Exercício de Programa 3

Caio Vinícius Dadauto 7994808

14 de maio de 2013

Problema 1

Seja a função $f(x)$ dada por:

$$f(x) = 8 - 5x^4 \tag{1}$$

integrando $f(x)$ dentro do intervalo de $0 \leq x \leq 1$, tem-se:

$$\int_0^1 (8 - 5x^4) = 7 \tag{2}$$

Tendo o valor analítico da integral de $f(x)$, é possível implementar um programa que aproxima o valor da integral em (2) pelo método de Simpson. Programa que analisa para cada interação o modulo do erro entre o valor numérico e o valor analítico da integral (2), ou seja,

$$erro = |I_{num} - I| \tag{3}$$

onde a I_{num} é o valor numérico e I é o valor analítico.

Tal programa segue abaixo.

Programa 1: Implementação para o método de Simpson.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 float erro (float integral) {
5 ···if (integral - 7 < 0)
6 ·····return 7 - integral;
7 ···return integral - 7;
8 }
```

```
9
10 void imprime (int p, FILE *saida, float integral) {
11 ···fprintf (saida, "%d & %8.f & %f & %f\\\\\n",
12 ···p, pow (2, p), integral, erro (integral));
13 ···fprintf (saida, "\\midrule\n");
14 }
15
16 float funcao (float a) {
17 ···return 8 - (5 * pow (a, 4));
18 }
19
20 void simpson (int p, float *integral) {
21 ···long
22 ·····j;
23 ···float
24 ·····soma_par = 0,
25 ·····soma_impar = 0,
26 ·····n_particao = pow (2, p),
27 ·····particao = 1 / pow (2, p);
28
29 /* soma_par ou _impar refere-se a particao par ou impar. */
30 /* O que difere do valor de j, o qual assume valores impares */
31 /* para particoes pares e valores pares para particoes impares. */
32 ···for (j = 1; j < n_particao; j += 2) {
33 ·····soma_par +=  funcao (particao * j);
34 ···}
35 ···soma_par *= 4;
36 ···for (j = 2; j < n_particao - 1; j += 2) {
37 ·····soma_impar += funcao (particao * j);
38 ···}
39 ···soma_impar *= 2;
40 ···*integral += soma_par + soma_impar;
41 ···*integral *= (particao / 3);
42 }
43
44 int main () {
45 ···FILE
46 ·····*arquivo;
47 ···int
48 ·····p;
49 ···float
50 ·····integral;
51
52 ···arquivo = fopen ("simpson.tex", "w");
53 ···for (p = 1; p <= 25; ++ p) {
54 ·····integral = funcao( 0 ) + funcao( 1 );
55 ·····simpson (p, &integral);
```

