INSTITUTO POLITÉCNICO DE LISBOA
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Matemática Aplicada à Tecnologia e à Empresa

Introdução à Análise Numérica - 2023/2024

Número 50844. Nome Daniel Nóbrega [1]
Número 50849. Nome Afonso Santos [2]

[1] A50844@alunos.isel.pt
[2] A50849@alunos.isel.pt

1 Primeira Parte

Face à tarefa atribuida , vamos aproximar o valor da área da região plana limitada pelos gráficos de f e g, sendo que estas duas funções são diferenciáveis em $\mathbb{R}$ e que se intersetam pelo menos duas vezes, implementamos a seguinte solução, computacionalmente. Inicialmente, definimos a função f , g e depois h, sendo $h = |f - g|$.

De seguida criamos um código para desenhar o gráfico das funções f, g e h de forma a observar onde se intersetam as duas funções f e g e onde h se interseta com o eixo do x, ou seja, $h(x) = 0$.

Neste código, tem de se adaptar o gráfico conforme as funções para se conseguir observar os pontos de interseção.
Definimos a função Bisseção, esta função realiza o método da bisseção computacionalmente , este método é um algoritmo de busca de raízes que divide repetidamente um intervalo ao meio e seleciona o subintervalo no qual uma mudança de sinal ocorre. Continua esse proceso até que a solução desejada seja inferior ou igual ao erro definido. O erro utilizado no nosso trabalho é 10^-2, é um erro grande, pois temos como objetivo demostrar que quanto maior o erro maior será a diferença entre os resultados obtidos e o valor exato. Este código tem de ser usado duas vezes pois as funções f e g intersetam-se em dois pontos. Os intervalos atribuídos são com base na observação do gráfico das funções. Escolhemos este método por ser o mais simples e exato em termos de cálculos.

Com este método vamos obter um a e um b que serão os limites do integral para calcular a área. Fizemos então uma função que realiza o método de Simpson com base em a e b. Este é uma técnica de integração numérica usada para encontrar a área aproximada sobre uma curva. Escolhemos este por ser dos mais exatos e, porque temos de integrar polinómios de grau superior ou igual a 3. A área será então o módulo deste resultado.

Para finalizar criámos uma última função que nos permitisse obter os zeros exatos de $h(x)$, neste caso os pontos de interseção de f e g, calculando também a devida área para podermos comparar os resultados obtidos com aqueles que seriam suposto obter e perceber que matematicamente não foram muito próximos devido ao erro ser elevado.

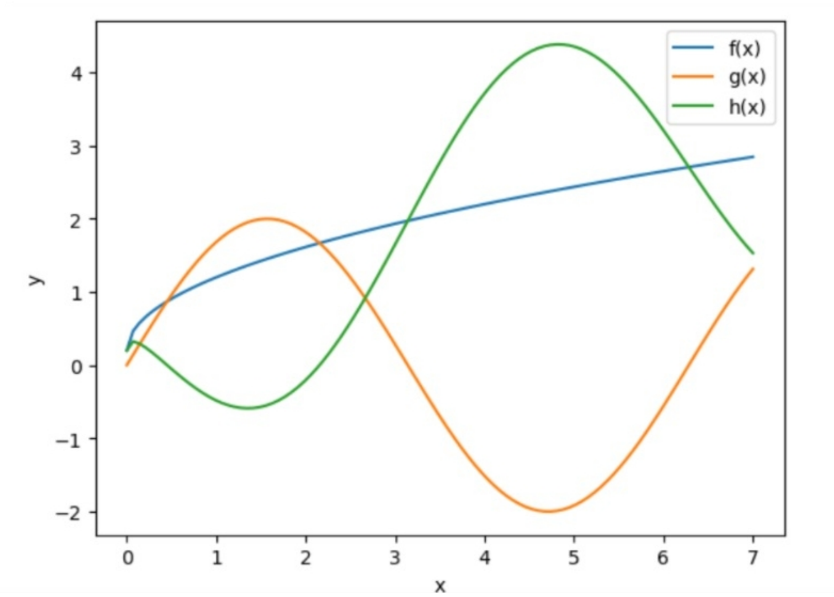
No âmbito do trabalho o professor pediu para corrermos dois exemplos pelo código, no qual um deles foi o próprio professor que disponibilizou.
Vamos começar pelo exemplo do professor:

Valor da área com base no método de Simpson:

$$f(x) = \sqrt{x} + 0.2$$

$$g(x) = 2\sin(x)$$

$$h(x) = \sqrt{x} + 0.2 - 2\sin(x)$$



Valores aproximados de a e b:

$$a = 0.4453125$$

$$b = 2.1484375$$

Valor da área com base no método de Simpson:

Valor do método de Simpson = -0.6671813813241539

A área será então o módulo do valor de Simpson obtido:

Área = 0.66718138132415391

Agora os valores exatos:

a(exato) = 0.45084255
b(exato) = 2.15521559

Valor da área com base no método de Simpson , porém com os valores certos nos limites do integral:

