Matemática Computacional
Introdução

Sumário

1 Contextualização e importância da Matemática Computacional

A **Matemática Computacional** é uma área da Matemática Aplicada cuja importância é facilmente explicada pelo papel fundamental que hoje em dia as **ferramentas computacionais** desempenham em todos os ramos das Ciências e Engenharia. Permite obter **resultados numéricos** para inúmeros problemas cuja solução analítica não é conhecida ou efetivamente calculável, o que a torna indispensável no estudo e na simulação de **modelos matemáticos** usados nas mais variadas áreas das Ciências e Engenharia. Por outro lado, é uma disciplina transversal às diferentes áreas da Matemática, já que o seu estudo e desenvolvimento assentam em conhecimentos de *Álgebra Linear*, *Análise Matemática*, *Equações Diferenciais*, etc. O desenvolvimento e implementação de algoritmos computacionais faz com que a *Programação* seja essencial na Matemática Computacional.

Em que situações surge a Matemática Computacional?
Muitos problemas da vida real, da indústria ao mercado financeiro, passando pela investigação científica em Física, Engenharia e Ciências da vida, têm "soluções matemáticas", mas é impossível colocá-las em prática sem a ajuda da computação.

Veremos, em seguida, alguns exemplos relacionados com os tópicos do programa de Matemática Computacional.

1.1 Sistemas de ponto flutuante e cálculo computacional

Desde 1985, a implementação em computador de um sistema de representação numérica e respetiva aritmética computacional seguem os padrões definidos pelo Institute of Electrical and Electronics Engineers (IEEE). Os computadores modernos usam a norma IEEE 754 para representar números reais, sendo IEEE 754-2008 a versão mais recente.

Segundo a norma IEEE 754, podemos considerar dois esquemas principais de representação de números reais: precisão simples de 32 bits e precisão dupla de 64 bits (recorde que um bit é um dígito binário). A representação dos números de ponto flutuante obedece a

$$x = (-1)^s (d_1.d_2...d_n)_2 \times 2^t,\ d_1 \neq 0,\ s \in \{0,1\}, t \in \mathbb{Z} \cap [t_{max}, t_{min}],$$

onde o número t representa o *expoente* de x, s o *sinal* do número e a $(d_1.d_2...d_n)_2$ chama-se *mantissa*. A condição de normalização $d_1 \neq 0$ neste caso significa simplesmente $d_1 = 1$.

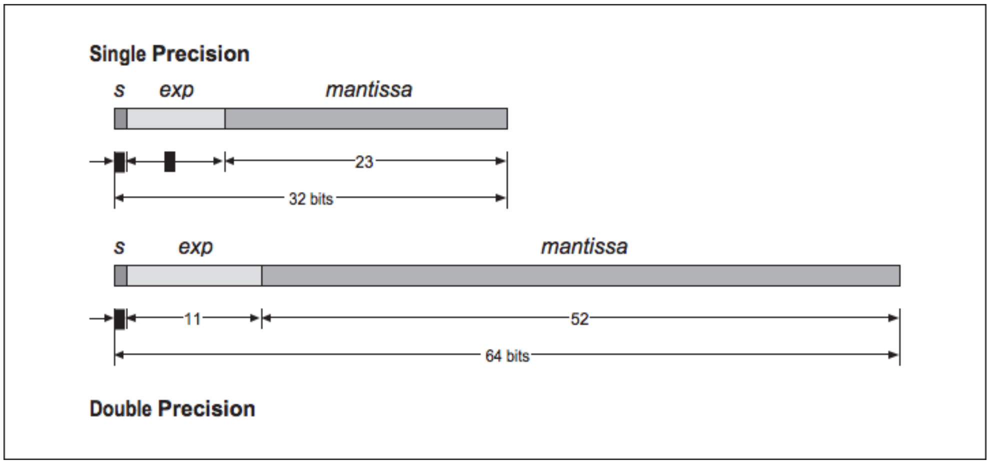


Figura 1: Sistemas de ponto flutuante definidos pela norma IEEE 754

As representações definidas pelas normas IEEE incluem ainda INF (infinito) e NaN (Not a Number). Por exemplo, $0/0$ e $\infty - \infty$ produzem NaN.

A representação de números inteiros no computador pode ser feita separadamente, usando esquemas de 8 bits, 16 bits, 32 bits ou 64 bits. Além disso, há dois subconjuntos de representação para inteiros: inteiros sem sinal (podem representar zero e inteiros positivos) e inteiros com sinal.