```
56 ......imprime (p, arquivo, integral);
57 ...}
58 ...fclose (arquivo);
59 ...return 0;
60 }
```

A tabela 1 apresenta os valores numéricos obtidos pelo programa para diferentes números de intervalos entre 0 e 1.

p	**N** (2^p)	I_{num}	**erro**
1	2	6.958333	0.041667
2	4	6.997396	0.002604
3	8	6.999837	0.000163
4	16	6.999990	0.000010
5	32	6.999999	0.000001
6	64	7.000000	0.000000
7	128	7.000000	0.000000
8	256	7.000001	0.000001
9	512	6.999999	0.000001
10	1024	6.999998	0.000002
11	2048	6.999996	0.000004
12	4096	7.000000	0.000000
13	8192	7.000000	0.000000
14	16384	7.000003	0.000003
15	32768	7.000002	0.000002
16	65536	7.000003	0.000003
17	131072	7.000101	0.000101
18	262144	7.000114	0.000114
19	524288	7.000272	0.000272
20	1048576	7.002830	0.002830
21	2097152	7.002614	0.002614

Continuação na proxima página.

Continuação da página anterior.			
22	4194304	7.017547	0.017547
23	8388608	7.097704	0.097704
24	16777216	7.181083	0.181083
25	33554432	7.565933	0.565933

Tabela 1: Dados obtidos pelo método de Simpson.

A partir da tabela 2 é possível perceber a convergência dos dados para pequenos valores de N e, para grandes valores de N, a divergência dos mesmo. Tal fato é dado pelo erro de roundoff, que passa a ser relevante devido ao grande número de partições do intervalo de integração. Assim, foi plotado dois conjuntos de pontos no plano (*erro* x *p*), um para o método de Simpson em precisão simples e outro em dupla precisão. A figura 1 apresenta o gráfico com os conjuntos de pontos obtidos para as duas precisões.

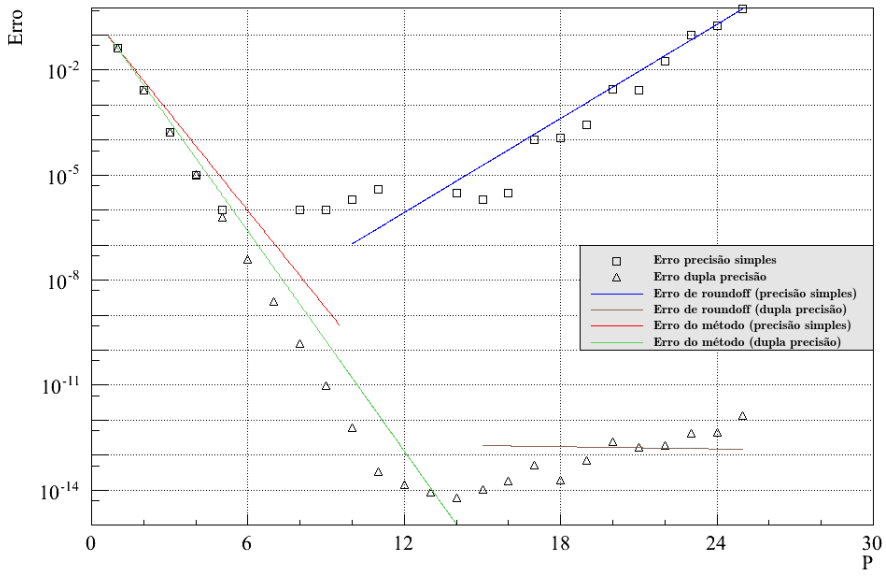

Figura 1: Gráfico para precisão simples e dupla .

Atrravés do método dos mínimos quadrados é possível ajustar uma função ao conjunto de pontos onde ainda há convergência do valor numérico para o valor literal da integral. Essa função pode ser dada por:

$$erro = \left(\frac{1}{2^p}\right)^a \tag{4}$$

onde a é um parametro a ser ajustado. Com o ajuste, obteve-se os seguintes valores para o parâmetro a:

Para precisão simples	3.8
Para dupla precisão	3.6

O que está proximo da previsão teórica, onde o erro do método é da ordem de $O(h^4)$.

Por outro lado, é possível ajustar outra função, porém, desta vez, ao conjunto de pontos onde o erro passa a aumentar conforme o aumento de partições. Função que pode ser dada por:

$$erro = (2^p)^b \tag{5}$$

onde b é um parametro a ser ajustado. Com o ajuste, obteve-se os seguintes valores para o parâmetro b:

Para precisão simples	0.6
Para dupla precisão	0.2

O que se assemelha a previsão teórica, onde o erro de roundoff é da ordem de $O(\sqrt{N})$.

Problema 2

O periodo para um pendulo que trabalha dentro de ângulos apreciaveis ($\theta_0 > 10°$), quando se desconsidera a resistência do ar, pode ser dada por:

$$4\sqrt{\frac{l}{g}}\int_0^{\pi/2}\frac{1}{\sqrt{1-\sin^2(\theta_0/2)\sin^2\theta}}\mathrm{d}\theta \tag{6}$$

onde θ_0 é o angulo inicial, l o comprimento do pêndulo e g a aceleração da gravidade.

Partindo da equação (6) é possível implementar um programa que determina o valor do periodo para diferentes valores de θ_0 (20 valores dentro do intervalo $[0, \pi[$). Programa, este, que utiliza do método dos trapézios, o qual divide o intervalo de $[0, \pi/2]$ em 100000 partições, onde, em cada uma delas, o ingrando é aproximado por uma reta. O Código do programa segue logo abaixo:

Programa 2: Implementação para o método do trapézio.

