método de Taylor para a ordem desejada é feita apenas com um truncamento na série acima até termos $O(h^{(n)})$ (de ordem $h^{(n)}$), isto é, ao fazermos

$$y(x+h) = \underbrace{\underbrace{y(x) + hf(x, y(x)) + \frac{h^2}{2!}f'(x, y(x))}_{\text{Taylor 2}} + \frac{h^3}{3!}f''(x, y(x)) + ... + \frac{h^n}{n!}f^{n-1}(x, y(x))}_{\text{Taylor 3}} + O(h^{(n+1)})}_{\text{Taylor n}} \quad (8.13)$$

Mostraremos aqui o código em Matlab para o caso de Taylor de ordem 2, porém os códigos para as demais ordens podem ser obtidos de maneira análoga.

Código em Matlab para Solução Numérica de EDO por Taylor n = 2:

```
----------------------------------
function [x,y]=taylor2(def,def2,a,b,y0,N)
f=inline(def);
f2=inline(def2);
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x(1)=a;
y(1)=y0;
for i=1:N;
y(i+1)=y(i)+h*feval(f,x(i),y(i))+(h^2/2)*feval(f2,x(i),y(i));
x(i+1)=x(i)+h;
end
y
plot(x,y);
end
----------------------------------
```

No caso de uma aproximação via Método de Taylor usando a ordem 2, devemos obter $f'(x,y(x))$, ou seja, precisamos derivar em relação a variável x pelo menos uma vez. Vejamos no exemplo a seguir:

Exemplo 2: Considerando o mesmo PVI do exemplo anterior

$$PVI = \begin{cases} y' = 3x^2 \\ y(1) = 1 \end{cases} \tag{8.14}$$

Vamos aproximar o valor de $y(2)$ usando o Método de Taylor de Ordem 2. Para isso, precisamos de $f'(x,y(x))$ que é dada por

$$f'(x,y(x)) = 6x \tag{8.15}$$

realizando as aproximações com $h = 0.1$. Para isso, vamos usar $N = 10$, pois

$$N = \frac{b-a}{h} = \frac{2-1}{0.1} = 10$$

Assim devemos chamar o programa com

```
>> taylor2('3*x.^2+0*y','6*x+0*y',1,2,1,10)
```

e teremos como resposta

```
y =
1.0000    1.3300    1.7260    2.1940    2.7400    3.3700
4.0900    4.9060    5.8240    6.8500    7.9900
```

Aqui vemos a aproximação tende muito mais veloz ao valor exato que é 8. Porém se a função $f(x,y(x))$ não depender explicitamente apenas de x, devemos ter atenção na derivação como veremos no exemplo a seguir.

Exemplo 3: Considere o PVI dado por

$$PVI = \begin{cases} y' = 3x^2 + y^2 \\ y(0) = 1 \end{cases} \tag{8.16}$$

Vamos aproximar o valor de $y(1)$ usando o Método de Taylor de Ordem 2. Para isso, precisamos de $f'(x, y(x))$ que é dada por

$$f'(x, y(x)) = 6x + 2yy'$$

e sabemos, utilizando o próprio PVI, que $y' = 3x^2 + y^2$, então na verdade, teremos que

$$f'(x, y(x)) = 6x + 2y(3x^2 + y^2)$$

realizando as aproximações com $h = 0.1$. Para isso, vamos usar $N = 10$, pois

$$N = \frac{b-a}{h} = \frac{1-0}{0.1} = 10$$

Assim devemos chamar o programa com

```
>> taylor2('3*x.^2+y.^2','6*x+2*y*(3*x.^2+y.^2)',0,1,1,10)
```

e teremos como resposta

```
y =
1.0000    1.1100    1.2532    1.4495    1.7299    2.1493
2.8166    3.9898    6.4433   13.6098   57.9422
```

Assim, temos que a aproximação é dada por $y(1) \approx 57,9422$.

