# CCI - 22
# Matemática Computacional

Prof. Paulo André

http://www.comp.ita.br/~pauloac

pauloac@ita.br

Sala 110 – Prédio da Computação

# Estrutura do Curso

- Introdução ao estudo de matemática numérica
  - Representação de dados e aritmética de máquina
  - Noções sobre erros, condicionamento e estabilidade numérica
- Resolução de sistemas lineares
- Zero (raízes) de funções reais
- Interpolação
- Ajuste de Curvas
- Integração numérica
- Resolução numérica de equações diferenciais ordinárias

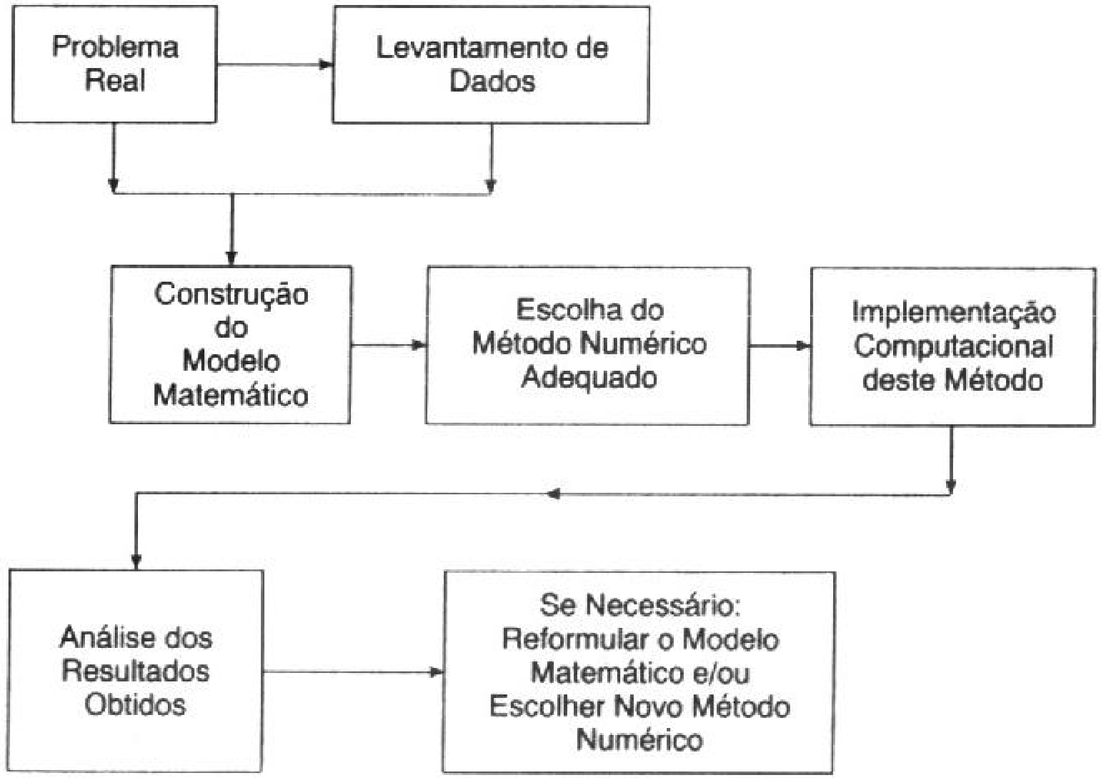
# Estrutura do Curso 2

- Avaliação
  - Por bimestre: 1 prova e 2-3 laboratórios
  - Nota bimestre = 60%* prova + 40%*média labs
  - Um exame final
  - As provas, os laboratórios e o exame são **individuais**

# Bibliografia


- Ruggiero, M.A.C. & Lopes. V. Cálculo Numérico, Aspectos Teóricos e Computacional. McGraw Hill, São Paulo, 1987
- Chapra, S.; Canale, R.; Métodos Númericos para Engenharia. McGraw Hill. 2002
- Cláudio, D.M. & Marins J.M. Cálculo Numérico Computacional. Atlas. São Paulo 1994 2 ed.
- Notas de Aula.

# Método de Solução em Matemática Computacional



# Exemplo

- Calcule a área de uma circunferência de raio 100 m.
- RESULTADOS OBTIDOS

    a) $A=31400\ m^2$

    b) $A = 31416\ m^2$

    c) $A = 31415.92654\ m^2$

- Como justificar as diferenças entre os resultados? É possível obter "exatamente" esta área?

