MÉTODOS NUMÉRICOS PARA EQUAÇÕES DIFERENCIAIS PARCIAIS

- Disciplina Obrigatória do Curso MCCT

- Número de Créditos 4, 60 horas

- Prof. Gustavo Benitez Alvarez

- Horário: 4ª feira das 14:00 – 16:00 horas na sala D42
  e 5ª feira das 16:00 – 18:00 horas na sala D42.

Ementa

1. Resolução de Sistemas Lineares e Não-lineares. Autovalores e autovetores.
2. Considerações gerais sobre o método de diferenças finitas aplicado às equações diferenciais parciais.
3. Método dos elementos finitos: malha de elementos finitos, conjunto completo de polinômios de grau k, mapeamento de elementos, elementos isoparamétricos, espaços de elementos finitos, método de Galerkin, integração numérica.
4. Natureza do problema de discretização de domínios contínuos, aproximação das condições de contorno.

Bibliografia

1. **Golub, E., Van Loan, C., Matrix Computations. 3rd Edition John Hopkins. Univ. Press, 1996.**
2. **S. D. Conte, Carl De Boor , Elementary Numerical Analysis: An Algorithmic Approach. Copyright © 1980, 1972, 1965 By Mcgraw-Hill.**
3. **B. P. Demidovich, I. A. Maron, Computational Mathematics. Second Edition, Mir Publishers, 1976.**
4. Ciarlet, P.G., Introduction À L´analyse Numérique Matricielle Et À L´optimisation. Masson, Paris, 1982. English Translation : 1989 (Cambridge University Press, Cambridge).
5. Ciarlet, P. G.; Miara, B.; Thomas, J.-M., Exercices D´ Analyse Numérique Matricielle Et D´optimisation. Masson, Paris, 1986. Second Edition, English Translation : 1989 (Cambridge University Press, Cambridge).
6. **Hughes, T. J. R., Finite Element Method - Linear Static and Dynamic Finite Element Analysis. Prentice-Hall, Englewood Cliffs, 1987, 803 pp., 2000, 682 pp.**
7. **J. N. Reddy , Introduction to the Finite Element Method. McGraw-Hill Science/Engineering/Math; 2 edition 1993, 896 pages.**
8. **O. C. Zienkiewicz, R. L. Taylor, J.Z. Zhu, The Finite Element Method: Its Basis and Fundamentals. Butterworth-Heinemann; 6 edition 2005, 752 pages.**

- Livro PDF **- Livro Impresso** **- Livro Copia Própria**

Método de Avaliação

1. Trabalhos individuais e/ou em grupos para cada uma das quatro unidades com possíveis apresentação em forma de seminários.
2. Duas Provas Escritas.
3. Trabalho final em grupos com apresentação em forma de seminários.
4. Nota de cada avaliação entre 0-10.
5. Nota final calculada como à media.
6. Freqüência mínima de 75%.

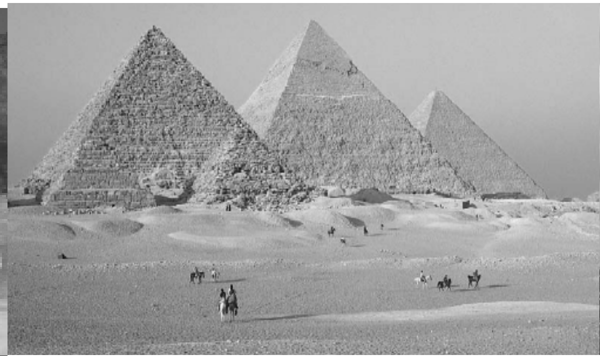
Objetivos Fundamentais

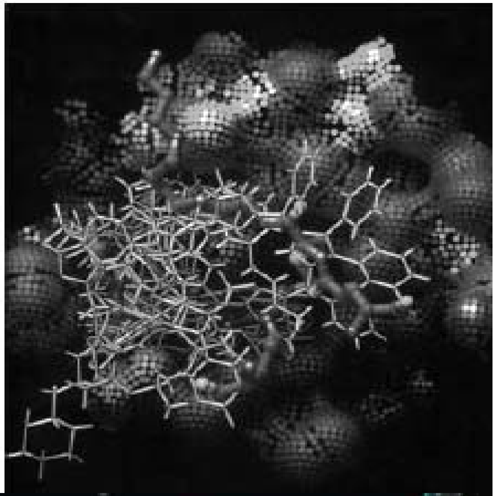
1. Aprender alguns métodos numéricos muito usados na modelagem computacional.
2. Aperfeiçoar no uso de Códigos, Programas e Linguagens de Programação para a implementação destes métodos numéricos em computadores.
3. Aplicação dos objetivos 1 e 2 na resolução, via modelagem computacional, de um problema concreto (Trabalho Final).

