Trabalho Prático nº 4

Análise Numérica

João Paulo Santos Monteiro

Nuno José Moreira Bernardes

Sérgio Alexandre Viveiros Araújo

Tiago André Carneiro Bessa

Maio 2020

Índice

1 Introdução

Neste quarto trabalho, usando o que aprendemos sobre integraçãao numérica, é-nos pedido para calcular valores aproximados de I em que será necessário usar três métodos abordados nas aulas: regra dos retângulos, regra dos trapézios e regra de Simpson, com majorantes de erros inferiores aos fornecidos,reforçando positivamente o nosso conhecimento sobre o capítulo estudado.

2 Desenvolvimento

2.1 Prefácio

Começámos o desenvolvimento do nosso programa por definir as funções e as variáveis que seriam comuns a todos métodos. A função a ser analisada, f(x) = sin(cos(sin(cos(x2)))), foi traduzida para a linguagem escolhida e foram adicionadas como variaveis de ambiente, funções da biblioteca math.

Também foi definido o intervalo [a,b], fornecido no enunciado como variáveis globais. De seguida, escrevemos a função do cálculo do valor de n, dada uma amplitude h, uma vez que esta também é comum as todas as regras a serem desenvolvidas. Por fim, definida a base do nosso programa, passámos para o desenvolvimento especifico de cada regra.

```
POW = math.pow
SIN = math.sin
COS = math.cos
a = 0
b = 3

def calcN( h ):
    return (b-a)/h

def f(x):
    return SIN(COS(SIN(POW(x,2))))

```

2.2 Regra dos retângulos

2.2.1 Código

A Regra dos retângulos permite o cálculo de uma aproximação para $\int_a^b f(x)dx$ através da soma da área de um número n de rectângulos, onde é definida por $I_f \approx R_n = h\sum_{i=1}^{n} f(x_{i-1})$.

A fórmula que define o cálculo desta regra é composta por duas partes lógicas na perspectiva de programação. Primeiro um somatório, definido por um ciclo que começa em i e termina em n, calculando para cada iteraçaõ o valor de x :

```
def somatorio( i, n, h):
    cur = a + h*i
    sum = 0
    while( i <= n ):
        sum += f(cur)
        cur += h
        i+=1
    return sum

```

Uma segunda parte de cálculos diretos, usando o somátorio anterior, conseguimos simular a regra pretendida, atráves da seguinte função:

```
    def rectangulos(h):
        n = calcN(h)
        return h * somatorio(0, n-1, h)
```

2.2.2 Cálculo do valor de h

$$E_n^R \leq \frac{b-a}{2} h f'(t), a < t < b$$

Através da fórmula descrita acima é-nos possível calcular o valor de h mediante o ϵ dado. Sendo $f'(t) = 1.26038$ os valores de h , segundo o software Wolframalpha, estão na tabela seguinte:

Epsilon(ϵ)	Valor de h
10^{-5}	$4.8*10^{-6}$
10^{-7}	$4.8*10^{-8}$
10^{-9}	$4.8*10^{-10}$

2.3 Regra dos trapézios

2.3.1 Código

A regra dos trapézios é definida por $I_f \approx T_n = h/2*(f(a)+f(b))+\sum_{i=1}^{n-1} f(x_i)$ e tal como a regra dos retângulos é constituída por um somatório e uma parte de matemática direta. Assim sendo, conseguimos reutilizar o nosso código do somatório definido na regra anterior, escrevendo a regra através da função:

```
def trapezios(h):
    n = calcN(h)
    return h/2 * ( f(a) + f(b) ) + h * somatorio(1, n-1, h)
```

2.3.2 Cálculo do valor de h

$$E_n^T \leq -\frac{h^2}{12}(b-a)f''(t), a < t < b$$

Através da formula descrita acima é nos possivel calcular o valor de h mediante o ϵ dado. Sendo $f''(x) = -17,045$ os valores de h, segundo o software Wolframalpha, estão na tabela seguinte:

Epsilon(ϵ)	Valor de h
10^{-5}	$1.0*10^{-3}$
10^{-7}	$1.0*10^{-4}$
10^{-9}	$1.0*10^{-5}$

