CI1164 – Introdução à Computação Científica

Prof. Guilherme Derenievicz
Prof. Armando Delgado

Exercícios de Revisão para Prova 02

Questão 1

Considere uma função para calcular $d = A \times B \times s$, onde $\{s, d\} \in \mathbb{R}^N$ são dois vetores de tamanho N e $\{A, B\} \in \{\mathbb{R}^N \times \mathbb{R}^N\}$ duas matrizes de tamanho $N \times N$. Considere ainda que esta função é executada muitas vezes, e que todas estruturas cabem na cache do processador.
Responda:

(a) Supondo que a ordem das operações não seja relevante neste caso, qual a forma mais eficiente de computar o valor de d ? Justifique sua resposta.
   i. $d = (A \times B) \times s$
   ii. $d = A \times (B \times s)$

(b) Escreva o código que efetue o cálculo de d de acordo com sua opção no item anterior.

```
void updCell(double *s, double *A, double *B, double *d, long SIZE)
{
  double *aux;
  // aloca estrutura aux, seja ela qual for (não precisa alocar)
  ...
  // inicia os cálculos

}
```

Questão 2

Observe o código abaixo que calcula a seguinte integral pelo método de Monte Carlo:

$$\iint_a^b f(x,y)\,dx\,dy, \quad onde \quad f(x,y) \;=\; 10^5 x^2 + y^2 - (x^2 + y^2)^2 + 10^{-5}(x^2 + y^2)^4$$

```
double calc_integral_mc(int n, double a, double b){
   double sum, x, y;
   for(int i=0; i < n; i++) {
      x = a + (double) rand() * ( (double) 1.0 / (RAND_MAX * (b – a)) );
      y = a + (double) rand() * ( (double) 1.0 / (RAND_MAX * (b – a)) );
      sum += 1e5 * pow(x, 2) + pow(y, 2) – pow( pow(x,2) + pow(y,2), 2 )
             + 1e-5 * pow(pow(x,2.0) + pow(y,2), 4)
 }
 return (b - a)*(b – a) * sum / n;
}
```

Otimize este código o máximo possível. Destaque as decisões de implementação que aumentam a eficiência do seu código, **justificando-as** (i.e. você deve explicar por que sua alternativa aumenta o desempenho). A eficiência do código é o principal critério de avaliação. A corretude é atributo indispensável.

Questão 3

Sejam um conjunto de pontos $(x_i, y_i) \in \mathbb{R}^2$, $x_i < x_{i+1}$, $1 \leqslant i \leqslant N$ e $f(x_i) = y_i$ representando uma função a ser utilizada por determinada aplicação. Você foi contratada(o) para implementar um programa que retorne/calcule o valor de $f(z)$, $z \in \mathbb{R}$ para $x_2 < z < x_{N-1}$. Caso o ponto $(z, f(z))$ não esteja definido no conjunto, você deve interpolar a função através de um polinômio de grau 4 (quatro) utilizando os pontos x_i mais próximos de z. Considerando que os pontos não são uniformemente espaçados, responda:

(a) Quantos pontos são necessários para calcular um valor interpolado $f(z)$?

(b) Qual dos métodos de interpolação deve ser utilizado: Newton ou Newton-Gregory? **Justifique!**

(c) Qual o problema em se utilizar um único polinômio interpolador definido a partir de todos os pontos, quando o número de pontos é muito grande?

(d) Considerando uma implementação eficiente do programa definido no enunciado, qual será o maior custo computacional: acesso à memória ou uso de CPU? Justifique.

Questão 4

a) Por que o código abaixo é ineficiente? **Justifique!**

b) Reescreva-o de forma a sanar o problema.

```
...
/* n é muito grande */
for(i=0; i<n; i++)  {
   if( x == A ) {
       FuncaoA(i);
   }
   else if( x == B ) {
       FuncaoB(i);
   }
   else {
       FuncaoC(i);
   }
}
```

Questão 5

Qual das versões de código abaixo é mais rápida e por que? **Justifique**!