# Exemplo 2

$$S = \sum_{i=1}^{30000} x_i \qquad \text{para } x_i = 0.5 \text{ e para } x_i = 0.11$$

**RESULTADOS OBTIDOS**

| | | | |
|---|---|---|---|
| *i*) | para $x_i = 0.5$: | na calculadora: | $S = 15000$ |
| | | no computador: | $S = 15000$ |
| *ii*) | para $x_i = 0.11$: | na calculadora: | $S = 3300$ |
| | | no computador: | $S = 3299.99691$ |

# Sistemas de Numeração

| Sistema | Base | Alfabeto |
|---|---|---|
| Decimal | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| Binário | 2 | 0, 1, |
| Octal | 8 | 0, 1, 2, 3, 4, 5, 6, 7, |
| Hexadecimal | 16 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,A,B,C,D,E,F |

- Equivalência entre os sistemas:

(29)base 10 = (11101)base 2 = (35)base 8 = (1D)base 16

# Representação de Números

- Um número em base B, pode ser representado como $(a_j, a_{j-1}, a_{j-2}, ... a_0)_B$ , onde $0 \leq a_k \leq B-1$ e escrito na forma polinomial:
  – $a_j B^j + a_{j-1} B^{j-1} + a_{j-2} B^{j-2} + ....... + a_1 B^1 + a_0 B^0$
- Logo:
- $(347)_{10} = 3 \times l0^2 + 4 \times l0^1 + 7 \times 10^0$
- $(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

# Mudança para base 10

- Podemos facilmente converter um número representado no sistema binário para o sistema decimal.
- Por exemplo:
- $(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

- Colocando agora o número 2 em evidência teremos:
- $(10111)_2 = 2 \times (1 + 2 \times (1 + 2 \times (0 + 2 \times 1))) + 1 = (23)_{10}$

- Assim podemos criar um algoritmo para converter um número de base 2 para 10 sem uso de operação de potência.

# Método para mudança de base.

- Base 2 para base 10

  $b_j = a_j$

  $b_{j-1} = a_{j-1} + 2 \times b_j$

  $b_{j-2} = a_{j-2} + 2 \times b_{j-1}$

  .......

  $b_0 = a_0 + 2 \times b_1$

- O número $(a_j, a_{j-1}, a_{j-2}, \ldots a_0)_2$ em base 10 está calculado em $b_{0.}$

# Exemplo – Método de mudança de Base

- Para $(10111)_2$

  $b_4 = a_4 = 1$

  $b_3 = a_3 + 2 \times b_4 = 0 + 2x1 = 2$

  $b_2 = a_2 + 2 \times b_3 = 1+2x2=5$

  $b_1 = a_1 + 2 \times b_2 = 1+2x5 = 11$

  $b_0 = a_0 + 2 \times b_1 = 1 + 2x11 = (23)_{10}$

- O número $(a_j, a_{j-1}, a_{j-2}, \ldots a_0)_2$ em base 10 está calculado em $b_0$.

# Mudança de Base 10 para base 2

- Seja $(a_j, a_{j-1}, a_{j-2}, \ldots a_0)_2$ a representação em base 2 do número $(347)_{10}$. Então:

$(347)_{10} = a_j 2^j + a_{j-1} 2^{j-1} + \ldots.. + a_1 2^1 + a_0 =$

- Colocando em evidência:

$= 2 \times (a_j 2^{j-1} + a_{j-1} 2^{j-2} + \ldots.. + a_1 2^0) + a_0$

$= 2 \times 173 + 1$

- Logo, $a_o = 1$
- Repetindo o processo para 173, encontramos $a_1$

# Método para mudança para base 2

$$
\begin{aligned}
N_0 = 347 &= 2 \times 173 + 1 \Rightarrow a_0 = 1 \\
N_1 = 173 &= 2 \times 86 + 1 \Rightarrow a_1 = 1 \\
N_2 = 86 &= 2 \times 43 + 0 \Rightarrow a_2 = 0 \\
N_3 = 43 &= 2 \times 21 + 1 \Rightarrow a_3 = 1 \\
N_4 = 21 &= 2 \times 10 + 1 \Rightarrow a_4 = 1 \\
N_5 = 10 &= 2 \times 5 + 0 \Rightarrow a_5 = 0 \\
N_6 = 5 &= 2 \times 2 + 1 \Rightarrow a_6 = 1
\end{aligned}
$$

# Algoritmo para Mudança para base 2

- Passo 1:
  - K=0; $N_k = N$
- Passo 2:
  - Encontrar $q_k$ e $r_k$ tais que $N_k = 2 \times q_k + r_k$
  - $a_k = r_k$
- Passo 3:
  - Se $q_k = 0$ Então parar
  - Senão $N_{k+1} = q_k$
  - Faça k=k+1, voltar passo 2

