Rio de Janeiro, 25 de Outubro de 2019



Computação Concorrente – Profª Silvana Rossetto

**Relatório sobre a implementação de uma solução concorrente para Integração Numérica**

Universidade Federal do Rio de Janeiro

Departamento de Ciência da Computação

Instituto de Matemática

**Aluno:**

Paulo Renato Mendes (DRE: 117207703)

**1 – Modularização da solução**

Para implementação a solução resolvi utilizar uma abordagem mais simplificada, colocando a apenas a funções matemáticas em uma biblioteca separada. A única estrutura de dados criada foi a **tPonto** que guarda dois *floats*, representando as coordenadas X e Y de um ponto, respectivamente, a biblioteca também contém implementadas as funções de teste do enunciado (de A a G).

A função **“area”** é usada para calcular a área dos retângulos entre a função e o eixo x, é feito um tratamento para o resultado retornado ser sempre positivo.

A função **“integracao\_numerica”** é a função que executa o objetivo principal do projeto, que é calcular e soma as áreas dos retângulos. Ela recebe como argumentos: Os valores a e b do intervalo de integração, o erro aceitável e um caractere para identificar a função. A lógica é a seguinte:

1. Atribui os valores do intervalo às coordenadas X dos pontos de início e fim do intervalo a ser calculado.
2. Executa um switch-case para verificar qual das funções do exemplo vai ser usada, calculando os respectivos valores de Y.
3. Calcula o ponto médio usando sua fórmula.
4. Calcula a área do retângulo maior e a soma das áreas dos menores usando o ponto médio.
5. Calcula a diferença e compara com o erro.
6. Se o erro for aceitável, a função retorna a área do triângulo maior, senão, ela divide mais uma vez os retângulos, numa chamada recursiva. Isso se repete até o erro ser aceitável.

**2 – Solução sequencial**

Tendo a biblioteca a implementação sequencial é bem simples, passo os argumentos recebidos para a função **“integracao\_numerica”** e recebo o valor numa variável local da *main*.

**3 – Solução Concorrente**

Para a implementação concorrente resolvi utilizar um paralelismo de dados. Só a integração em si foi paralelizada. Ao invés de uma *thread* calcular o intervalo inteiro, eu dividi o intervalo de integração pelo número de *threads*, e cada uma irá calcular o valor da integral no seu próprio bloco. Para conseguir esse efeito, todas as variáveis passadas como argumento e o valor total da integral se tornaram globais. A função concorrente da integração usa a função **“integracao\_numerica”.** A lógica de cada *thread* é a seguinte:

1. Recebe o identificador.
2. Calcula o início e fim do intervalo de integração usando o identificador e o tamanho dos blocos.
3. Chama a função de integração e armazena o valor retornado em uma variável local temporária.
4. Usa o *lock* para entrar na seção crítica.
5. Soma o valor da variável temporária ao valor da integral global.
6. Sai da seção crítica e encerra.

**4 - Testes**

Os resultados foram conferidos usando o site *Wolfram-Alpha,* com6 casas decimais de erro:

**Teste A (f(x) = 1 + x) (Intervalo: -20 a 40)**

Resultado Seq.: 660.054932 Tempo: 0.01562

Resultado Conc. 2 threads: 660.054932 Tempo: 0.01561 Ganho: 1.00064

Resultado Conc. 4 threads: 660.054932 Tempo: 0.01562 Ganho: 1.00000

Resultado Conc. 8 threads: 660.054932 Tempo: 0.06255 Ganho: 0.24972

**Teste B (f(x) = sqrt(1+x^2) entre -1 e 1) (Intervalo: -0.8 a 0.8)**

Resultado Seq.: 1.757170 Tempo: 0.01563

Resultado Conc. 2 threads: 1.757170 Tempo: 0.01562 Ganho: 1.00064

Resultado Conc. 4 threads: 1.757170 Tempo: 0.01562 Ganho: 1.00064

Resultado Conc. 8 threads: 1.757170 Tempo: 0.01562 Ganho: 1.00064

**Teste C (f(x) = sqrt(1+x^4)) (Intervalo: -40 a 20)**

Resultado Seq.: 24002.306641 Tempo: 0.06249

Resultado Conc. 2 threads: 24002.306641 Tempo: 0.06249 Ganho: 1.00000

Resultado Conc. 4 threads: 24002.306641 Tempo: 0.03125 Ganho: 1.99968

Resultado Conc. 8 threads: 24002.306641 Tempo: 0.04686 Ganho: 1.333546

**Teste D (f(x) = sin(x^2)) (Intervalo: -40 a 20)**

Resultado Seq.: 1.273945 Tempo: 0.09374

Resultado Conc. 2 threads: 1.273945 Tempo: 0.06248 Ganho: 1.500320

Resultado Conc. 4 threads: 1.273945 Tempo: 0.04297 Ganho: 2.181521

Resultado Conc. 8 threads: 1.273945 Tempo: 0.03999 Ganho: 2.344086

**Teste E (f(x) = cos(exp(-x)\*x) (Intervalo: -10 a 20)**

Resultado Seq.: 19.455467 Tempo: 0.10937

Resultado Conc. 2 threads: 19.455467 Tempo: 0.10933 Ganho: 1.000365

Resultado Conc. 4 threads: 19.455467 Tempo: 0.09376 Ganho: 1.166488

Resultado Conc. 8 threads: 19.455467 Tempo: 0.09374 Ganho: 1.166737

**Teste F (f(x) = cos(exp(-x)) (Intervalo: -10 a 20)**

Resultado Seq.: 200.240112 Tempo: 0.31248

Resultado Conc. 2 threads: 200.240112 Tempo: 0.31247 Ganho: 1.000032

Resultado Conc. 4 threads: 200.240112 Tempo: 0.32810 Ganho: 0.952392

Resultado Conc. 8 threads: 200.240112 Tempo: 0.28121 Ganho: 1.111198

**Teste G (cos(exp(-x))\*(0.005\*x^3)+1) (Intervalo: -10 a 20)**

Resultado Seq.: 219.410629 Tempo: 0.20311

Resultado Conc. 2 threads: 219.410629 Tempo: 0.18749 Ganho: 1.666648

Resultado Conc. 4 threads: 219.410629 Tempo: 0.18750 Ganho: 1.66656

Resultado Conc. 8 threads: 219.410629 Tempo: 0.18748 Ganho: 1.666737