8.4 Métodos de Passo Múltiplo

Todos métodos numéricos para a resolução de um PVI vistos até aqui baseiam-se em aproximar a solução via iterações que dependem unicamente do ponto anterior. Poderiamos classificá-los como métodos de passo simples ou de passo único. Os métodos de passo múltiplo usam dois ou mais pontos anteriores para a aproximar a solução e suas deduções resumem-se a aproximar a integral do remanescente do lado direito do PVI

$$PVI = \begin{cases} y' = f(x, y(x)) \\ y(x_0) = y_0 \end{cases}$$

pois temos (assim como fizemos na dedução do método de Euler) que ao integrar

$$\int_a^b y'(x)dx = \int_a^b f(x, y(x))dx$$

e se utilizarmos as fórmulas de Newton-Coates para ordem desejada, obteremos relação diferentes em cada lado, mas sempre combinações de y do lado esquerdo e de $f(x, y(x))$ do lado direito. Assim, de forma geral teremos que um método de passo múltiplo de ordem n é

$$a_n y_n + a_{n-1} y_{n-1} + \cdots + a_0 y_0 = b_n f_n + b_{n-1} f_{n-1} + \cdots + b_0 f_0 \tag{8.17}$$

onde a_i e b_i são coeficientes reais, $y_i = y(x_i)$ e $f_i = f(x_i, y_i)$. Um dos métodos mais usados é o método de Adams-Bashford de quarta ordem, que fica

$$y_4 = y_3 + \frac{h}{24}(55 f_3 - 59 f_2 + 37 f_1 - 9 f_0) \tag{8.18}$$

Note que o método necessita de informações ainda não calculadas como, por exemplo, os valores y_1, y_2 e y_3 que por sua vez são necessários para calcular f_1, f_2 e f_3. Os métodos de passo múltiplo, em geral, precisam do auxilio de métodos mais simples (como Euler, por exemplo) para o cálculo das primeiras aproximações. Isto é, podemos aproximar os primeiros ponto via método de Euler e continuar as aproximações via Adams. Segue abaixo o código em Matlab que usa como predictor o método de Euler.

Código em Matlab para Solução Numérica de EDO por Adams (Euler Predictor):

```
---------------------------------
function [x,y]=adams(def,a,b,y0,N)
f=inline(def);
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x(1)=a;
y(1)=y0;
for i=1:3;
y(i+1)=y(i)+h*feval(f,x(i),y(i));
x(i+1)=x(i)+h;
end
for i=4:N;
y(i+1)=y(i)+(h/24)*(55*feval(f,x(i),y(i))-59*feval(f,x(i-1),y(i-1))...
...+37*feval(f,x(i-2),y(i-2))-9*feval(f,x(i-3),y(i-3)));
x(i+1)=x(i)+h;
end
y
plot(x,y);
end
---------------------------------
```

8.5 Métodos de Runge-Kutta

Os métodos de Runge Kutta visam solucionar um problema de valor incial usando ponderações e incrementos nas funções dentro de seus próprios argumentos. Os métodos em questão tem forma geral em ordem n dada por

$$y_{n+1} = y_n + \alpha_1 k_1 + \alpha_2 k_2 + \cdots + \alpha_n k_n \tag{8.19}$$

onde os valores de cada k dependem da função f e de incrementos em x (que serão combinações de h) e incrementos em y (que serão combinações dos valores anteriores já obtidos de k). Trabalharemos aqui com os métodos de ordem 2 e 4. O método de Runge-Kutta é deduzido comparando cada parcela e incremento com a série de Taylor da mesma ordem a que se deseja usar no método. O método para ordem 2 fica

$$y_{n+1} = y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2 \tag{8.20}$$

com

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + h, y_n + k_1) \end{aligned}$$

e seu código em Matlab é descrito abaixo:
Código em Matlab para Solução Numérica de EDO por Runge-Kutta n = 2:

```
---------------------------------
function [x,y]=rk2(def,a,b,y0,N)
f=inline(def);
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x(1)=a;
y(1)=y0;
for i=1:N;
k1(i)=h*feval(f,x(i),y(i));
k2(i)=h*feval(f,x(i)+h,y(i)+k1(i));
y(i+1)=y(i)+(1/2)*(k1(i)+k2(i));
x(i+1)=x(i)+h;
end
y
plot(x,y);
end
---------------------------------
```