Vejamos alguns problemas que podem surgir com a representação de números no computador.

Exemplo 1.1. Um problema com a divisão em ponto flutuante no Pentium

Em 1993, um problema com os microprocessadores provocou uma falha na divisão de números em ponto flutuante. Por exemplo, ao dividir 4195835.0 por 3145727.0, o resultado apresentado era 1.33374 ao invés de 1.33382.

Ainda que o erro fosse pequeno (erro de 0.006%) e a falha afetasse poucos utilizadores, isto resultou num sério problema para a Intel, que se viu obrigada a trocar entre 3 e 5 milhões de chips, uma operação com custo superior a 500 milhões de dólares.

Exemplo 1.2. Desintegração do foguetão Ariane 5 devido a um erro numérico

No Ariane 5 foi reutilizado parte do código do Ariane 4, mas o computador de voo do novo foguetão incorporavam também, sem que ninguém desse conta, um erro de programação numa rotina aritmética. A 4 de junho de 1996, aquando do lançamento do Ariane 5, esse erro de programação provocou uma falha poucos segundos após a descolagem. Como consequência, meio segundo depois, o computador principal entrou em colapso. Assim, o Ariane

5 desintegrou-se 40s após o seu lançamento, com prejuízos estimados em várias centenas de milhões de euros.

A anomalia interna de software do SRI (Inertial Reference System) ocorreu durante a conversão de um número de 64 bits em ponto flutuante para um inteiro de 16 bits com sinal: o valor do número era maior do que seria possível representar como inteiro de 16 bits (overflow).

A título de curiosidade, referimos que o relatório do acidente foi elaborado por uma comissão presidida pelo matemático Jacques-Louis Lions (1928–2001), da Académie des Sciences (França). Informação adicional pode ser encontrada em

http://www-users.math.umn.edu/ arnold/disasters/ariane5rep.html

Exemplo 1.3. Um exemplo de cálculo em sistemas de ponto flutuante

Consideremos a função real de variável real

$$f(x) := \frac{1}{x} - \frac{1}{x+1} \quad (x \in \mathbb{R} \setminus \{-1, 0\})$$

No MATLAB *(IEEE-754 com precisão dupla) obtivémos o seguinte resultado:*

```
>> format long
>> 1/10^20-1/(10^20+1)
ans = 0
```

Note-se que ans = 0 tem erro de 100%.

Problema: *Como se explica este mau resultado? Como calcular um valor mais preciso para $f(10^{20})$?*

1.2 Resolução de equações não-lineares

O exemplo que se segue destina-se a motivar o estudo de métodos numéricos para resolver equações.

Exemplo 1.4. Um problema da Física

Na equação do foguete de Tsiolkovsky, a velocidade de um projétil é dada, em cada instante t, por

$$v(t) = u \ln\left(\frac{m_0}{m_0 - qt}\right) - gt$$

onde u é a velocidade de expulsão do combustível (em relação ao projétil), m_0 é a sua massa inicial, q é o coeficiente de consumo do combustível e $g = 9.8$ m/s^2 é a aceleração da gravidade.

Problema: *Determinar em que instante t a velocidade do projétil atinge o valor $v = 1000$ m/s, quando $u = 2200$ m/s, $m_0 = 16 \times 10^4$ kg e $q = 2680$ Kg/s.*

Numa primeira abordagem ao problema colocado no Exemplo 1.4, procedemos à localização gráfica de soluções de $v(t) = 1000$ na Fig.2. É claro que o estudo se deve limitar ao intervalo $]0, m_0/q[$ para t.

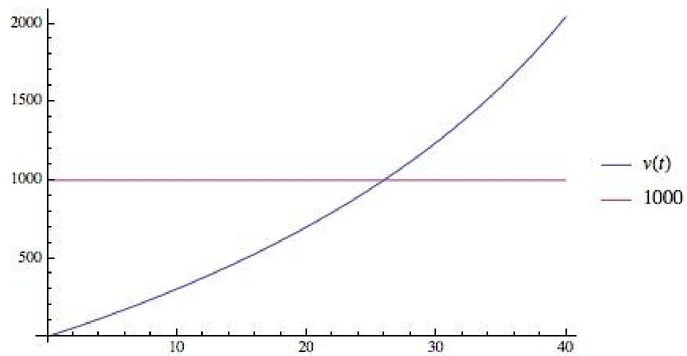


Figura 2: Localização das raízes através de esboço gráfico

Em termos da localização e separação de raízes de uma equação, é válido o seguinte resultado (teórico) que assegura a **existência e unicidade de solução de uma equação** $f(x) = 0$ num certo intervalo limitado.