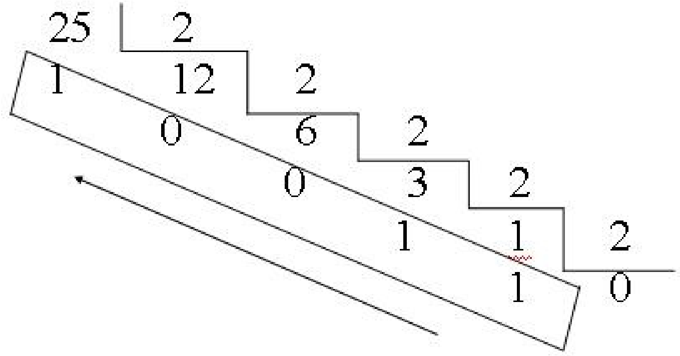
# Conversão de Base

$$(11100101)_2 = 2^7 + 2^6 + 2^5 + 2^2 + 2^0 = (229)_{10}$$

$$(271)_8 = 2*8^2 + 7*8^1 + 1*8^0 = (185)_{10}$$

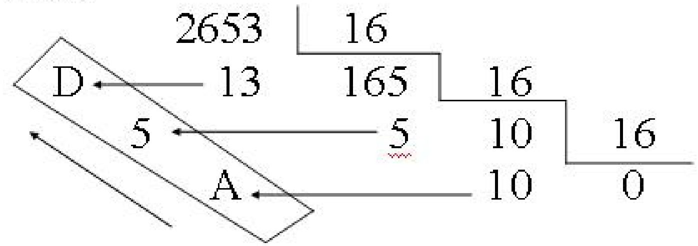
$$(2AB3)_{16} = 2*16^3 + 10*16^2 + 11*16^1 + 3*16^0 = (10931)_{10}$$

- $(25)_{10}$ para a base 2:




então $(25)_{10} = (11001)_2$

- $(2653)_{10}$ para a base 16:




então $(2653)_{10} = (A5D)_{16}$

# Computadores e Sistema Binário

- Números Negativos (dois zeros)
  - Complemento de 2

- Números Fracionários (ponto flutuante)
  - Mantissa e expoente

# Complemento de 2

| N.o | Comp_2 | N.o | Comp_2 |
|---|---|---|---|
| +15 | 01111 | -1 | 11111 |
| +14 | 01110 | -2 | 11110 |
| +13 | 01101 | -3 | 11101 |
| +12 | 01100 | -4 | 11100 |
| +11 | 01011 | -5 | 11011 |
| +10 | 01010 | -6 | 11010 |
| +9 | 01001 | -7 | 11001 |
| +8 | 01000 | -8 | 11000 |
| +7 | 00111 | -9 | 10111 |
| +6 | 00110 | -10 | 10110 |
| +5 | 00101 | -11 | 10101 |
| +4 | 00100 | -12 | 10100 |
| +3 | 00011 | -13 | 10011 |
| +2 | 00010 | -14 | 10010 |
| +1 | 00001 | -15 | 10001 |
| 0 | 00000 | -16 | 10000 |

Conversão de Base:

- Se Positivo (isto é, bit de sinal igual a zero): Conversão Normal
- Se Negativo: Complementa número sem sinal, adiciona 1 e faz a Conversão Normal

# Deslocamento

Deslocamento em base 10 e em base 2

**Base 10**

| | |
|---|---|
| 15000,00 | |
| 1500,0 | |
| 150,0 | x 10 |
| 15,0 | |
| 1,5 | |
| 0,15 | |
| 0,015 | ÷ 10 |
| 0,0015 | |
| 0,00015 | |

**Base 2**

| Base 2 | | Base 10 |
|---|---|---|
| 10000,0 | | 16 |
| 1000,0 | | 8 |
| 100,0 | x 2 | 4 |
| 10,0 | | 2 |
| 1,0 | | 1 |
| 0,1 | | 0,5 |
| 0,01 | ÷ 2 | 0,25 |
| 0,001 | | 0,125 |
| 0,0001 | | 0,0625 |

# Números Fracionários (Ponto Flutuante)

- Inteiros
  - $(15)_{10} = (01111)_2 = 2^3+2^2+2^1+2^0 = 8+4+2+1$
- Números binários fracionários
  - $(1,5)_{10} = (1,1)_2 = 2^0+2^{-1} = 1+0,5 = (1,5)_{10}$
  - $(0,101)_2 = 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = 0,5+0+0,125 = (0,625)_{10}$

# Conversão de Números Fracionários

- De binário para Decimal:
  - $(0.1011)_2 = 2^{-1}+2^{-3}+2^{-4} = 0,5+0,125+0,0625 =$
    $= (0,6875)_{10}$
  - $(0.1001)_2 = 2^{-1}+2^{-4} = 0,5+0,0625 = (0,5625)^{10}$

  - $(111.101) = 2^2 + 2^1+2^0 + 2^{-1} + 2^{-3} = 4+2+1 + 0,5+0,125= (7,625)_{10}$
- Conversão de Fracionários para base 2

  2,25 -> Parte Inteira: $(2)_{10}= (10)_2$

  -> Parte Fracionária $(0,25)_{10} = (0.01)_2$

  Número: $(2,25)_{10} = (10,01)_2$

# Conversão de Fracionários

Dado um número entre 0 e 1 no sistema decimal, como obter sua representação binária?