Para o caso de ordem 4 teremos uma formulação similar com ponderações e incrementos diferentes. Ficando com

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \tag{8.21}$$

com

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\
k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\
k_4 &= hf(x_n + h, y_n + k_3)
\end{aligned}
$$

e seu código em Matlab é descrito abaixo:
Código em Matlab para Solução Numérica de EDO por Runge-Kutta $n = 4$:

```
---------------------------------
function [x,y]=rk4(def,a,b,y0,N)
f=inline(def);
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x(1)=a;
y(1)=y0;
for i=1:N;
k1(i)=h*feval(f,x(i),y(i));
k2(i)=h*feval(f,x(i)+0.5*h,y(i)+0.5*k1(i));
k3(i)=h*feval(f,x(i)+0.5*h,y(i)+0.5*k2(i));
k4(i)=h*feval(f,x(i)+h,y(i)+k3(i));
y(i+1)=y(i)+(1/6)*(k1(i)+2*k2(i)+2*k3(i)+k4(i));
x(i+1)=x(i)+h;
end
y
plot(x,y);
end
---------------------------------
```

8.6 Sistemas de Equações Diferenciais

Certos problemas em engenharia envolvem mais de uma EDO no seu processo de solução. Se estas EDO's estiverem acopladas é possível sempre resolvermos utilizando os métodos já vistos nesse capítulo. Considerando que os problemas em geral podem ser descritos por

$$
\begin{aligned}
y' &= f(x, y(x), w(x)) \\
w' &= g(x, y(x), w(x)) \\
y(x_0) &= y_0 \\
w(x_0) &= w_0
\end{aligned}
$$

o processo de resolução segue de maneira similar. Vamos aproximar cada caso com passo h por Euler como sendo

$$y_1 = y_0 + hf(x_0, y_0, w_0) \tag{8.22}$$

e também

$$w_1 = w_0 + hg(x_0, y_0, w_0) \tag{8.23}$$

Teremos então para esse caso que o código em Matlab é descrito por
Código em Matlab para Solução Numérica de Sistema de EDO por Euler:

```
-----------------------------------
function [x,y,w]=euler_sis(def,def2,a,b,y0,w0,N)
f=inline(def);
g=inline(def2);
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x(1)=a;
y(1)=y0;
w(1)=w0;
for i=1:N;
y(i+1)=y(i)+h*feval(f,x(i),y(i),w(i));
w(i+1)=w(i)+h*feval(g,x(i),y(i),w(i));
x(i+1)=x(i)+h;
end
y
w
plot(x,y,x,w);
end
-----------------------------------
```

Exemplo 1: Considere o sistema de EDO's dado por:

$$\begin{aligned} y' &= x + y - w \\ w' &= y - x \\ y(0) &= 1 \\ w(0) &= 1 \end{aligned}$$

vamos aproximar as soluções de y e w no intervalo de 0 a 1. Para tal, vamos usar Euler para aproximar ambas soluções, podemos chamar o código

```
>> euler_sis('x+y-w','y-x+0*w',0,1,1,1,10)
```

e teremos como resposta que

```
y=
1.0000    1.2000    1.4100    1.6290    1.8556    2.0879
2.3236    2.5597    2.7925    3.0177    3.2301
```

```
w =
1.0000    1.0000    0.9800    0.9370    0.8678    0.7690
0.6371    0.4685    0.2594    0.0060   -0.2951
```
que correspondem as aproximações nesse intervalo das funções y e w.

