Matemática Computacional - MEBiol, MEBiom e MEFT

Responsável: Ana Leonor Silvestre

ana.silvestre@math.tecnico.ulisboa.pt

Instituto Superior Técnico, $1^{\underline{o}}$ *Semestre, 2020/2021*

Considerações gerais sobre a Matemática Computacional

A **Matemática Computacional** é uma área da Matemática cuja importância é evidenciada pelo papel fundamental que hoje em dia as **ferramentas computacionais** desempenham em todos os ramos das Ciências e da 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**.

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 dos 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.

Representação de números e sistemas de ponto flutuante

A implementação em computador de um sistema de representação numérica normalizada obedece a regras definidas pelo Institute of Electrical and Electronics Engineers (IEEE).
Existem dois esquemas principais de **representação de números reais**: *precisão simples* de 32 bits e *precisão dupla* de 64 bits (bit=dígito binário)

$$x = (-1)^s (d_1.d_2...d_n)_2 \times 2^t,\ s \in \{0,1\},\ d_i \in \{0,1\}, d_1 \neq 0$$

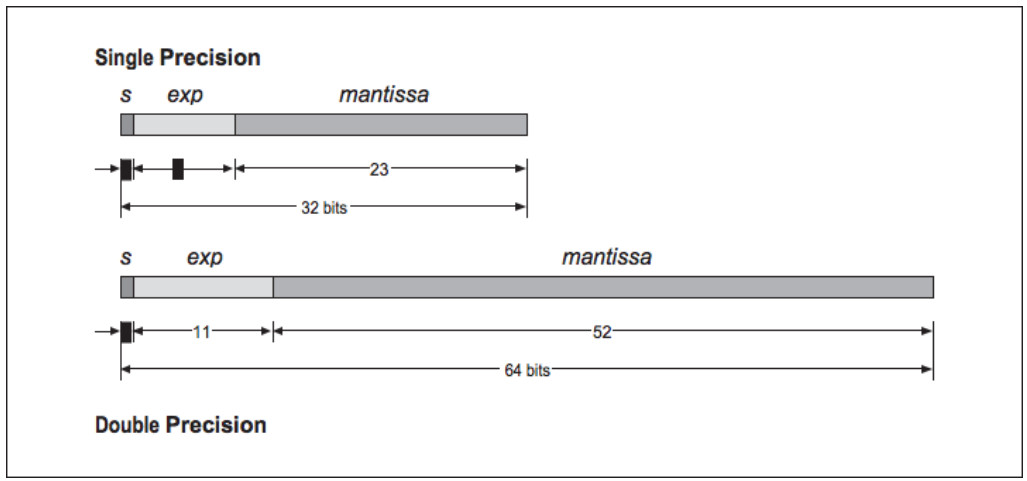


Figura: Sistemas de ponto flutuante definidos pela norma IEEE 754 (a versão mais recente é: IEEE 754-2019)

Representação de números no computador

Na representação

$$x = (-1)^s (d_1.d_2...d_n)_2 \times 2^t,\ d_1 \neq 0,\ s \in \{0, 1\},$$

o inteiro t representa o **expoente**, s o **sinal** do número e a $(d_1.d_2...d_n)_2$ chama-se **mantissa**.

As representações definidas pelas normas IEEE incluem

INF (infinito)

e

NaN (not a number).

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 com a representação de números e cálculos efetuados no computador.

- **1993: Divisão em ponto flutuante no Pentium**

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

- **4 de junho de 1996: Desintegração do Ariane 5 devido a um erro numérico (overflow)**

  O computador de voo do novo foguetão Ariane 5 incorporava, sem que ninguém esperasse, um erro de programação numa rotina aritmética, o qual 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 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).

- **Um exemplo de cálculo em sistemas de ponto flutuante**

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

No Matlab R2015b (IEEE-754 com precisão dupla):

» format long

» $1/10^{20} - 1/(10^{20}+1)$

ans = 0

» $1/((10^{20}+1) * 10^{20})$

ans = 9.999999999999999e-41 $\quad (\approx 10^{-40})$

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

Como se explica este resultado?