Versão A	Versão B

`int p[SIZE];`
`for (long x=0; x<NUM_COL; ++x) {`
`   for (long y=0; y<NUM_LIN; ++y) {`
`      p[x+y*NUM_COL]++`
`   }`
`}`
	`int p[SIZE];`
	`for (long y=0; y<NUM_LIN; ++y) {`
	`   for (long x=0; x<NUM_COL; ++x) {`
	`      p[x+y*NUM_COL]++`
	`   }`
	`}`

Questão 6

Sejam um conjunto de pontos $(x_i, y_i) \in \mathbb{R}^2$, $x_i < x_{i+1}$, $1 \le i \le N$ e $f(x_i) = y_i$ uma função utilizada por determinada aplicação. Você foi contratada(o) para implementar um programa que retorne/calcule o valor de $f(z), z \in \mathbb{R}$ para $x_2 \le z \le x_{N-1}$. Caso o ponto $(z, f(z))$ não esteja definido no conjunto, você deve interpolar a função através de um polinômio de grau 3 (três) utilizando os pontos x_i mais próximos de z. Responda:

(a) Quantos pontos são necessários para calcular um valor interpolado $f(z)$?

(b) Qual dos método de interpolação pode ser utilizado: Newton ou Newton-Gregory? **Justifique!**

(c) Caso você utilizasse o polinômio interpolador de Lagrange ao invés de Newton, qual das estruturas de dados abaixo seria mais eficiente para armazenar o conjunto de pontos? **Justifique!**

Estrutura A	**Estrutura B**

`struct Ponto {`
`   double x,y;`
`}`
 	
`struct Ponto p[MAXPTOS];`
	`struct Pontos {`
	`   double x[MAXPTOS];`
	`   double y[MAXPTOS];`
	`}`
	
	`struct Pontos p;`

Questão 7

Seja uma função $f: \mathbb{R}^2 \rightarrow \mathbb{R}$. Escreva um programa em linguagem C que calcule a integral $\iint_a^b f(x,y)\,dx\,dy$ utilizando o Método de Monte Carlo e a função `f(x,y)` declarada na primeira linha do código. O número de pontos n a ser inicialmente amostrado é dado. O programa deve executar diversas iterações, dobrando o número de pontos amostrados a cada iteração, até que a diferença entre a integral calculada entre duas iterações consecutivas seja menor do que ϵ (epsilon).

Integral por Monte Carlo: $\iint_a^b f(x,y)\,dx\,dy \approx \frac{(b-a)^2}{n}\left(\sum_{i=0}^{n-1} f(x_i, y_i)\right)$

```
double f (double x, double y);
...
double integral (double a, double b, double epsilon, uint n)
{

}
```

Questão 8

Otimize o código abaixo o máximo possível. Destaque as decisões de implementação que aumentam a eficiência do seu código, **justificando-as** (i.e. você deve explicar por que sua alternativa aumenta o desempenho). A eficiência do código é o principal critério de avaliação. A corretude é condição essencial.

```
int n;
double A, B, C, D, E;
double F[n*n], double G[n*n], double R[n*n];

for(int h = 0; h < pow(n, 2.0); h++) {
    R[h] = G[h] – A * F[h];
    if(h-1 >= 0)
      R[h] -= B * F[h - 1];
    if(h+1 < pow(n, 2.0) - 1)
      R[h] -= C * F[h + 1];
    if(h - n >= 0)
      R[h] -= D * F[h - n];
    if(h + n < pow(n, 2.0) – 1)
      R[h] -= E * F[h + n];
}
```