```
1  #include <stdio.h>
2  #include <math.h>
3  #define PI 3.14159265
4  #define G 9.80665
5  #define MAX 100000
6
7  void imprime (float theta, FILE *saida, float periodo) {
8  ···fprintf (saida, "%f & %f \\\\\n", theta, periodo);
9  ···fprintf (saida, "\\midrule\n");
10 }
11
12 float funcao (float rad, float theta) {
13 ···return 1 / (sqrt (1 - pow (sin (theta), 2) *
14 ···pow (sin (rad), 2)));
15 }
16
17 float trapezio (float theta) {
18 ···int
19 ·····i;
20 ···float
21 ·····integral,
22 ·····soma = 0,
23 ·····particao = PI / (2 * MAX);
```

```
24
25 ···integral = funcao (0, theta) + funcao (PI / 2, theta);
26 ···for (i = 1; i < MAX; ++ i) {
27 ·····soma += funcao (particao * i, theta);
28 ···}
29 ···integral += 2 * soma;
30 ···integral *= particao / 2;
31 ···return integral;
32 }
33
34 int main () {
35 ···FILE
36 ·····*arquivo;
37 ···float
38 ·····theta,
39 ·····periodo;
40
41 ···arquivo = fopen ("trapezio.tex", "w");
42 ···for (theta = 0; theta < PI; theta += PI / 20) {
43 ·····periodo = trapezio (theta) * 4 * sqrt (1 / G);
44 ·····imprime (theta, arquivo, periodo);
45 ···}
46 ···fclose (arquivo);
47 ···return 0;
48 }
```

A tabela 2 apresenta os valores do periodo para seus respectivos valores de θ_0.

θ_0 **(rad)**	**T (s)**
0.000000	2.006409
0.157080	2.018821
0.314159	2.057160
0.471239	2.124052
0.628319	2.224411
0.785398	2.368301
0.942478	2.571656
1.099557	2.865801
1.256637	3.320819
1.413717	4.158063
1.570796	65.931862
1.727876	4.158067
1.884955	3.320821
2.042035	2.865802
2.199115	2.571656
2.356194	2.368302
2.513274	2.224411
2.670354	2.124052
2.827433	2.057160
2.984513	2.018821

Tabela 2: Valores daequação (6) para diferentes valores de θ_0.

Agora, fazendo θ_0 assumir 200 valores diferentes dentro do intervalo $[0, \pi[$, é possível plotar um gráfico do periodo contra θ_0. O qual é apresentado na figura 2.

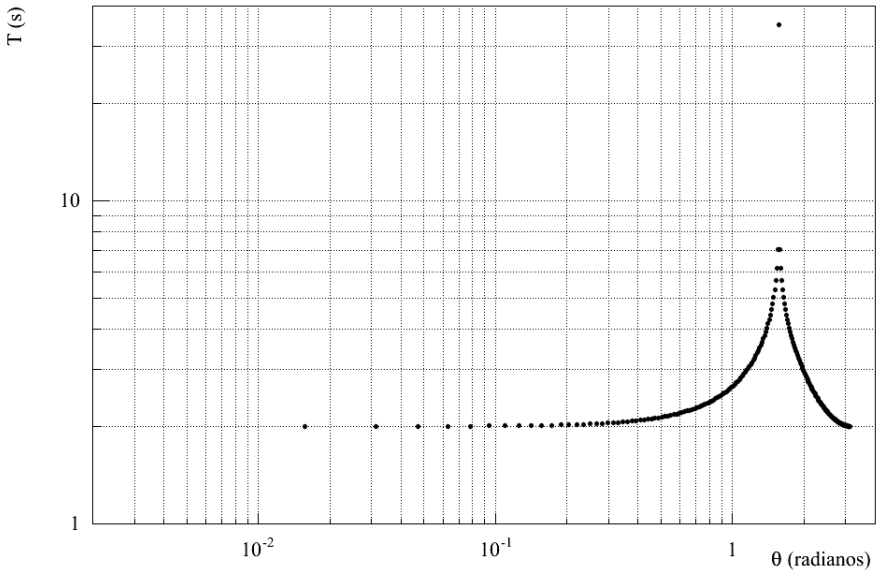

Figura 2: Gráfico do periodo em função e de θ_0.

Problema 3

Dado $f(x)$ como:

$$f(x) = x^3 \tag{7}$$

é possível determinar a área sobre o gráfico de $f(x)$ no intervalo $0 < x < 1$, através do método de Monte-Carlo. Para tal, implementou-se um programa que utiliza do "linear congruential method" para gerar os números pseudo-aleatórios uniformemente distribuidos. Tomando com semente o número 7994808.

O programa se baseia em diferentes números de tentativas, sendo que, em cada tentativa, são sorteados 100 pontos ($0 < x < 1$ e $0 < y < 1$). Onde a cada 2^n ($1 \leq n \leq 17 | n \in \mathbb{N}$) tentativas é calculado a média das áreas

obtidas, o desvio padrão e o desvio padrão da média. O código do pragrama segue logo abaixo.

Programa 3: Implementação para o método de Monte-Carlo.

