TRABALHO 4

May 27, 2020

Análise Numérica (M2018)

Francisco Gonçalves - 201604505
Márcia Dias - 201704466

Departamento de Ciência de Computadores
Faculdade de Ciências de Universide do Porto

Contents

0.1 ReadMe . . . 2
0.2 Primeiro Exercício . . . 3
0.2.1 Regra dos retângulos . . . 3
0.2.2 Regra dos trapézios . . . 4
0.2.3 Regra de Simpson . . . 4
0.3 Segundo Exercício . . . 5
0.3.1 Regra dos retângulos . . . 5
0.3.1.1 Cálculo de M . . . 6
0.3.1.2 Erro absoluto majorado inferior a $\epsilon = 10^{-5}$ . . . 6
0.3.1.3 Erro absoluto majorado inferior a $\epsilon = 10^{-7}$ . . . 6
0.3.1.4 Erro absoluto majorado inferior a $\epsilon = 10^{-9}$ . . . 6
0.3.1.5 Resultados . . . 6
0.3.2 Regra dos trapézios . . . 7
0.3.2.1 Cálculo de M . . . 8
0.3.2.2 Erro absoluto majorado inferior a $\epsilon = 10^{-5}$ . . . 8
0.3.2.3 Erro absoluto majorado inferior a $\epsilon = 10^{-7}$ . . . 8
0.3.2.4 Erro absoluto majorado inferior a $\epsilon = 10^{-9}$ . . . 8
0.3.2.5 Resultados . . . 8
0.3.3 Regra de Simpson . . . 9
0.3.3.1 Cálculo de M . . . 10
0.3.3.2 Erro absoluto majorado inferior a $\epsilon = 10^{-5}$ . . . 10
0.3.3.3 Erro absoluto majorado inferior a $\epsilon = 10^{-7}$ . . . 10
0.3.3.4 Erro absoluto majorado inferior a $\epsilon = 10^{-9}$ . . . 10
0.3.3.5 Resultados . . . 10
0.3.4 Terceiro Exercício . . . 11

0.1 ReadMe

Este trabalho é composto pelos seguintes programas:

- *RectangleRule.c* - [C]
- *TrapezoidalRule.c* - [C]
- *SimpsonRule.c* - [C]

Para testar os programas basta fazer:

1. *gcc $FILENAME$ -lm*
2. *./a.out*

⚠ NOTA ⚠

Por favor assegure-se que instalou todos os módulos e programas necessários ao bom funcionamento dos programas, nomeadamente o `gcc`

Os programas aqui exibidos assim como os resultados obtidos podem ser consultados em detalhe aqui: `https://github.com/1Skkar1/Numerical-Analysis/tree/master/Trabalho4`

Pretende-se calcular:

$$I = \int_0^3 \sin(\cos(\sin(\cos(x^2)))) \, \mathrm{d}x$$

0.2 Primeiro Exercício

Escrevam programas, na linguagem que preferirem, que calculem valores aproximados de I usando as seguintes fórmulas de integração numérica:

- *Regra dos retângulos*
- *Regra dos trapézios*
- *Regra de Simpson*

com erro absoluto majorado inferior a ϵ *dado, fazendo uma partição do intervalo* [0,3] *em* n *subintervalos de igual amplitude. O valor de* n *pode ser calculado fora do programa.*

0.2.1 Regra dos retângulos

rectangle_rule( ) in *RectangleRule.c* (C)

```c
double rectangle_rule(float a, float b, float n){
  int j = b - a;
  float h = j / n;
  float s_1 = 0.0;
  for(int i = 1; i < n; i++){
    s_1 += f(i - 1);
  }
  return s_1 * h;
}
```

0.2.2 Regra dos trapézios

trapezoidal_rule( ) in *TrapezoidalRule.c* (C)

```c
double trapezoidal_rule(float a, float b, int n){
  float h = (b - a) / n;
  float s = (h / 2) * (f(a) + f(b));
  for(int i = 1; i < n; i++)
    s += f(i);
  return h * s;
}
```