Considerando o número $r = 0.125$, existem digitos binários: $d_1, d_2, \ldots, d_j, \ldots$, tais que $(0.\, d_1 d_2 \ldots d_j \ldots)_2$ será sua representação na base 2.

Assim,

$$(0.125)_{10} = d_1 \times 2^{-1} + d_2 \times 2^{-2} + \ldots + d_j \times 2^{-j} + \ldots$$

Multiplicando cada termo da expressão acima por 2 teremos:

$$2 \times 0.125 = 0.250 = 0 + 0.25 = d_1 + d_2 \times 2^{-1} + d_3 \times 2^{-2} + \ldots + d_j \times 2^{-j+1} + \ldots$$

# Conversão de Fracionários - 2

e, portanto, $d_1$ representa a parte inteira de $2 \times 0.125$ que é igual a zero e $d_2 \times 2^{-1} + d_3 \times 2^{-2} + \ldots + d_j \times 2^{-j+1} + \ldots$ representa a parte fracionária de $2 \times 0.125$ que é 0.250.

Aplicando agora o mesmo procedimento para 0.250, teremos:

$$\begin{aligned} 0.250 &= d_2 \times 2^{-1} + d_3 \times 2^{-2} + \ldots + d_j \times 2^{-j+1} + \ldots \Rightarrow 2 \times 0.250 = 0.5 = \\ &= d_2 + d_3 \times 2^{-1} + d_4 \times 2^{-2} + \ldots + d_j \times 2^{-j+2} + \ldots \Rightarrow d_2 = 0 \end{aligned}$$

e repetindo o processo para 0.5:

$$\begin{aligned} 0.5 &= d_3 \times 2^{-1} + d_4 \times 2^{-2} + \ldots + d_j \times 2^{-j+2} + \ldots \Rightarrow \\ & 2 \times 0.5 = 1.0 = d_3 + d_4 \times 2^{-1} + \ldots + d_j \times 2^{-j+3} + \ldots \Rightarrow d_3 = 1 \end{aligned}$$

# Algoritmo para Conversão de Base

- A representação em base 10 $(0,d_1\ d_2\ ..d_{j-1}\ d_j)$ pode ser obtida através dos seguintes passos:

- Observe que o algoritmo pode nunca parar...
- Como solucionar o problema do possível loop infinito?

Passo 0: $r_1 = r$; $k = 1$

Passo 1: Calcule $2r_k$.
Se $2r_k \geq 1$, faça: $d_k = 1$,
caso contrário, faça: $d_k = 0$

Passo 2: Faça $r_{k+1} = 2r_k - d_k$.
Se $r_{k+1} = 0$, pare.
Caso contrário:

Passo 3: $k = k + 1$.
Volte ao passo 1.

# Conversão de Base

**Exemplo 1:**

$(0{,}625)_{10}$ para base 2:

0,625 * 2 = 1,25 -> 0,25

1    x 2

0,50 - > 0,5

0    x 2

1,0

1

Então $(0{,}625)_{10} = (0{,}101)_2$

**Exemplo 2:**

$(0{,}3)_{10}$ para base 2:

0,3 * 2 = 0,6 -> 0,6

0    x 2

1,2 -> 0,2

1    x 2

0,4 -> 0,4

0    x 2

0,8 - > 0,8

0    x 2

1,6 -> 0,6

1

Então

$(0{,}3)_{10} = (0{,}0\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001....\ )_2$

# Ponto Flutuante em Computadores

- $0.5625 = (0.1001)_2 = (0.1001)_2 * 2^0$

  característica = 0 + 128 = 128 = $(10000000)_2$

  mantissa = 10010000000000000000000

| | | |
|---|---|---|
| 0 | 10010000000000000000000 | 10000000 |

| Sinal<br>1 bit | Mantissa<br>23 bits | Expoente<br>8 bits |
|---|---|---|

- $2.3 = (10.0\ 1001\ 1001\ 1001\ \ldots.)_2 * 2^0$

  É necessário **normalizar** (colocar o primeiro bit 1 da mantissa à direita do ponto, o mais próximo dele possível.

  $2.3 = (0.100\ 1001\ 1001\ 1001\ \ldots) * 2^2$

  característica = 2 + 128 = 130 = $(10000010)_2$

  mantissa = 100 1001 1001 1001 1001 1001

| | | |
|---|---|---|
| 0 | 100 1001 1001 1001 1001 1001 | 10000010 |