8.7 Solução de PVI's de ordem $n \geq 2$

As soluções vistas anteriormente para o caso de sistemas lineares não restringem sua utilidade apenas à eles. Equações diferenciais de ordem maior podem ser resolvidas utilizando a mesma teoria. Iniciamos abordando o seguinte problema:

$$\begin{aligned} y'' + xy' - y &= 2 \\ y(1) &= 1 \\ y'(1) &= 0 \end{aligned} \tag{8.24}$$

o que exatamente devemos fazer para obter a aproximação desejada em um intervalo dado?

O procedimento mais adequado nesse caso é a introdução de uma variável auxiliar para solução desse sistema. Seja w uma função auxiliar tal que

$$w(x) = y'(x) \tag{8.25}$$

e consequentemente teremos que

$$w'(x) = y''(x) \tag{8.26}$$

Ao substituirmos y'' por w' e y' por w na equação diferencial do problema, ficaremos com $w' + xw - y = 2$ tornando-a uma equação diferencial de ordem 1 e juntamente com $y' = w$ formam o seguinte sistema

$$\begin{aligned} w' &= 2 - xw + y \\ y' &= w \\ y(1) &= 1 \\ w(1) &= 0 \end{aligned}$$

o qual procederemos de maneira análoga à seção anterior para obter a aproximação.

Exemplo 1: Vamos resolver o problema

$$\begin{aligned} y'' + xy' - y &= 2 \\ y(1) &= 1 \\ y'(1) &= 0 \end{aligned} \tag{8.27}$$

no intervalo de $[1,2]$ com $h = 0{,}1$. Faremos então que $y' = w$ e $y'' = w'$ ficando com

$$\begin{aligned} w' &= 2 - xw + y \\ y' &= w \\ y(1) &= 1 \\ w(1) &= 0 \end{aligned}$$

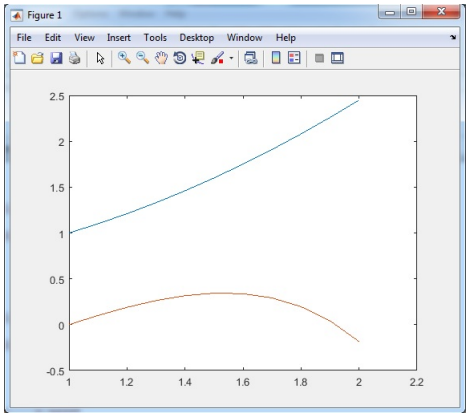
que é um sistema de EDO's. Usando o Matlab para resolvê-lo, chamamos o código

```
>> euler_sis('-x*w+y+2','w+0*x+0*y',1,2,0,1,10)
```

obtendo:

```
y =
1.0000    1.1000    1.2100    1.3300    1.4600    1.6000
1.7500    1.9100    2.0800    2.2600    2.4500
w =
0.0000    0.1000    0.1890    0.2627    0.3161    0.3433
0.3376    0.2914    0.1958    0.0410   -0.1843
```

e o comportamento gráfico



onde claramente a função que começa em 1 (condição inicial) é a solução aproximada para y no intervalo dado.

8.8 Exercícios

Solução Numérica de EDO's :

1. Considerando cada PVI abaixo e realize as aproximações pedidas usando o Método de Euler.

   (a) $\begin{cases} y' = 3x^2 + y^2 \\ y(0) = 1 \end{cases}$ com $h = 0.1$ aproxime $y(1)$.

   (b) $\begin{cases} y' = \frac{1}{x} \\ y(10) = 1 \end{cases}$ com $N = 20$ aproxime $y(12)$.

   (c) $\begin{cases} y' = y + x^2 \\ y(2) = 3 \end{cases}$ com $N = 10$ aproxime $y(4)$.

2. Considerando cada PVI abaixo e realize as aproximações pedidas usando o Método de Taylor de ordem 2.

   (a) $\begin{cases} y' = 3x^2 + x \\ y(1) = 1 \end{cases}$ com $N = 10$ aproxime $y(2)$.

   (b) $\begin{cases} y' = x^2 + y \\ y(2) = 2 \end{cases}$ com $h = 0.01$ aproxime $y(2.3)$.