0.2.3 Regra de Simpson

simpson_rule( ) in *SimpsonRule.c* (C)

```c
double simpson_rule(float a, float b, int n){
  if(a > b){
    printf("Incorrect bounds.");
    return -1;
  }
  if(n % 2 != 0){
    printf("n must be an even integer.");
    return -1;
  }
  float h = (b - a) / n;
  float s_1 = 0.0;
  for(int i = 1; i < n; i += 2)
    s_1 += 4 * f(i);
  float s_2 = 0.0;
  for(int i = 2; i < n; i += 2)
    s_2 += 2 * f(i);
  return (h / 3) * (f(a) + f(b) + s_1 + s_2);
}
```

Relativamente aos valores que cada função necessita:

- $a \rightarrow$ corresponde ao limite mínimo da função, sendo sempre 0.
- $b \rightarrow$ corresponde ao limite máximo da função, sendo sempre 3.
- $n \rightarrow$ corresponde ao número de partições do intervalo de integração, variando de acordo com o erro a considerar e com a regra que o usa. Os cálculos efetuados para obter estes valores são demonstrados no próximo exercício.

0.3 Segundo Exercício

Usem os vossos programas para calcular, se possível, valores aproximados de I com:

- *Erro absoluto majorado inferior a* $\epsilon = 10^{-5}$
- *Erro absoluto majorado inferior a* $\epsilon = 10^{-7}$
- *Erro absoluto majorado inferior a* $\epsilon = 10^{-9}$

As três implementações relativas a cada regra de integralização apresentam uma função principal *main()* semelhante, alterando apenas a função da regra a usar e o valor de n, que é distinto para cada método e para cada erro absoluto:

0.3.1 Regra dos retângulos

main( ) in *RectangleRule.c* (C)

```c
int main(){
  clock_t start = clock();
  printf("Com 5 Casas decimais: %.5f\n",rectangle_rule(0.0, 3.0, 123570.0));
  clock_t end = clock();
  float seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);

  start = clock();
  printf("Com 7 Casas decimais: %.7f\n",rectangle_rule(0.0, 3.0, 12500000.0));
  end = clock();
  seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);

  start = clock();
  printf("Com 9 Casas decimais: %.9f\n",rectangle_rule(0.0, 3.0, 1250000000.0));
  end = clock();
  seconds = (float)(end - start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);
}
```

0.3.1.1 Cálculo de M

$M = max|d^{'}(x)| = 1.373$

0.3.1.2 Erro absoluto majorado inferior a $\epsilon = 10^{-5}$

$\frac{3}{2} \times h \times 1.373 \leq 5 \cdot 10^{-5}$
$\iff h \leq 2.4 \cdot 10^{-5}$

$n = \frac{3}{2.4 \cdot 10^{-5}} = 125000$

0.3.1.3 Erro absoluto majorado inferior a $\epsilon = 10^{-7}$

$\frac{3}{2} \times h \times 1.373 \leq 5 \cdot 10^{-7}$
$\iff h \leq 2.4 \cdot 10^{-7}$

$n = \frac{3}{2.4 \cdot 10^{-7}} = 12500000$

0.3.1.4 Erro absoluto majorado inferior a $\epsilon = 10^{-9}$

$\frac{3}{2} \times h \times 1.373 \leq 5 \cdot 10^{-9}$
$\iff h \leq 2.4 \cdot 10^{-9}$

$n = \frac{3}{2.4 \cdot 10^{-9}} = 1250000000$

0.3.1.5 Resultados

```
Com 5 casas decimais: 2.16803
Time: 0.019557
Com 7 casas decimais: 2.1801884
Time: 1.700926
Com 9 casas decimais: 0.040265318
Time: 212.122772
```

0.3.2 Regra dos trapézios

main( ) in *TrapezoidalRule.c* (C)

```c
int main(){
  clock_t start = clock();
  printf("Com 5 Casas decimais: %.5f\n",trapezoidal_rule(0.0, 3.0, 883.0));
  clock_t end = clock();
  float seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);

  start = clock();
  printf("Com 7 Casas decimais: %.7f\n",trapezoidal_rule(0.0, 3.0, 8824.0));
  end = clock();
  seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);

  start = clock();
  printf("Com 9 Casas decimais: %.9f\n",trapezoidal_rule(0.0, 3.0, 88236.0));
  end = clock();
  seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);
}
```

0.3.2.1 Cálculo de M

$M = max|d''(x)| = 17.045$

0.3.2.2 Erro absoluto majorado inferior a $\epsilon = 10^{-5}$

Cálculo de n:
$|-\frac{h^2}{12} \times 3 \times 17.045| \leq 5 \cdot 10^{-5}$
$\iff h \leq 3.4 \cdot 10^{-3}$