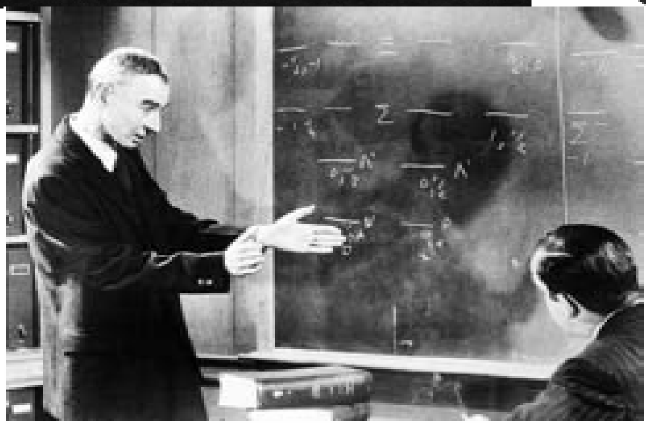
Por que esta ementa?

Problemas de Nosso Cotidiano: Macro e Micro

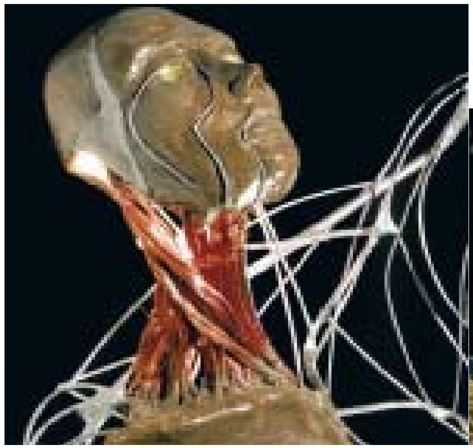
Modelos do Macro e Micro

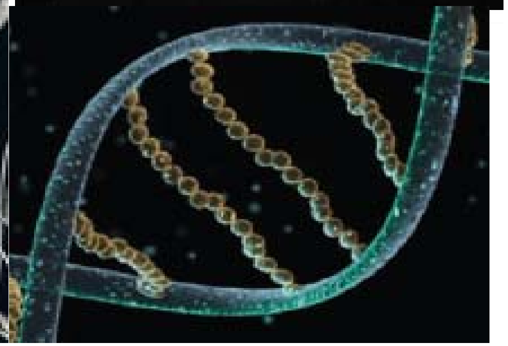






Modelos Matemáticos





Por que esta ementa?

Problemas

Conhecimento Qualitativo

Conhecimento Quantitativo

Modelos Matemáticos

Equações Algébricas

Equações Transcendentais

Equações Diferenciais Ordinárias

Equações Diferenciais Parciais

Outras Formas de Modelar

Resolução de Sistemas Lineares e Não-lineares.

Autovalores e autovetores.

Método de diferenças finitas.