Teorema 1.1. *Seja f uma função contínua em $[a, b]$ e diferenciável em $]a, b[$. Se*

- $f(a)f(b) < 0$
- *f' não se anula em $]a, b[$*

então existe um e um só $z \in]a, b[$ tal que $f(z) = 0$, ou, por outras palavras: no intervalo $]a, b[$, a equação $f(x) = 0$ tem uma e uma só solução.

Exemplo 1.5. *Voltando ao Exemplo 1.4, consideramos a equação*

$$2200 \ln \left(\frac{16 \times 10^4}{16 \times 10^4 - 2680t} \right) - 9.8t - 1000 = 0.$$

Vamos aplicar o Teorema 1.1 com $f(t) := 2200\ln\left(\frac{16\times10^4}{16\times10^4-2680t}\right) - 9.8t - 1000$ *e* $[a,b] = [20,30]$. *Tem-se*

$$f(20) = -298.47,\ f(30) = 241.951$$

$$f'(t) = -9.8 + \frac{589600}{16000-268t},\ f''(t) = \frac{9875800}{(4000-67t)^2},\ f \in C^2([20,30]).$$

Notamos que, de $f'' > 0$ *em* $[20,30]$ *e* $f'(20) > 0$, *resulta que* $f' > 0$ *em* $[20,30]$. *Como* $f' > 0$ *em* $[20,30]$ *e* $f(20)f(30) < 0$, *conclui-se que a equação tem em* $]20,30[$ *uma e uma só solução.*

Para terminar, vejamos alguns recursos computacionais que podemos usar para resolver equações não lineares e a sua aplicação à equação do foguete de Tsiolkovsky.

Exemplo 1.6. *Uma vez localizada a solução pretendida, é fácil obter uma solução para a equação* $f(t) = 0$ *usando software como o* MATHEMATICA *e o* MATLAB*:*

- MATHEMATICA

```
In[1]:= FindRoot[2200Log[16*10^4/(16*10^4-2680t)]-9.8t-1000,{t,20}]
Out[1]:={t->25.9424}

In[2]:= FindRoot[2200Log[16*10^4/(16*10^4-2680t)]-9.8t-1000,{t,30}]
Out[2]:={t->25.9424}

In[3]:= FindRoot[2200Log [16*10^4/(16*10^4-2680t)]-9.8t-1000,{t,20,30}]
Out[3]= {t->25.9424}
```

- MATLAB

```
>> fzero(@(t)2200*log(16*10^4/(16*10^4-2680*t))-9.8*t-1000,[20,30])
ans = 25.9424

>> fzero(@(t)2200*log(16*10^4/(16*10^4-2680*t))-9.8*t-1000, 20)
ans = 25.9424

>> fzero(@(t)2200*log(16*10^4/(16*10^4-2680*t))-9.8*t-1000, 30)
ans = 25.9424
```

- PYTHON

  Está disponível a função `nsolve`.

1.3 Ajustamento de dados discretos

Consideremos um caso prático como motivação.

Exemplo 1.7. Um problema de Biologia
Os dados representados na tabela

Dia	0	2	4	6	8	10	12	14	16	18	20
Quant. $\cdot 10^{-6}$	67.38	70.93	74.67	78.60	82.74	87.10	91.69	92.51	101.60	106.95	112.58

referem-se ao crescimento de uma bactéria num meio de cultura líquido ao longo de vários dias.

Problema: *Obter previsões da quantidade de bactérias ao fim de 15 e 30 dias.*

A resolução deste problema passa por construir uma função contínua que "*se ajuste*", num certo sentido, aos dados conhecidos. Depois, usa-se essa função para prever valores que não podem ser obtidos/medidos experimentalmente.

Há vários recursos computacionais disponíveis que permitem efetuar ajustamento de dados discretos.

Exemplo 1.8. *Para interpolação polinomial e ajustamento segundo o critério dos mínimos quadrados destacamos:*

• MATHEMATICA

A função `InterpolatingPolynomial` *faz* **interpolação polinomial** *e a função* `Fit` *faz ajustamento segundo o critério dos* **mínimos quadrados**.

Para o problema do Exemplo 1.7 obtivémos os gráficos da Fig. 3 usando o MATHEMATICA.