```
#include <stdio.h>
#include <math.h>
#define M 2147483647
#define A 16807
#define EPS 1E-4

void imprime (FILE *saida, float area[]);
float desv_pad (float area_media, float area[], int N);
float media (float area[], int N);
short sao_iguais (float a, float b);
void aleatorio (unsigned long *aux, float *x, float *y);
short ta_dentro (float a, float b);

void imprime (FILE *saida, float area[]) {
   int
      i;
   float
      area_media,
      desvio;

   for (i = 1; i <= 17; ++i ) {
      area_media = media (area, (int) pow(2, i)),
      desvio = desv_pad (area_media, area, (int) pow(2, i));
      fprintf (saida, "%d & %.5f & %.3f & %.5f\\\\\n", (int)pow(2,i),
      area_media, desvio, desvio/(sqrt ((int)pow(2,i) + 1)));
      fprintf (saida, "\\midrule\n");
   }
}

float desv_pad (float area_media, float area[], int N) {
   int
      i;
   float
      soma = 0;

   for (i = 0; i < N; ++ i)
      soma += pow ((area_media - area[i]), 2);
   return sqrt (soma / (N - 1));
}

float media (float area[], int N) {
   int
```

```
43 ......i;
44 ...float
45 ......soma = 0;
46
47 ...for (i = 0; i < N; ++ i)
48 ......soma += area[i];
49 ...return soma /= N;
50 }
51
52 short sao_iguais (float a, float b) {
53 ...if (a - b <= EPS && b - a <= EPS)
54 ......return 1;
55 ...return 0;
56 }
57
58 short ta_dentro (float a, float b) {
59 ...if (b < a * a * a || sao_iguais (b, a * a * a))
60 ......return 1;
61 ...return 0;
62 }
63
64 void aleatorio (unsigned long *aux, float *x, float *y) {
65 ...*x = (A * *aux) % M;
66 ...*aux = (unsigned long) *x;
67 ...*x /= M;
68 ...*y = (A * *aux) % M;
69 ...*aux = (unsigned long) *y;
70 ...*y /= M;
71 }
72
73 int main () {
74 ...FILE
75 ......*arquivo;
76 ...int
77 ......k,
78 ......j,
79 ......i;
80 ...unsigned long
81 ......aux = 7994808;
82 ...float
83 ......x,
84 ......y,
85 ......area[131072];
86
87 ...arquivo = fopen ("montecarlo.tex", "w");
88 ...for (j = 0; j < 131072; ++j) {
89 ......k = 0;
```

```
90 ......for (i = 0; i < 100; ++i) {
91 ..........aleatorio (&aux, &x, &y);
92 ..........if (ta_dentro (x, y))
93 ..............++k;
94 ......}
95 ....../* A area do quadrado que contem y = x^3 é 1 */
96 ......area[j] = (float) k / 100;
97 ...}
98 ...imprime (arquivo, area);
99 ...fclose (arquivo);
100 ...return 0;
101 }
```

A tabela 4 apresenta os os valores médios da área, o desvio padrão e o desvio padrão da média a cada 2^n tentativas.

$N_t(2^n)$	I_m	σ	σ_m
2	0.18500	0.035	0.02041
4	0.21250	0.038	0.01688
8	0.22750	0.040	0.01342
16	0.24812	0.045	0.01085
32	0.24563	0.044	0.00771
64	0.24422	0.041	0.00505
128	0.24398	0.040	0.00348
256	0.24418	0.039	0.00240
512	0.24414	0.038	0.00170
1024	0.24406	0.038	0.00119
2048	0.24404	0.038	0.00084
4096	0.24403	0.038	0.00059
8192	0.24404	0.038	0.00042
16384	0.24403	0.038	0.00030
32768	0.24405	0.038	0.00021
65536	0.24405	0.038	0.00015
131072	0.24402	0.038	0.00011

Tabela 3: Valores obtidos para o método de Monte-Carlo.