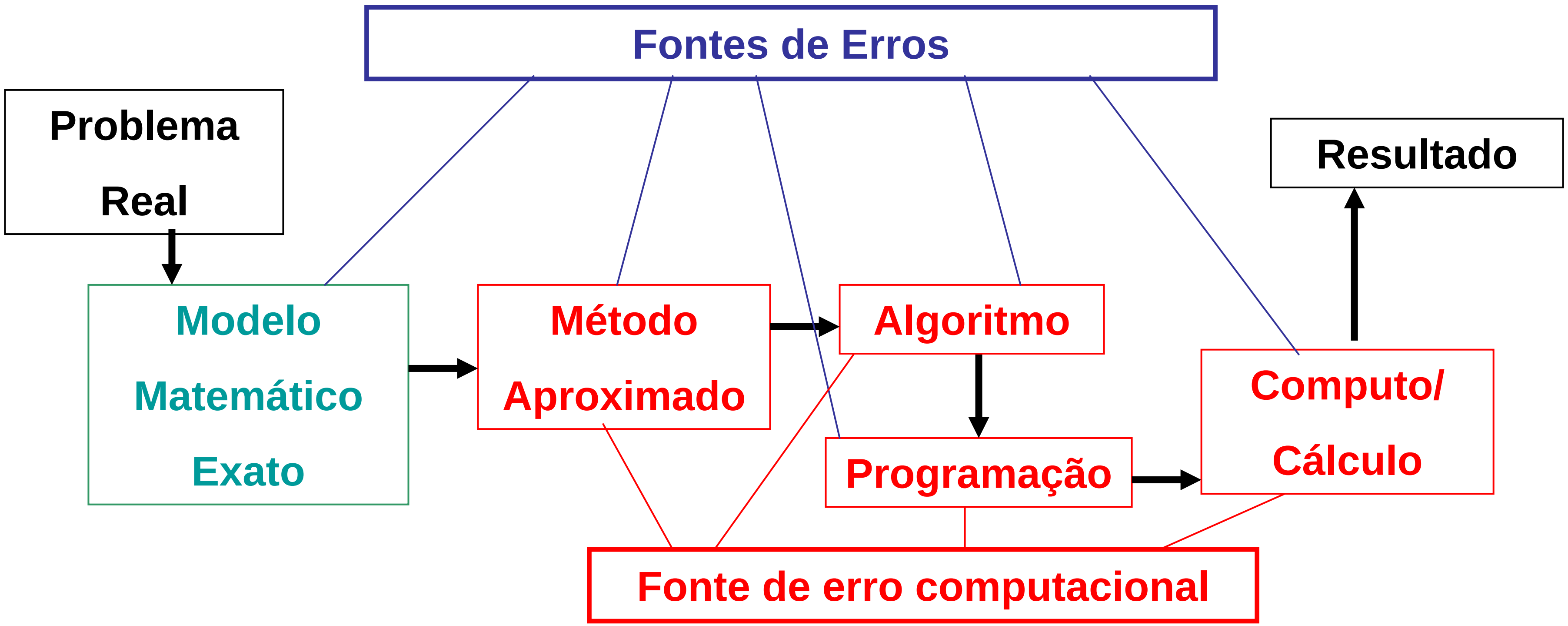
Método dos elementos finitos.

Discretização de domínios contínuos e aproximação das condições de contorno.

Computador

Preliminares: Representação de Números e Erros no Computador

- Objetivo: Estudar a representação de números no computador, erros introduzidos por esta representação, fontes de alguns tipos de erros computacionais e sua propagação.



Fontes de Erros

Modelo Matemático: Introduz simplificações do problema real.

Método Aproximado (Métodos Numéricos): Introduz erro correspondente ao métodos usado para obter uma solução aproximada do modelo matemático.

Algoritmo: Introduz erro se o algoritmo usado precisa de infinitos passos, já que na pratica é impossível realizar infinitos passos.

Programação: Introduz erro na implementação do algoritmo na linguagem escolhida.

Computo/Cálculo: Introduz erro correspondente as operações aritméticas (suma, resta, multiplicação, divisão), já que o computador possui precisão finita.

Fonte de erro computacional

Três Formas de Expressar o Erro

1. Erro Absoluto: Valor absoluto da diferença entre o valor exato da grandeza e seu valor aproximado

$$e_a = \left| X_{exato} - \tilde{X}_{aproximado} \right|$$

2. Erro Relativo:

$$e_r = \frac{e_a}{X_{exato}} = \left| \frac{X_{exato} - \tilde{X}_{aproximado}}{X_{exato}} \right|$$

3. Erro Porcentual:

$$e_p = 100 e_r = 100 \left| \frac{X_{exato} - \tilde{X}_{aproximado}}{X_{exato}} \right|$$

Representação de Números Inteiros (positivos/negativos) na base β

Seja o polinômio $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 x^0$

com coeficientes $a_i \ (i = 0 \ldots n)$ inteiros (positivos/negativos). Se $x = \beta$ é um inteiro positivo (base), então o número (positivos/negativos)

$N = p(\beta) = a_n \beta^n + a_{n-1} \beta^{n-1} + \ldots + a_1 \beta^1 + a_0 \beta^0$ é um inteiro.