• MATLAB

A função `polyfit` *permite fazer o* **ajustamento de dados discretos através de interpolação polinomial e melhor aproximação mínimos quadrados**.

•PYTHON

A função `polyfit`, *faz* **ajustamento de polinómios a dados discretos**. *Está disponível no pacote* `interpolate` *da biblioteca* `scipy` *(*`scipy.interpolate`*).*

1.4 Cálculo de integrais

A solução de muitos problemas requer o cálculo de integrais de funções que não possuem primitiva explícita.

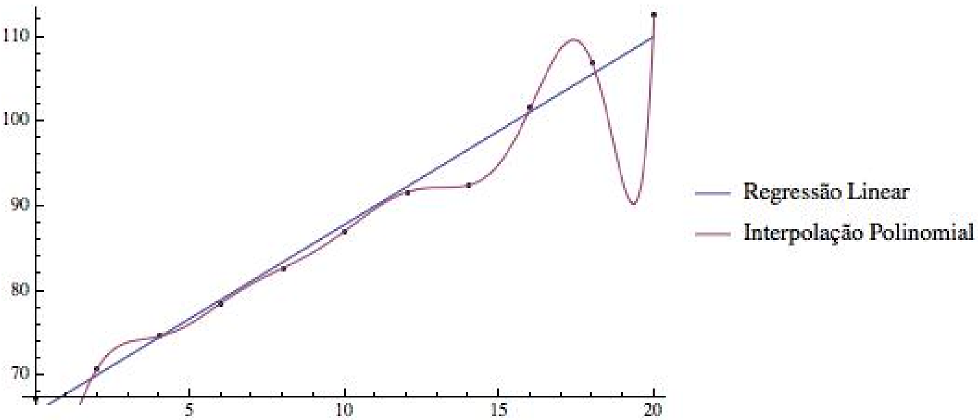


Figura 3: Funções de ajustamento

Exemplo 1.9. Um problema de Probabilidades e Estatística

Consideremos uma população com $M = 200$ *indivíduos. Sabe-se que a distribuição das alturas destes indivíduos pode ser representada por uma Gaussiana que se caracteriza pelo valor médio* $\bar{a} = 1.70$ *m e pelo desvio padrão* $\sigma = 0.10$ *m (distribuição normal):*

$$f(a) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\bar{a}-a)^2/(2\sigma^2)}.$$

Problema: *Qual o número* N *de indivíduos cuja altura varia na faixa* $1.80 - 1.90$ *m?*

Uma estimativa para N *é dada por*

$$N \approx M \int_{1.80}^{1.90} f(a)da = 797.885 \int_{1.80}^{1.90} e^{-50(1.70-a)^2} da.$$

Problema*: Calcular* $$\int_{1.80}^{1.90} e^{-50(1.70-a)^2} da.$$

Integrais deste tipo aparecem nas tabelas de distribuição normal usadas em Probabilidades e Estatística.

Também podemos recorrer ao software já apresentado para resolver este problema de integração numérica.

Exemplo 1.10. *Vejamos então como se pode calcular numericamente o integral.*

• MATHEMATICA

A função `NIntegrate` *faz* **integração numérica**. *Para resolver o problema do Exemplo 1.9 fazemos*

```
In[1]:= 797.885* NIntegrate[Exp[-50 (1.7-a)^2],{a,1.8,1.9}]
Out[1]= 27.181
```

Assim, há 27 pessoas com altura entre $1.80 - 1.90$ *m.*

Note-se que a função `Integrate` *usa apenas as capacidades de cálculo simbólico do* MATHEMATICA, *as quais não permitem obter um valor numérico para o integral em causa.*

• MATLAB

A função `integral` *faz* **integração numérica**. *Para o exemplo em estudo, obtemos:*

```
>> 797.885*integral(@(a)(exp(-50*(1.7-a).^2)),1.8,1.9)
ans = 27.1810
```

A **regra dos trapézios** *está implementada em* `trapz` *e* `cumtrapz`.