Questão 9

Observe o código para o método de Jacobi em duas dimensões apresentado abaixo.

```
double phi[iMAX+1][jMAX+1][2];
int t0=0, t1=1, aux;
...
for (long it=1; it<ITER; ++it) {
  for (long i=1; i < iMAX; ++i)
    for (long j=1; j < jMAX; ++j) {
       phi[i][j][t1] = 0.25 * (phi[i-1][j][t0] + phi[i+1][j][t0] +
                       phi[i][j-1][t0] + phi[i][j+1][t0]);
    }
   aux = t0; t0 = t1; t1 = aux; /* troca os vetores */
}
```

a) Descreva dois **problemas** na implementação do código acima que o tornam ineficiente

b) Reescreva o código de forma a torná-lo o mais eficiente possível

c) **Explique** por que sua versão melhora cada um dos problemas apresentados

Questão 10

Sejam um conjunto de pontos $P=(x_i, y_i) \in \mathbb{R}^2, \quad x_i < x_{i+1}, \quad 1 \le i \le N$ e $f(x_i) = y_i$ uma função utilizada por determinada aplicação. Você foi contratada(o) para implementar um programa que retorne/calcule o valor de $f(z), z \in \mathbb{R}$ para $x_2 \le z \le x_{N-1}$. Caso o ponto $(z, f(z))$ não esteja definido no conjunto P , você deve interpolar a função através de um polinômio de <u>grau três</u> utilizando os pontos $x_i, x_{i+1} < z < x_{i+2}, x_{i+3}$ mais próximos de z .

Lagrange: $p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$ e $L_i(x) = \prod_{j=0, j \ne i}^{n} \frac{(x - x_j)}{(x_i - x_j)}$

Newton: $p_n(x) = d_0 + d_1(x - x_0) + d_2(x - x_0)(x - x_1) + \cdots + d_n(x - x_0) \cdots (x - x_{n-1})$, onde $d_k, k = 0, 1, \ldots, n$ são as diferenças divididas de ordem k .

a) Qual o método mais eficiente para o seu programa: Newton ou Lagrange? **Justifique!**

b) Considerando os polinômios interpoladores de Lagrange e Newton, qual das estruturas de dados abaixo seria mais eficiente para armazenar o conjunto de pontos em cada caso? **Justifique!**

<table>
<tr><th>Estrutura A</th><th>Estrutura B</th></tr>
<tr><td><pre>struct Ponto {
   double x,y;
}

struct Ponto p[MAXPTOS];</pre></td><td><pre>struct Pontos {
   double x[MAXPTOS];
   double y[MAXPTOS];
}
struct Pontos p;</pre></td></tr>
</table>

c) Considerando uma implementação eficiente do programa definido no enunciado, qual será o maior custo computacional: acesso à memória ou uso de CPU? Justifique.

Questão 11

a) Reescreva o código abaixo de forma a melhorar seu desempenho (`M_PI` é uma constante) e justifique **por que** sua versão do código é mais eficiente?

```
...
double a[n][n],x[n],y[n],b[n],z[n];
...
for (i=0; i<n; i++)
  for (j=0; j<n; j++)
    a[j][i] = x[i] + y[j] * cos((i%8)*M_PI/7.0);
for (i=0; i<n; i++)
  b[i]=1.0/x[i]+z[i];
```

b) Considerando que a instrução “`pragma unroll (8)`” desenrola o laço 8 vezes, por que motivo o **Código A** tem um desempenho <u>pior</u> do que o **Código B**?

Código A

```
#pragma unroll (8)
for (i=0; i<n; i++)
{
   a[i] = b[i]+c[i]*d[i];
   e[i] = f[i]-g[i]*h[i]+p[i];
   q[i] = r[i]+s[i];
}
```

Código B

```
#pragma unroll (8)
for (i=0; i<n; i++)
   a[i] = b[i]+c[i]*d[i];

#pragma unroll (8)
for (i=0; i<n; i++)
   e[i] = f[i]-g[i]*h[i]+p[i];

#pragma unroll (8)
for (i=0; i<n; i++)
   q[i] = r[i]+s[i];
```

Questão 12

Seja uma função $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Escreva um programa em linguagem C que calcule a integral $\iint_a^b f(x,y)\,dx\,dy$ utilizando o Método dos Retângulos. O número de pontos n inicial é dado, e o espaçamento entre os pontos h é igual em ambas dimensões e dado por $h = b - a/n \Rightarrow \{x_i, y_i\} = a + h * i$. O programa deve executar diversas iterações, reduzindo o valor do intervalo ao meio a cada iteração, até que o Erro Aproximado Absoluto da integral seja menor do que ϵ dado.