2.4 Regra de simpson

2.4.1 Código

A regra de simpson, definida por $S_n = \frac{h}{3}(f(a) + 4f(x_1) + ... + 2f(x_{n-2}) + 4f(x_{n-1}) + f(b))$, apresentou um desafio mais complexo, uma vez que o seu somatório anterna o multiplicador entre 2 e 4. Por conseguinte, resolvemos este desafio, adaptando o somatório anterior e adicionando um booleano, que alterna e indica em que iteração é suposto ser um valor ou outro:

```
def somatorio_simpson( i, n, h):
    cur = a + h*i
    sum = 0
    dois = False
    while( i <= n ):
        sum += ( 2 if dois else 4 ) * f(cur)
        dois = not(dois)
        cur += h
        i+=1
    return sum

```

Usando o somátorio anterior, conseguimos simular a Regra de Simpson atráves da seguinte função:

```
def simpson (h):
    n = calcN(h)
    return h/3 *( f(a) + somatorio_simpson(1,n-1, h) + f(b) )

```

2.4.2 Cálculo do valor de h

$$E_n^S \leq -\frac{h^4}{180}(b-a)f^4(t), a < t < b$$

Através da formula descrita acima é nos possivel calcular o valor de h mediante o ϵ dado. Sendo $f^4(x) = 2.96243$ segundo o software Wolframalpha, h poderá tomar qualquer valor, assim sendo os valores escolhidos foram os seguintes :

Epsilon(ϵ)	Valor de h
10^{-5}	1.0*10^{-4}
10^{-7}	3.0*10^{-3}
10^{-9}	1.0*10^{-4}

3 Resultados

3.1 Regra dos retângulos

Erro	Valor aproximado
10^{-5}	2.11119424828177
10^{-7}	2.111194133871715
10^{-9}	Inconclusivo...

3.2 Regra dos trapézios

Erro	Valor aproximado
10^{-5}	2.111194243849217
10^{-7}	2.111194133871715
10^{-9}	2.111194132773304

3.3 Regra de simpson

Erro	Valor aproximado
10^{-5}	2.111194132760847
10^{-7}	2.1111941326997927
10^{-9}	2.111194132760847

3.4 Análise dos resultados

Comparativamente com $\int_0^3 sin(cos(sin(cos(x^2))))dx \approx 2.12685$ e os valores aproximados que obtivemos nas diferentes regras, podemos afirmar que obtivemos valores ligeiramente inferiores ao esperado inicialmente. Primeiramente, na regra dos retângulos, ao calcularmos para os dois primeiros erros obtivemos valores parcialmente semelhantes. Já para o último valor, não foi possível obter um resultado, visto que a amplitude é demasiado pequena e o programa necessita de muito tempo de modo a apresentar uma resposta. Nas outras regras, podemos analisar que todas as aproximações são relativamente semelhantes, obtendo o esperado.

Em termos de eficiencia algorítmica concluimos que a regra do rectângulos, apresenta uma convergencia mais lenta aquando comparada com as outras duas .Por outro lado não conseguimos distinguir a olho humano qualquer diferença de eficiencia temporal entre a regra de simpson e a regra dos trapézios. Quando consideramos a amplitude necessária para atingir o erro, conseguimos justificar a eficiencia dos algoritmos. Uma vez que a regra dos rectangulos necessita de uma amplitude muito pequena e os restantes não.

4 Conclusão

Este trabalho, comparativamente ao três passados, trouxe uma dificuldade de elaboração um pouco inferior para o nosso grupo, pois as ferramentas que conhecemos são suficientes para o que é pedido. Decidimos fazer o exercício em Python, pois sentimos mais à vontade em usar a linguagem e o Wolframalpha para ajudar nos cálculos necessários para o relatório. Apesar de tudo, acreditámos que os valores que calculámos estão corretos e que obtivemos os resultados esperados.

5 Bibliografia

1. Slides da professora Maria João Rodrigues - Análise Numérica 2019/2020;
2. Software Wolframalpha (cálculo das derivadas, máximos e mínimos no intervalo dado).