- **Um problema da Física: 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.

Localização gráfica das soluções de $v(t) = 1000$

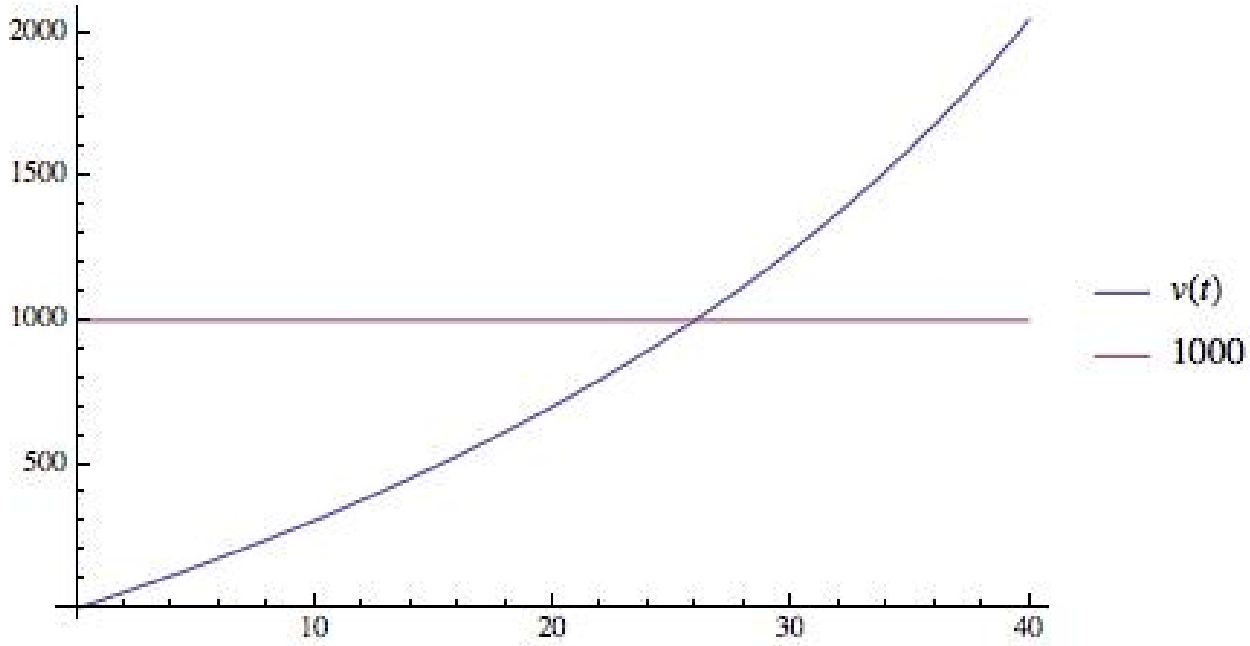


Figura: Localização da solução através de esboço gráfico

Resultado (teórico) que assegura a **existência e unicidade de solução de uma equação** $f(x) = 0$ num certo intervalo limitado:

Teorema 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**.

Aplicação à equação do foguete de Tsiolkovsky

Voltando ao exemplo, consideramos a equação

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

Seja $f(t) := 2200 \ln\left(\frac{16\times 10^3}{16\times 10^3 - 268t}\right) - 9.8t - 1000$. Tem-se

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

$$f'(t) = -9.8 + \frac{589600}{16000 - 268t}$$

$$f'(t) = 0 \iff t = \frac{16000 - \frac{589600}{9.8}}{268} = -164.788$$

Como f' não se anula em $[20, 30]$ e $f(20)f(30) < 0$, conclui-se que a equação tem em $]20, 30[$ uma e uma só solução.