• PYTHON

As funções `quad, trapz, simps,...`, *fazem* **integração numérica**. *Estão disponíveis no pacote* `integrate` *da biblioteca* ***scipy*** *(****scipy.integrate****).*

1.5 Resolução de equações diferenciais ordinárias

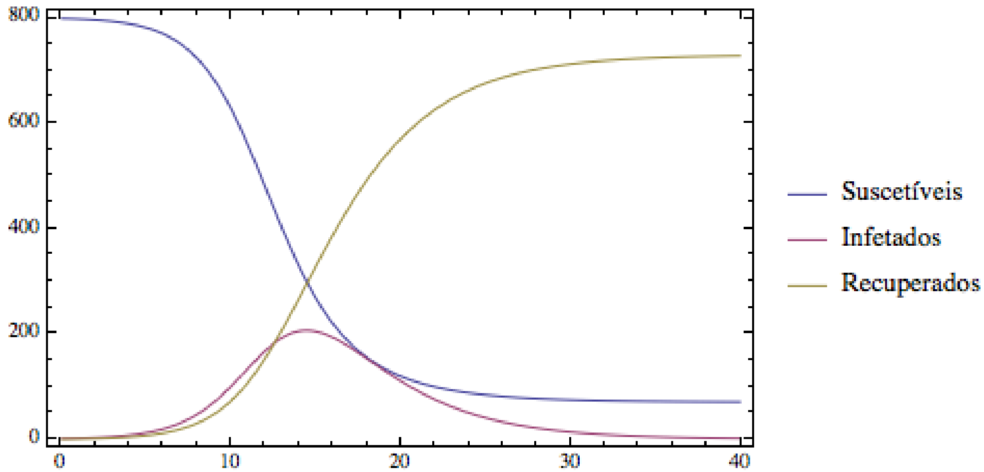
Para terminar apresentamos um modelo matemático (modelo SIR) para epidemias.

Exemplo 1.11. *Um problema de Medicina*

Numa comunidade de 800 crianças suscetíveis de contrair varicela, uma delas é diagnosticada com esta doença. Suponhamos que a propagação da doença é modelada pelo sistema de equações diferenciais

$$\begin{cases} \dfrac{d\,S}{dt} & = -0.001\,S\,I \\ \dfrac{d\,I}{dt} & = 0.001\,S\,I - 0.3\,I \\ \dfrac{d\,R}{dt} & = 0.3\,I \end{cases}$$

onde $S = S(t)$ *é o número de crianças suscetíveis de contrair a doença no momento* t, $I = I(t)$ *é o número de infetados na mesma altura e que podem propagar a doença, e* $R(t)$ *é o número de recuperados, ou seja, que já contraíram a doença e adquiriram imunidade.*



Problema: *Estudar a evolução de* (S, I, R) *desde o início da epidemia até que esta termina.*

Exemplo 1.12. *Os recursos computacionais disponíveis permitem, por exemplo, apresentar a solução de um sistema de equações diferenciais ordinárias através de gráficos.*

- MATHEMATICA
  A função `NDSolve` *faz* **resolução numérica de equações diferenciais ordinárias**.
- MATLAB*: podemos usar* `ode23`, `ode45`, ...
- PYTHON*:* `odeint` *disponível no pacote* `scipy.integrate`.

2 Breves considerações sobre os recursos computacionais disponíveis

O MATLAB, marca registada da MATHWORKS, foi originalmente concebido por Cleve Moler, no final de 1970, e é uma abreviatura de "Matrix Laboratory". É um software interativo de alta performance destinado principalmente ao cálculo numérico. Inicialmente foi desenvolvido com o intuito de resolver sistemas e realizar cálculos com matrizes. Atualmente, integra métodos numéricos, processamento de sinais e construção de gráficos, em ambiente de fácil utilização devido à forma, muito próxima da linguagem matemática, como os problemas e as soluções são expressos. Documentação de apoio variada está disponível neste links:

https://www.mathworks.com/moler/chapters.html

https://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

O GNU Octave é uma linguagem computacional criada por John W. Eaton, sendo tão robusta e poderosa quanto o Matlab, com a vantagem de ser open source. Está disponível neste link:

https://www.gnu.org/software/octave/

O sistema computacional Wolfram Mathematica (conhecido simplesmente como Mathematica) foi criado por Stephen Wolfram no final de 1980, e continuamente desenvolvido pela Wolfram Research. A juntar à sua enorme capacidade de cálculo numérico e simbólico, o Mathematica também serve como um ambiente para desenvolvimento rápido de programas e pode ser usado para a elaboração de documentos com formatação matemática complexa. Destaca-se também pela fácil utilização devido à proximidade da linguagem matemática.

O Python, outra ferramenta de cálculo e visualização, é uma linguagem de programação recente, inventada por Guido van Rossum no final de 1980. Em 2007 e em 2010, foi considerada a linguagem de programação do ano, sendo atualmente a oitava linguagem mais popular, depois de C, Java, Objective-C, C++, C#, PHP e Visual BASIC. Entre as grandes organizações que utilizam o Python incluem-se a Google, a Yahoo!, o YouTube, o CERN e a NASA.

