Computação Científica

SEMANA 1

O método "divisão e média", um antigo método para a aproximação da raiz quadrada de um número positivo a, pode ser formulado por:

$$x_{i+1} = \frac{x_i + a/x_i}{2}$$

Com $i = 0, 1, 2, \ldots, n$ e condição de parada que garanta 4 algarismos significativos, escreva um script Scilab para implementar este método, onde:

Considere o valor inicial estimado para a raiz de a igual a 1.

Exercício 1

Exercício 1

```
xn = 0, x = 1, erro=100; n_sig = 4
es = 0.5*10^(2-n_sig);      // condição de parada relativa %
a = input("Entre com um numero real positivo:")
while (erro >= es)do
   xn = (x + a/x)/2
   erro = abs((xn - x)/xn)*100
   x = xn
   printf(" e = %.6f \n",erro)
end
printf(" Valor exato = %.6f\nValor calculado = %.6f ",sqrt(a),xn);
```

Exercício 2

Para computadores, o épsilon da máquina pode ser pensado como o menor número que quando adicionado a 1 resulta um número maior que 1. Um algoritmo baseado nesta ideia pode ser desenvolvido da seguinte maneira:

- **Passo 1:** Faça $\varepsilon = 1$;
- **Passo 2:** Se $(1 + \varepsilon) \leq 1$, então vá para o passo 5
- **Passo 3:** Faça $\varepsilon = \varepsilon / 2$
- **Passo 4:** Retorne ao passo 2
- **Passo 5:** Imprima $2 * \varepsilon$

Escreva um script Scilab para determinar o épsilon da máquina e valide o resultado comparando-o valor calculado com o valor da constante %eps.

Exercício 2

```
e = 1;
while (1+e)> 1 do
    e = e/2;
end
disp('epsilon =',2*e);
```

- A série infinita

$$f(n) = \sum_{i=1}^{n} \frac{1}{i^4}$$

converge para o valor de f(n) = $\pi^4/90$ quando n se aproxima do infinito.

- Escreva um programa em C em precisão simples para calcular f(n) para n = 10000 computando a soma de i = 1 a 10000 usando incrementos de 1
- Repita o cálculo mas na ordem inversa, isto é de i = 10000 a 1, usando incrementos de -1.
- Em cada caso estime o erro relativo percentual real. Explique os resultados.

Resposta: Crescente – f(n) = 1,082322, ε_t = 0,000103 %

Decrescente – f(n) = 1,082323, ε_t = 0,000004 %

Exercício 3

Exercício 3

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
    float i, fn = 0;
    double const real = pow(M_PI,4)/90;
    double erro;
    for(i=1;i<=10000;i++)
    {
        fn = fn + (1/(i*i*i*i));
    }
    printf("real = %.11f\n",real);
    erro = (real - fn)*100/real;
    printf("Crescente: fn = %f com erro de %f\n",fn,erro);
    fn = 0;
    for(i=10000;i>=1;i--)
    {
        fn = fn + (1/(i*i*i*i));
    }
    erro = (real - fn)*100/real;
    printf("Decrescente: fn = %f com erro de %f",fn,erro);
    return 0;
}
```

Exercício 4

Seja o seguinte padrão de representação de um número flutuante de um computador de 16 bits:

Bit de sinal (S) : 1 bit

- S = 0, número positivo
- S = 1, número negativo

Expoente: 4 bits, em excesso de 7:

- expoente = expoente real + 7
- Para representar números utiliza-se o expoente entre 1 e 14. resultando em expoentes reais de -6 a 7

Mantissa: 12 bits (11 armazenados explicitamente)

Representação: $(-1)^{sinal} \times 1.mantissa \times 2^{expoente}$

Exercício 4

a) Menor número:

$$1{,}00000000000 \times 2^{-6} = 0{,}015625$$

b) Maior número

$$1{,}11111111111 \times 2^{7} \cong 2^{8} \cong 256$$

ou $$1{,}11111111111 \times 2^{7} = (2 - \varepsilon) \times 2^{7} = 255{,}9375$$

c) Épsilon

$$\varepsilon = 2^{-11} \cong 4{,}883 \times 10^{-4}$$

d) Menor número sub normal

$$\varepsilon \times 2^{-6} = 2^{-17} \cong 7{,}629 \times 10^{-6}$$

e) Número de algarismos significativos

$$n = 12\log(2) = 3{,}6$$

Logo n = 3 a 4 algarismos significativos de precisão

Bibliografia e crédito das figuras

CHAPRA, Steven. **Applied numerical methods with MATHLAB for engineers and scientists.** McGrawHill, 2012.

CHAPRA, Steven e CANALE, Raymond. **Numerical methods for engineers.** McGrawHill, 2010.

SCARBOROUGH, James. **Numerical Mathematical Analysis**. London: Oxford Press, 1930.