Solução para a equação de Tsiolkovsky obtida com

- 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}
```

Nota: " **FindRoot** searches for a numerical root of f, starting from the point $x = x_0$.
FindRoot uses a damped **Newton's method**, the **secant method**, and *Brent's method*."

Outro software para resolução numérica de equações

- Matlab : a **resolução numérica de equações não lineares** pode fazer-se recorrendo à função `fzero`.

- Python: está disponível `nsolve`.

- **Um problema de Biologia**
  Os dados representados na tabela

Dia	0	2	4	6	8	10	12
Quant. $\times 10^{-6}$	67.38	70.93	74.67	78.60	82.74	87.10	91.69

Dia	14	16	18	20
Quant. $\times 10^{-6}$	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.

Resolução do problema: Construir uma função contínua que "*se ajuste*" aos dados disponíveis. Usar essa função para prever valores que não podem ser obtidos/medidos experimentalmente.

Ajustamento de dados discretos

- Mathematica

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

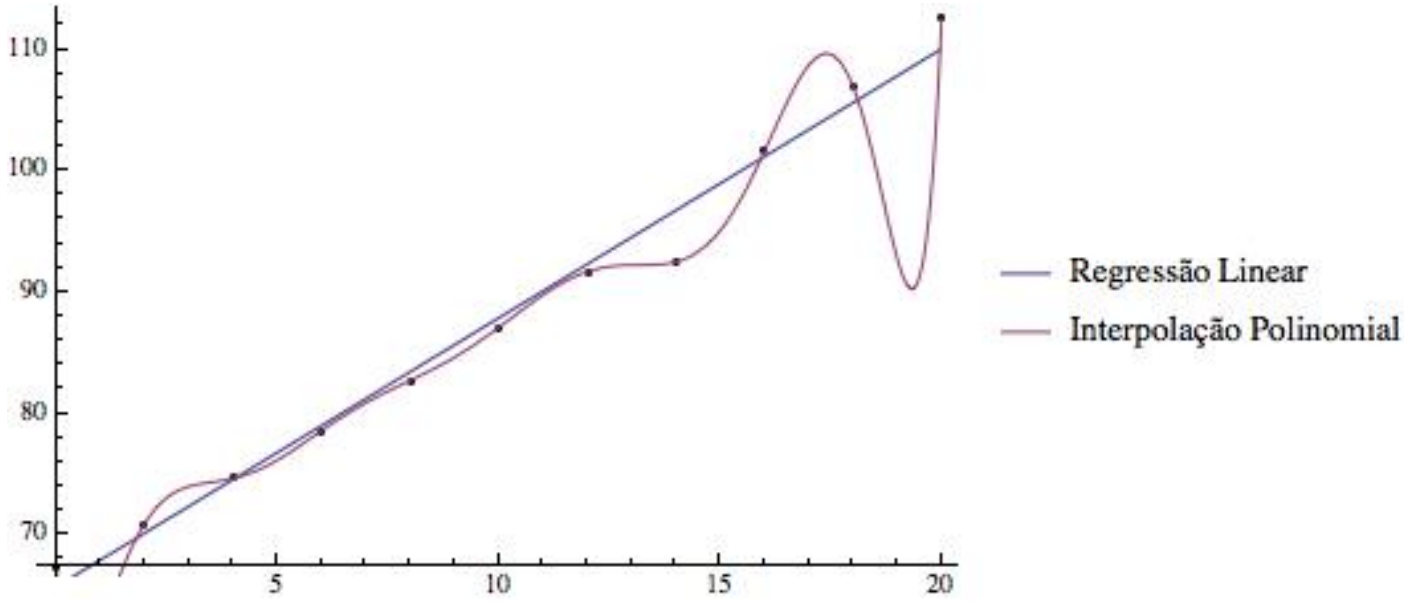


Figura: Funções de ajustamento aos dados experimentais relativos às bactérias

Ajustamento de dados discretos

- 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`).

- **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?

  Resolução do problema: 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 numérico: Calcular

$$797.885 \int_{1.80}^{1.90} e^{-50(1.70-a)^2} da.$$

Nota: integrais deste tipo aparecem nas tabelas de distribuição normal usadas em Probabilidades e Estatística.

- Mathematica

A função **NIntegrate** faz **integração numérica**:

```
In[1]:=797.885*NIntegrate[Exp[-50 (1.70 - a)^2], {a, 1.80, 1.90}]
Out[1]:=27.181
```

27 pessoas com altura entre $1.80 - 1.90$ **m**

Integração numérica

- Matlab

A função `integral` faz **integração numérica**. 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`).

- **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 (modelo SIR)

$$\begin{cases} S'(t) &= -0.001\, S(t)\, I(t) \\ I'(t) &= 0.001\, S(t)\, I(t) - 0.3\, I(t) \\ R'(t) &= 0.3\, I(t) \end{cases}$$

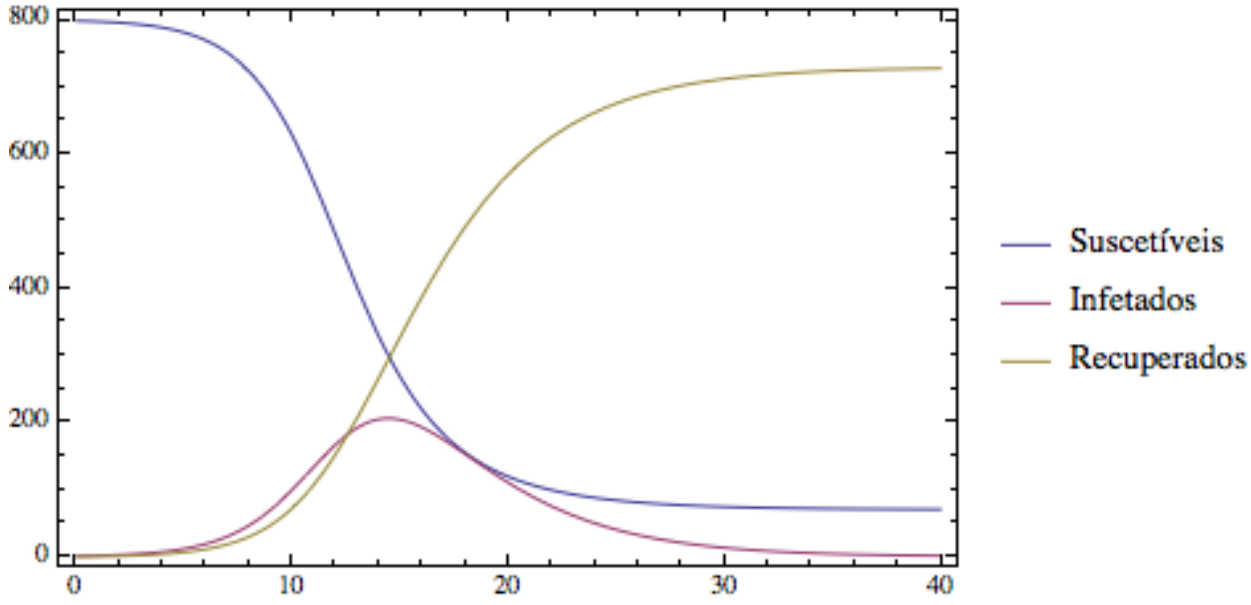
onde

- $S(t)$ é o número de crianças suscetíveis de contrair a doença no momento t,
- $I(t)$ 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.

Resolução numérica de equações diferenciais

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

- Mathematica: a rotina **NDSolve** permite a **resolução numérica de equações diferenciais ordinárias**.



- Matlab: `ode23`, `ode45`, ...
- Python: `odeint` disponível no pacote `scipy.integrate`.

Programa da UC Matemática Computacional

1. **Conceitos Básicos do Cálculo Científico**
   Erros absoluto e relativo. Representação de números no computador e sistemas de ponto flutuante. Arredondamento. Teoria linear de erros. Propagação de erros em funções e algoritmos. Condicionamento e estabilidade.
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**
   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.

Programa (cont.)

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). Análise de erros. Consistência, convergência e estabilidade. Aproximação de problemas de valores na fronteira através de diferenças finitas.