$n = \frac{3}{3.4 \cdot 10^{-3}} = 883$

0.3.2.3 Erro absoluto majorado inferior a $\epsilon = 10^{-7}$

Cálculo de n:
$|-\frac{h^2}{12} \times 3 \times 17.045| \leq 5 \cdot 10^{-7}$
$\iff h \leq 3.4 \cdot 10^{-4}$

$n = \frac{3}{3.4 \cdot 10^{-4}} = 8824$

0.3.2.4 Erro absoluto majorado inferior a $\epsilon = 10^{-9}$

Cálculo de n:
$|-\frac{h^2}{12} \times 3 \times 17.045| \leq 5 \cdot 10^{-9}$
$\iff h \leq 3.4 \cdot 10^{-5}$

$n = \frac{3}{3.4 \cdot 10^{-5}} = 88236$

0.3.2.5 Resultados

```
Com 5 casas decimais: 2.16623
Time: 0.000447
Com 7 casas decimais: 2.1696892
Time: 0.001567
Com 9 casas decimais: 2.168345151
Time: 0.013880
```

0.3.3 Regra de Simpson

main( ) in *SimpsonRule.c* (C)

```c
int main(){
  clock_t start = clock();
  printf("Com 5 Casas decimais: %.5f\n",simpson_rule(0.0, 3.0, 96.0));
  clock_t end = clock();
  float seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);

  start = clock();
  printf("Com 7 Casas decimais: %.7f\n",simpson_rule(0.0, 3.0, 310.0));
  end = clock();
  seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);

  start = clock();
  printf("Com 9 Casas decimais: %.9f\n",simpson_rule(0.0, 3.0, 968.0));
  end = clock();
  seconds = (float)(end-start)/CLOCKS_PER_SEC;
  printf("Time: %f\n",seconds);
}
```

0.3.3.1 Cálculo de M

$M = max|d''''(x)| = 3384.805$

0.3.3.2 Erro absoluto majorado inferior a $\epsilon = 10^{-5}$

Cálculo de n:
$|-\frac{h^4}{180} \times 3 \times 3384.805| \leq 5 \cdot 10^{-5}$
$\iff h \leq 0.031$

$n = \frac{3}{0.031} = 98$

0.3.3.3 Erro absoluto majorado inferior a $\epsilon = 10^{-7}$

Cálculo de n:
$|-\frac{h^4}{180} \times 3 \times 3384.805| \leq 5 \cdot 10^{-7}$
$\iff h \leq 0.0097$

$n = \frac{3}{0.0097} = 310$

0.3.3.4 Erro absoluto majorado inferior a $\epsilon = 10^{-9}$

Cálculo de n:
$|-\frac{h^4}{180} \times 3 \times 3384.805| \leq 5 \cdot 10^{-9}$
$\iff h \leq 0.0031$

$n = \frac{3}{0.0031} = 968$

0.3.3.5 Resultados

```
Com 5 casas decimais: 2.15546
Time: 0.000108
Com 7 casas decimais: 2.1635022
Time: 0.000056
Com 9 casas decimais: 2.167365353
Time: 0.000161
```

0.3.4 Terceiro Exercício

Comentem os resultados obtidos:

Comparando os resultados previamente expostos para as implementações de cada regra em causa, é possível concluir que a regra dos trapézios obteve, neste caso, valores mais precisos para I_f, notando-se que a regra dos retângulos foi a menos precisa.

No entanto, considerando a natureza de cada método, a regra de Simpson é geralmente mais precisa visto que usa sequências de parabólicas quadradas que permitem uma melhor aproximação à função f.

Por outro lado, as duas outras regras aqui abordadas recorrem a linhas retas de forma a obter diversas das suas figuras geométricas, respetivamente, retângulos e trapézios. Assim, estas regras raramente atingem aproximações mais precisas que a regra de Simpson dada a área que são obrigadas a ocupar entre o eixo das abcissas x e a curva da função f.

Considerando também o tempo necessário para obter o resultado usando cada uma das três regras, embora a regra de Simpson seja o método mais complexo, a sua facilidade em obter valores mais precisos acelera o processo para a função em causa.