   (c) $\begin{cases} y' = \cos x + x \\ y(1) = 0.5 \end{cases}$ com $h = 0.1$ aproxime $y(2)$.

3. Crie um código em Matlab que realize as aproximações de Taylor de ordem 3 usando como base o código fornecido de ordem 2. Teste para o item (a) do exercício 1 e veja se os resultados são coerentes.

4. Calcule numericamente o valor do PVI

$$\begin{cases} y' = 2x^2 + x \\ y(2) = 3 \end{cases} \tag{8.28}$$

   diga qual é a aproximação de $y(3)$ usando os métodos de Euler, Taylor de ordem 2 e de Taylor de ordem 3 e preencha a tabela abaixo.

N	Euler	Taylor ordem 2	Taylor ordem 3
5			
10			
15			
30			

5. Considerando um sistema massa mola em que a sua deriva da posição é dada por

$$y'(x) = c_1 w \cos(wx) - c_2 w \sin(wx)$$

onde $w = \frac{2\pi}{T}$, $c_1 = 2$, $c_2 = -9$, $T = 2$ com posição inicial dada por $y(0) = 3$. Aproxime $y(3)$ com $h = 0.1$.

6. Considerando um modelo logístico de crescimento populacional que tem por definição a EDO dada por

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$$

onde N é a população de uma cerca cultura, r a taxa de crescimento natural e K a capacidade suporte do meio. Então considerando t dado em anos, aproxime a população após 2 anos com $r = 1,2$ e capaxidade suporte do meio de 10000 com população inicial de 2000 usando um dos métodos numéricos. Iterprete o resultado e diga o que tende a acontecer com o aumento dos anos.

7. Interprete o que aconteceria se na dedução do método de Euler usassemos para aproximar a integral por Riemann à direita e não à esquerda. Seria possível usar Euler? Em qualquer situação?

8. Considerando cada PVI abaixo e realize as aproximações pedidas usando o Método de Adams.

(a) $\begin{cases} y' = 3x^2 \\ y(1) = 1 \end{cases}$ com $h = 0.2$ aproxime $y(4)$.

(b) $\begin{cases} y' = \frac{1}{x+1} \\ y(10) = 1 \end{cases}$ com $N = 10$ aproxime $y(11)$.

(c) $\begin{cases} y' = -y + x^2 + 1 \\ y(0) = 1 \end{cases}$ com $N = 20$ aproxime $y(2)$.

9. Considerando cada PVI abaixo e realize as aproximações pedidas usando o Método de Runge-Kutta de ordem 2 e 4.

(a) $\begin{cases} y' = -xy^2 \\ y(1) = 1 \end{cases}$ com $N = 20$ aproxime $y(2)$.

(b) $\begin{cases} y' = x^2 + y \\ y(2) = 2 \end{cases}$ com $h = 0.1$ aproxime $y(2.5)$.

(c) $\begin{cases} y' = \cos x + x \\ y(1) = 0.5 \end{cases}$ com $h = 0.1$ aproxime $y(2)$.

10. Crie um código em Matlab que realize as aproximações via método de Adams com o método de Taylor de ordem 2 como predictor usando como base o código fornecido de Adams com Euler de predictor. Teste para para algum exercício da lista já resolvido e veja se os resultados são coerentes.

11. Calcule numericamente o valor do PVI

$$\begin{cases} y' = -yx^2 \\ y(0) = 2 \end{cases} \tag{8.29}$$

diga qual é a aproximação de $y(2)$ usando os métodos de Adams, Runge-Kutta de ordem 2 (RK2) e 4 (RK4) e preencha a tabela abaixo.

N	Adams	RK2	RK4
5			
10			
15			
30			

12. Use o método de Euler e de Runge Kutta ordem 2 para obter uma aproximação numérica do valor de $u(1)$ quando $u(t)$ satisfaz o seguinte problema de valor inicial

$$\begin{cases} u'(t) = -u(t) + e^{u(t)} \\ u(0) = 0 \end{cases}$$

usando passos $h = 0,1$ e $h = 0,01$.