Referências

[1] Carlos Caleiro e Jaime Ramos, Introdução à Programação em Pyhton (IPython Notebooks), 2016 DMIST.

[2] J. Carmo, A. Sernadas, C. Sernadas, F.M. Dionísio e C. Caleiro, Introdução à Programação em Mathematica, 2004 IST Press, 1999, (2ª edição).

[3] D. Goldberg, What every computer scientist should know about floating-point arithmetic, *ACM Comput. Surv.*(1991), **23**, 1, 5–48.

[4] John V. Guttag, Introduction to Computation and Programming Using Python (revised and expanded edition): 2013 MIT Press.

[5] C. Moler, Numerical Computing with Matlab, SIAM, 2004.

[6] A. Quarteroni, R. Sacco e F. Saleri, Cálculo Científico com Matlab e Octave, Springer-Verlag, 2007 (traduzido por Adélia Sequeira).

[7] João Pavão Martins, Programação em Python: Introdução à programação utilizando múltiplos paradigmas, 2015 IST Press.

3 Objetivos e Programa da Unidade Curricular

Através dos algoritmos estudados em Matemática Computacional, dá-se a oportunidade de aplicar os conhecimentos já adquiridos em disciplinas anteriores, como Cálculo Diferencial e Integral, Álgebra Linear e Equações Diferenciais, levando à procura, de modo construtivo, de soluções aproximadas de problemas cujos modelos matemáticos frequentemente não são resolúveis a não ser por métodos numéricos. A Matemática Computacional destina-se a fornecer as bases que permitam usar as potencialidades do computador para resolver problemas numéricos comuns a todas as áreas da Engenharia. Através dos exemplos numéricos que servem de ilustração ao curso, bem como do projecto computacional que é proposto, incentivam-se boas práticas de Programação através da resolução de problemas numéricos concretos.

3.1 Objetivos

A Unidade Curricular de Matemática Computacional tem como objetivo:

- Introduzir os sistemas de ponto flutuante com um breve estudo das suas propriedades e limitações.
- Apresentar conceitos e resultados teóricos como introdução ao estudo de métodos numéricos para resolução de equações, ajustamento de dados discretos e funções, integração e equações diferenciais.
- Analisar os resultados das simulações numéricas com base nas noções de erro, convergência e estabilidade.

3.2 Programa

1. **Conceitos Básicos do Cálculo Científico**

   Erros absoluto e relativo. Representação de números no computador e sistemas de ponto flutuante. Arredondamentos. Teoria linear de erros. Propagação de erros em funções e algoritmos. Condicionamento, estabilidade algorítmica.

2. **Resolução Numérica de Equações Não Lineares**

   Localização de raízes. Método da bisseção. Método da secante. Método de Newton. Métodos de ponto fixo. Análise de convergência (local e global) e estimativas de erro para cada um destes métodos.

3. **Resolução Numérica de Sistemas de equações**

   3.1. **Sistemas Lineares** Normas matriciais. Condicionamento. Métodos iterativos (ex: métodos de Jacobi, de Gauss-Seidel, SOR,...). Análise de convergência e estimativas de erro.

   3.2. **Sistemas Não-Lineares** Método de Newton. Métodos de ponto fixo. Análise de convergência e estimativas de erro.

4. **Ajustamento de Dados Discretos e Aproximação de Funções**

   4.1. **Interpolação Polinomial** Solução através da matriz de Vandermonde. Fórmula interpoladora de Lagrange. Diferenças divididas de Newton. Fórmula interpoladora de Newton. Erro de interpolação na aproximação de funções.

   4.2. **Teoria de Aproximação** Melhor aproximação mínimos quadrados.

5. **Integração Numérica**

   Grau de exatidão de uma regra de quadratura. Método dos coeficientes indeterminados. Fórmulas de Newton-Cotes (ponto médio, trapézio, Simpson). Fórmulas compostas. Quadraturas de Gauss. Análise de erros. Ordem de precisão.

6. **Resolução Numérica de Equações Diferenciais Ordinárias**

   Métodos de passo único (Euler, Taylor de segunda ordem, Runge-Kutta de segunda e quarta ordem). Análise de erros. Consistência, convergência e estabilidade. Aproximação de problemas de valores na fronteira através de diferenças finitas.