Método dos Retângulos: $\int_b^a f(x)\,dx \approx \int_b^a p_0(x)\,dx = h\left(\sum_{i=0}^{n-1} f(x_i)\right)$

```
double f (double x, double y);
...
double integral (double a, double b, double epsilon, uint n)
{

}
```

O Método dos Retângulos é apropriado para calcular a integral de funções de alta dimensionalidade? **Justifique**.

Questão 13

A versão **A** do código abaixo demora o dobro do tempo para executar do que a versão **B**. Por que isso ocorre?

Versão A	Versão B

`struct DATA {`
`   int a, b, c, d;`
`};`
`DATA p[N];`
 	
`for (long i=0; i<N; ++i) {`
`   p[i].a = p[i].b`
`}`
	`struct DATA {`
	`   int a, b;`
	`};`
	`DATA p[N];`
	
	`for (long i=0; i<N; ++i) {`
	`   p[i].a = p[i].b`
	`}`

Questão 14

Considere o código abaixo:

```
for (int i=0; i<N; ++i)
   for (int j=0; j<N; ++j)
      c[i] = c[i] + A[i][j] * b[j]
```

a) Reimplemente este código aplicando apropriadamente a técnica de “loop unroll” com tamanho quatro.

b) O código com o laço desenrolado é mais eficiente que o código original em uma arquitetura x64? Justifique sua resposta.

Questão 15

Responda às seguintes questões:

a) Qual o problema de se utilizar muitos pontos para calcular o polinômio interpolador de uma função tabulada? Como proceder para calcular um valor interpolado a partir de um grande conjunto de pontos?

b) Explique que tipo de problemas com registradores podem ser causados por um “loop unroll”.

c) Porque o acesso em coluna é ineficiente para matrizes bidimensionais em linguagem C?

d) Por que a integração numérica pelo método dos trapézios não é uma boa solução para problemas de alta dimensionalidade?

Questão 16

Considerando a seguinte implementação do Método de Lagrange para interpolação, responda as questões abaixo.

```
// Método de Lagrange para interpolação da função f(x) tabelada em n pontos.
// x: valor no qual f(x) deve ser aproximada
// n: quantidade de pontos
// tab: vetor de tamanho 2n contendo pares (xi, f(xi)): [x0, fx0, x1, fx1, ...]
double lagrange(double x, double *tab, int n) {
	double Px, Li, xi, xj, fi;

	Px = 0.0;
	for (i = 0; i < n; ++i)
	{
		Li = 1.0;
		xi = tab[2*i];
```

```
            for (int j = 0; j < n, ++j)
            {
                  xj = tab[2*j];
                  Li *= (x-xj)/(xi-xj);
            }
            fi = tab[2*i+1];
            Px += Li*fi;
      }
      return Px;
}
```

- **a)** Identifique o que está incorreto no código acima e mostre como resolver, otimizando a sua solução.
- **b)** Por que a estrutura tab não é boa para esse programa? Proponha uma alternativa melhor e justifique.
- **c)** Considerando o programa resultante dos itens (a) e (b) há a possibilidade do uso de AVX? Se sim, indique em qual trecho, justificando sua resposta. Se não, indique quais modificações podem ser feitas a fim de aproveitar os registradores AVX.
- **d)** Sabendo que essa função nunca será usada com x pertencente à tabela de pontos (isto é, x != xi para todo i=0..n-1) modifique o programa resultante dos itens (a) e (b) a fim de diminuir a quantidade de operações de ponto flutuante executadas e responda: ao comparar as duas versões, há diferença no tempo de execução? e na taxa de MFLOP/s? Justifique sua resposta.
- **e)** É possível otimizar o acesso à memória do programa resultante dos itens (a) e (b) com a técnica de Loop Unroll & Jam? Se sim, mostre como fica o código otimizado. Se não, justifique sua resposta.