Este resultado pode ser usado para representar números inteiros na forma:

$$N = p(\beta) = a_n \beta^n + a_{n-1} \beta^{n-1} + \ldots + a_1 \beta^1 + a_0 \beta^0 = \pm(a_n a_{n-1} \ldots a_1 a_0)_\beta$$

onde β é a base e $0 \leq a_i < \beta \ (i = 0 \ldots n)$.

Exemplos de Representação de Números Inteiros (positivos/negativos) na base β

$$N = p(\beta) = a_n\beta^n + a_{n-1}\beta^{n-1} + \ldots + a_1\beta^1 + a_0\beta^0 = \pm(a_n a_{n-1}\ldots a_1 a_0)_\beta$$

$$0 \le a_i < \beta \ \ (i = 0\ldots n)$$

(base decimal $\beta = 10$)

$$+123 = N = p(10) = \underbrace{(+1)}_{a_2}\times 10^2 + \underbrace{(+2)}_{a_1}\times 10^1 + \underbrace{(+3)}_{a_0}\times 10^0 = +(\underbrace{1}_{a_2}\ \underbrace{2}_{a_1}\ \underbrace{3}_{a_0})_{10}$$

$$-123 = -N = -p(10) = \underbrace{(-1)}_{a_2}\times 10^2 + \underbrace{(-2)}_{a_1}\times 10^1 + \underbrace{(-3)}_{a_0}\times 10^0 = -(\underbrace{1}_{a_2}\ \underbrace{2}_{a_1}\ \underbrace{3}_{a_0})_{10}$$

(base binária $\beta = 2$)

$$+123 = N = p(2) = \underbrace{(+1)}_{a_6}\times 2^6 + \underbrace{(+1)}_{a_5}\times 2^5 + \underbrace{(+1)}_{a_4}\times 2^4 + \underbrace{(+1)}_{a_3}\times 2^3 + \underbrace{(+0)}_{a_2}\times 2^2 + \underbrace{(+1)}_{a_1}\times 2^1 + \underbrace{(+1)}_{a_0}\times 2^0 = +(\underbrace{1}_{a_6}\underbrace{1}_{a_5}\underbrace{1}_{a_4}\underbrace{1}_{a_3}\underbrace{0}_{a_2}\underbrace{1}_{a_1}\underbrace{1}_{a_0})_2$$

$$-123 = -N = -p(2) = \underbrace{(-1)}_{a_6}\times 2^6 + \underbrace{(-1)}_{a_5}\times 2^5 + \underbrace{(-1)}_{a_4}\times 2^4 + \underbrace{(-1)}_{a_3}\times 2^3 + \underbrace{(-0)}_{a_2}\times 2^2 + \underbrace{(-1)}_{a_1}\times 2^1 + \underbrace{(-1)}_{a_0}\times 2^0 = -(\underbrace{1}_{a_6}\underbrace{1}_{a_5}\underbrace{1}_{a_4}\underbrace{1}_{a_3}\underbrace{0}_{a_2}\underbrace{1}_{a_1}\underbrace{1}_{a_0})_2$$

Exemplos de Representação de Números Inteiros (positivos/negativos) na base β

- Podemos representar no computador os números inteiros sem erro!
- Quanto menor é a base maior quantidade de dígitos é necessária para representar o número inteiro.
- Posso armazenar no computador qualquer quantidade de dígitos? Não. O computador funciona com um comprimento finito de palavra (8 bits, 16 bits, 32 bits, 64 bits, etc).
- Exemplo para o FORTRAN existem os seguintes inteiros:

INTEGER*1 representa inteiros entre –128 a 127

INTEGER*2 representa inteiros entre –32 768 a 32 767

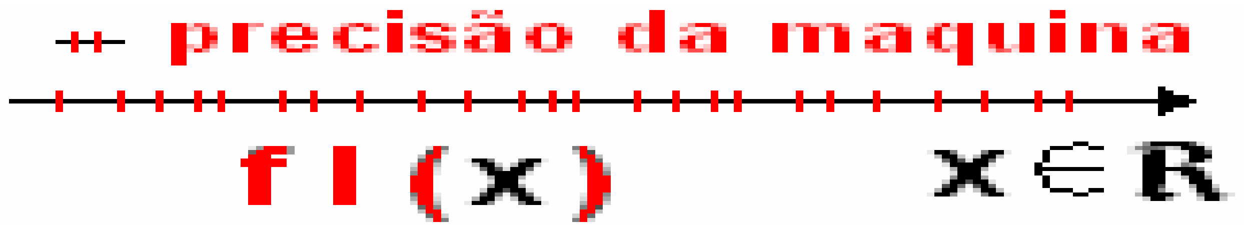
INTEGER*4 –2 147 483 648 a 2 147 483 647