13. Considere o seguinte modelo para a evolução da velocidade de um objeto em queda (unidades no SI):

$$v' = g - \alpha v^2$$

Sabendo que $g = 9,8$ e $\alpha = 10^{-2}$ e $v(0) = 0$. Pede-se a velocidade ao tocar o solo, sabendo que a altura inicial era 100.

14. Aproxime a solução do PVI

$$\begin{cases} \frac{dy}{dt} = \sin(t) \\ y(0) = 1 \end{cases}$$

para $t \in [0, 10]$.

(a) Plote a solução para $h = 0.16, 0.08, 0.04, 0.02, 0.01$ para o método de Taylor de ordem 2 e Adams.

(b) Utilizando a solução exata, plote um gráfico do erro em escala logarítmica. Comente os resultados (novamente, em cada gráfico separado para cada método repita os valores acima)

(c) Fixe agora o valor $h = 0.02$ e plote no mesmo gráfico uma curva para cada método.

15. Considere o seguinte modelo para o crescimento de uma colônia de bactérias:

$$\frac{du}{dt} = \alpha u(A-u) \tag{8.30}$$

onde u indica a densidade de bactérias em unidades arbitrárias na colônia e α e A são constantes positivas. Pergunta-se:

(a) Qual a solução quando a condição inicial $u(0)$ é igual a 0 ou A?

(b) O que acontece quando a condição inicial $u(0)$ é um número entre 0 e A?

(c) O que acontece quando a condição inicial $u(0)$ é um número negativo?

(d) O que acontece quando a condição inicial $u(0)$ é um número positivo maior que A?

(e) Se $A = 10$ e $\alpha = 1$ e $u(0) = 1$, use métodos numéricos para obter tempo necessário para que a população dobre?

(f) Se $A = 10$ e $\alpha = 1$ e $u(0) = 4$, use métodos numéricos para obter tempo necessário para que a população dobre?

16. Considerando cada sistema de EDO abaixo, realize as aproximações pedidas usando o Método de Euler para sistemas.

(a) $\begin{cases} y' = w + y^2 \\ w' = x - y \\ y(0) = 1 \\ w(0) = 2 \end{cases}$ com $h = 0.1$ aproxime no intervalo $[0, 2]$.

(b) $\begin{cases} y' = w - y \\ w' = x^2 - 2y \\ y(1) = 2 \\ w(1) = 2 \end{cases}$ com $N = 20$ aproxime no intervalo $[1, 2]$.

(c) $\begin{cases} y' = \cos(x) + y^2 + w \\ w' = w^2 - 2 \\ y(\pi/2) = 1 \\ w(\pi/2) = 1 \end{cases}$ com $N = 10$ aproxime no intervalo $[\pi/2, \pi]$.

17. Considere a equação do pêndulo dada por:

$$\frac{d^2\theta(t)}{dt^2} + \frac{g}{\ell}\sin(\theta(t)) = 0$$

onde g é força da gravidade e ℓ o cumprimento da haste. Resolva o sistema para $g = 9,8$ e $\ell = 1$. Resolva para:

(a) $\theta(0) = 0.5$ $\theta'(0) = 0$.

(b) $\theta(0) = 1.0$ $\theta'(0) = 0$.

(c) $\theta(0) = 1.5$ $\theta'(0) = 0$.

18. Considere o modelo simplificado de FitzHugh-Nagumo para o potencial elétrico sobre a membrana de um neurônio:

$$\begin{aligned} \frac{dV}{dt} &= V - \frac{V^3}{3} - W + I \\ \frac{dW}{dt} &= 0.08(V + 0.7 - 0.8W) \end{aligned}$$

onde I é a corrente de excitação. Sabendo que as condições iniciais são $(V_0, W_0) = (0.2, 0.5)$, resolva numericamente o sistema com condições iniciais dadas e com

(a) $I = 0$.

(b) $I = 0.2$.

(c) $I = 0.4$.

(d) $I = 0.8$.