Os números após o * indicam quantos bytes a variável ocupa na memória do computador.

$$1 \text{ bytes}=8 \text{ bit}= \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}$$

Representação de Números Reais (positivos/negativos) na base β

Precisão Finita do Computador: Diferente da Aritmética Exata (Matemática) as operações aritmética (+, -, *, /) no computador (Aritmética com Precisão Finita) são afetadas por um erro (round-off error) porque o hardware apenas pode representar um subconjunto de todos os números reais. Este subconjunto é denotado por F e seus elementos são chamados de “Floating Point Numbers” (fl(x)).



Round-off Error = x – fl(x)

Representação de Números Reais (positivos/negativos) na base β

Um número real *x* pode ser representado no sistema de ponto flutuante com *n* dígitos na base β como:

$$x = \pm(.d_1 d_2 \cdots d_n)_\beta \, \beta^p \qquad -M < p < M$$

onde $(.d_1 d_2 \cdots d_n)_\beta$ é a fração (mantissa) e *p* o expoente.

A precisão ou quantidade *n* de dígitos do número depende do comprimento de palavra do computador.

Exemplo para o FORTRAN existe simples e dupla precisão

REAL*4 ±3.402823E+38 (simples precisão, 6 casas decimais)

REAL*8 ±1.797693134862316D+308 (dupla precisão, 15 casas decimais)

Representação de Números Reais (positivos/negativos) na base β

Para o computador e/ou outros dispositivos eletrônicos existem dois tipos de número:

- Números Exatos que são representados por um número finito de dígitos. Exemplo 2, 15, 3/2, 5.43, etc.

- Números Aproximados que não podem ser representados por um número finito de dígitos. Exemplo

$$\frac{4}{3} = 1.3333333\ldots, \quad \sqrt{2} = 1.414213\ldots$$

$$\pi = 3.141592\ldots$$

$$e = 2.71828182845904523536028747135 27\ldots$$

Representação de Números Reais (positivos/negativos) na base β

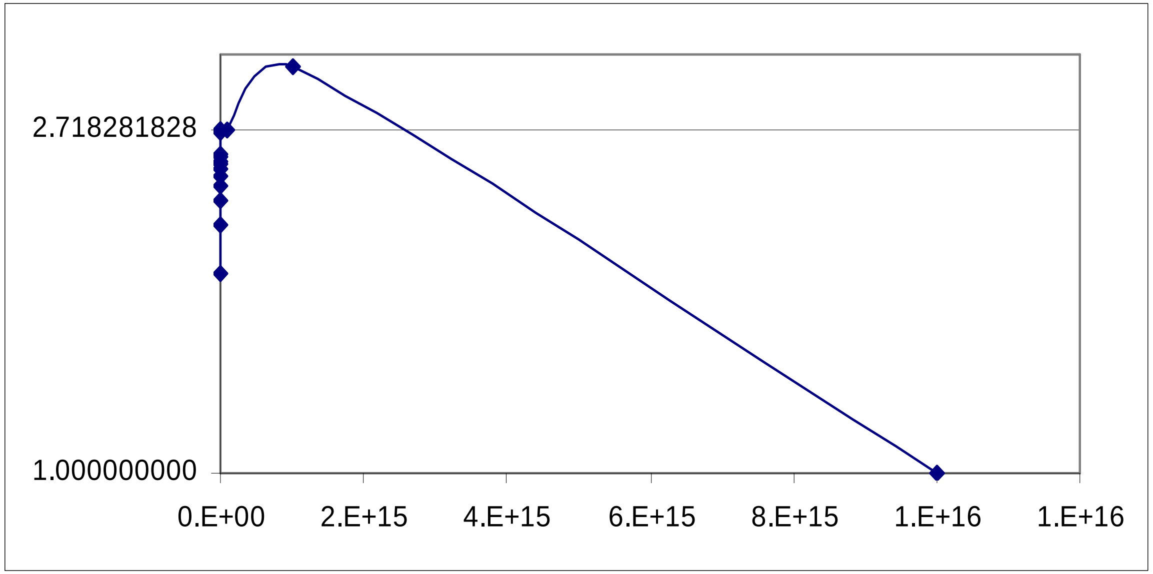
Quando $\begin{cases} |x| \geq \beta^{M} & \rightarrow \text{ overflow} \\ |x| \leq \beta^{-M-n} & \rightarrow \text{ underflow} \end{cases}$ $x = \pm(.d_1 d_2 \cdots d_n)_\beta \beta^{p}$

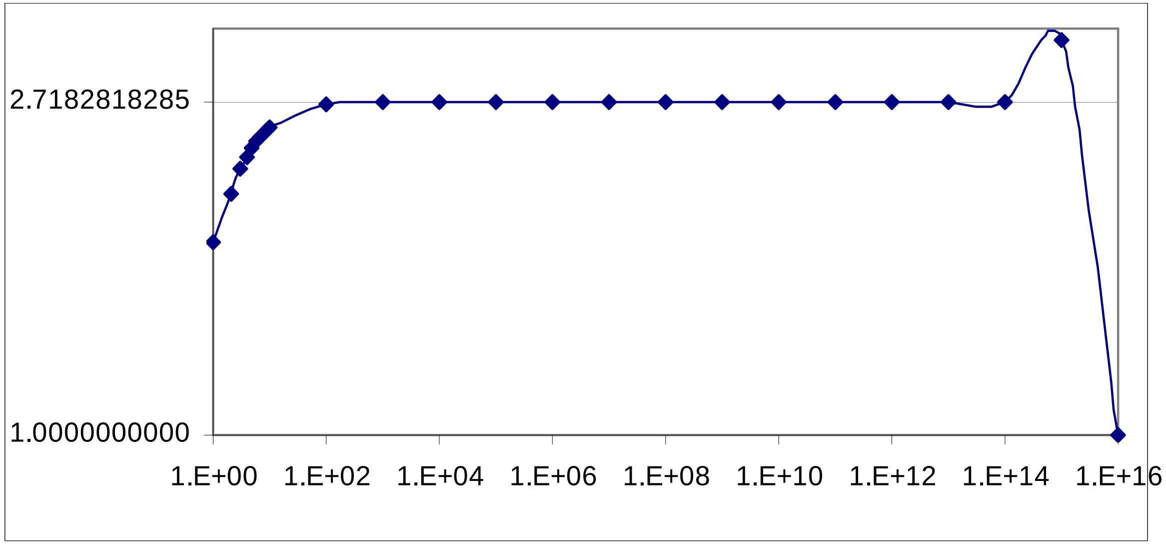
$-M < p < M$

e nestes casos o número não está definido, causando uma parada, ou pode ser representado como um número especial que não obedece as operações aritméticas quando combinado com outro número.

$$\begin{cases} e = 2.71828182845904523536028747135 27\ldots \\ e = \lim_{n \to \infty} \left(1 + \dfrac{1}{n}\right)^{n} \end{cases}$$

$$e = \lim_{n \to \infty}\left(1 + \frac{1}{n}\right)^n = 2.71828182845904523536028747135 27\ldots$$





n	(1+1/n)^n
1	**2.000000000000000E+00**
2	**2.250000000000000E+00**
3	**2.370370370370370E+00**
4	**2.441406250000000E+00**
5	**2.488320000000000E+00**
6	**2.521626371742110E+00**
7	**2.546499697040710E+00**
8	**2.565784513950340E+00**
9	**2.581174791713200E+00**
10	**2.593742460100000E+00**
100	**2.704813829421530E+00**
1000	**2.716923932235520E+00**
10000	**2.718145926824360E+00**
100000	**2.718268237197530E+00**
1000000	**2.718280469156430E+00**
10000000	**2.718281693980370E+00**
100000000	**2.718281786395800E+00**
1.E+09	**2.718282030814510E+00**
1.E+10	**2.718282053234790E+00**
1E+11	**2.718282053357110E+00**
1E+12	**2.718523496037240E+00**
1E+13	**2.716110034086900E+00**
1E+14	**2.716110034087020E+00**
1E+15	**3.035035206549260E+00**
1E+16	**1.000000000000000E+00**

Cálculos feitos com Excel 2003

Propagação de Erros na Aritmética do Computador

Seja a função de duas variáveis $\boldsymbol{y = g(x_1, x_2)}$

e os erros de cada variável denotados por

$\boldsymbol{\Delta x_1, \Delta x_2}$ e $\boldsymbol{\Delta y}$. Logo $\boldsymbol{y + \Delta y = g(x_1 + \Delta x_1, x_2 + \Delta x_2)}$.

A expansão em serie de Taylor desta função é:

$$y + \Delta y = g(x_1, x_2) + \left(\frac{\partial g}{\partial x_1} \Delta x_1 + \frac{\partial g}{\partial x_2} \Delta x_2 \right) + o(\Delta x_1, \Delta x_2)^2$$

Se $\boldsymbol{\Delta x_1}$ e $\boldsymbol{\Delta x_2}$ são suficientemente pequenos tais que potencias maiores podem ser desprezadas temos

$$\Delta y \approx \left(\frac{\partial g}{\partial x_1} \Delta x_1 + \frac{\partial g}{\partial x_2} \Delta x_2 \right)$$

Propagação de Erros na Aritmética do Computador

1. Erro na Soma de Números $X = x_1 + x_2$

$$X + \Delta X = (x_1 + \Delta x_1) + (x_2 + \Delta x_2)$$

Erro Absoluto $\Delta X = \Delta x_1 + \Delta x_2$

Erro Relativo $\dfrac{\Delta X}{X} = \dfrac{\Delta x_1}{X} + \dfrac{\Delta x_2}{X}$

Máximo Erro Relativo $\left|\dfrac{\Delta X}{X}\right| \leq \left|\dfrac{\Delta x_1}{X}\right| + \left|\dfrac{\Delta x_2}{X}\right|$

Propagação de Erros na Aritmética do Computador

2. Erro na Diferença de Números $X = x_1 - x_2$

$$X + \Delta X = (x_1 + \Delta x_1) - (x_2 + \Delta x_2) = (x_1 - x_2) + (\Delta x_1 - \Delta x_2)$$

Erro Absoluto $\Delta X = \Delta x_1 - \Delta x_2$

Erro Relativo $\dfrac{\Delta X}{X} = \dfrac{\Delta x_1}{X} - \dfrac{\Delta x_2}{X}$

Máximo Erro Relativo $\left|\dfrac{\Delta X}{X}\right| \leq \left|\dfrac{\Delta x_1}{X}\right| + \left|\dfrac{\Delta x_2}{X}\right|$

Máximo Erro Absoluto $|\Delta X| \leq |\Delta x_1| + |\Delta x_2|$

Propagação de Erros na Aritmética do Computador

3. Erro no Produto de Números $X = x_1 \bullet x_2$

$$X + \Delta X = (x_1 + \Delta x_1) \bullet (x_2 + \Delta x_2)$$

Máximo Erro Relativo $\left|\frac{\Delta X}{X}\right| \le \left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right|$

Máximo Erro Absoluto $|\Delta X| = \left|\frac{\Delta X}{X}\right| X \le \left(\left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right|\right) X$

Máximo Erro Relativo na Potencia $X = x^k$

$$\left|\frac{\Delta X}{X}\right| \le k \left|\frac{\Delta x}{x}\right|$$

Propagação de Erros na Aritmética do Computador

4. Erro na divisão de Números $X = \dfrac{x_1}{x_2}$

$$X + \Delta X = \frac{(x_1 + \Delta x_1)}{(x_2 + \Delta x_2)}$$

Máximo Erro Relativo $\left|\dfrac{\Delta X}{X}\right| \leq \left|\dfrac{\Delta x_1}{x_1}\right| + \left|\dfrac{\Delta x_2}{x_2}\right|$

Máximo Erro Absoluto $\left|\Delta X\right| = \left|\dfrac{\Delta X}{X}\right| X \leq \left(\left|\dfrac{\Delta x_1}{x_1}\right| + \left|\dfrac{\Delta x_2}{x_2}\right|\right) X$

Frase do Dia

“Attractive mathematics does not protect one from the rigors of digital computation.”

J. H. Wilkinson, von Neumann Lecture,
SIAM - Society for Industrial and Applied Mathematics Meeting, Boston, Fall 1970