Valor do método de Simpson = -0.6672337376960409

A área será então o módulo do valor de Simpson com os valores certos nos limites do integral:

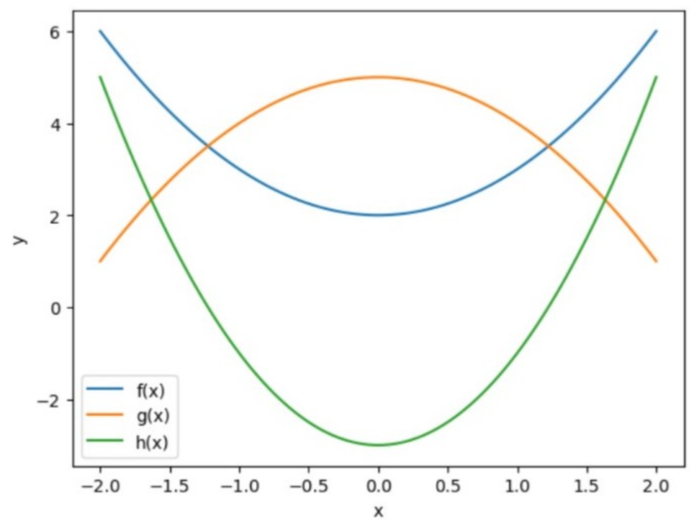
Área = 0.6672337376960409

Prosseguindo agora para o exemplo que nós criámos:

$$f(x) = x^2 + 2$$

$$g(x) = -x^2 + 5$$

$$h(x) = 2x^2 - 3$$



Valores aproximados de a e b:

$$a = -1.2265625$$

$$b = 1.2265625$$

Valor da área com base no método de Simpson:

Valor do método de Simpson =-4.8989793956279755

A área será então o módulo do valor de Simpson:

Área = 4.8989793956279755

Agora os valores exatos:

a(exato) = -1.22474487
b(exato) = 1.22474487

Valor da área com base no método de Simpson , porém com os valores certos nos limites do integral:

valor do método de Simpson = -4.898979485566356

A área será então o módulo do valor de simpson com os valores certos nos limites do integral:

$$\text{Área} = 4.898979485566356$$

Neste dois exemplos podemos observar que nos intervalos onde as funções se intersetam $f(x)$ é menor que $g(x)$ daí o valor do integral pelo método de Simpson ser negativo, visto que $h(x)$ é igual a $f(x) - g(x)$. Com base no nosso erro de $10^{-}2$, podemos concluir que os nossos resultados apresentam uma diferença matematicamente elevada em comparação aos resultados que deviam ser obtidos, pois quanto maior o erro, maior a disparidade dos resultados.

2 Segunda Parte

Face agora à segunda parte do trabalho, vamos implementar o Método dos Mínimos Quadrados computacionalmente para qualquer função de ajustamento , com uma qualquer base de dados, sendo ambas determinadas pelo utilizador do código. Deparamo-nos, inicialmente, com um problema, a resolução do sistema linear associado à implementação do MMQ. Por isso decidimos criar a função **gauss elimination** com o objetivo desta implementar o Método de eliminação de Gauss, também conhecido como eliminação gaussiana que é um procedimento utilizado na álgebra linear para resolver sistemas de equações lineares. O objetivo do método é transformar uma matriz aumentada de um sistema de equações lineares em uma forma escalonada por linhas (ou, idealmente, na forma escalonada reduzida por linhas), a partir da qual as soluções do sistema podem ser facilmente encontradas. Consiste em criar a matriz aumentada, que inclui tanto os coeficientes das variáveis quanto os termos constantes. A matriz aumentada é transformada através de operações elementares de linha, que incluem a troca de duas linhas de posição, a multiplicação de uma linha por um escalar diferente de zero e/ou a adição ou subtração de um múltiplo de uma linha a outra linha. Após isto ser realizado o objetivo é obter uma matriz identidade onde facilmente são obtidos os valores de cada variável.

O código que foi implementado utiliza o método de eliminação de Gauss e o método dos mínimos quadrados para determinar os coeficientes de uma função de ajustamento. Neste código usámos as funções componentes e a base de dados que o professor disponibilizou.

Com o problema da resolução do sistema linear resolvido, criamos agora a função **ajuste mmq** , que realiza o Método dos Mínimos Quadrados, usando a eliminação de gauss para a resolução do sistema linear , uma técnica usada para ajustar uma linha reta a um conjunto de dados, de modo que, em cada ponto, a linha esteja o mais próxima possível. Ele faz isso minimizando a soma das diferenças quadradas entre os valores observados e os valores previstos pela linha. Em outras palavras, o método encontra a linha que melhor representa a tendência dos dados, fazendo com que a soma das distâncias verticais entre os pontos e a linha seja a menor possível.

Para o programa poder realizar o MMQ é necessário introduzir uma base de dados, para conseguirmos abrir o ficheiro no jupyter é necessário este estar em arquivo CSV e os dados devem de estar identificados como o x e y sendo o y a imagem do x por base numa função, esta claro vem dos dados do utilizador do código.

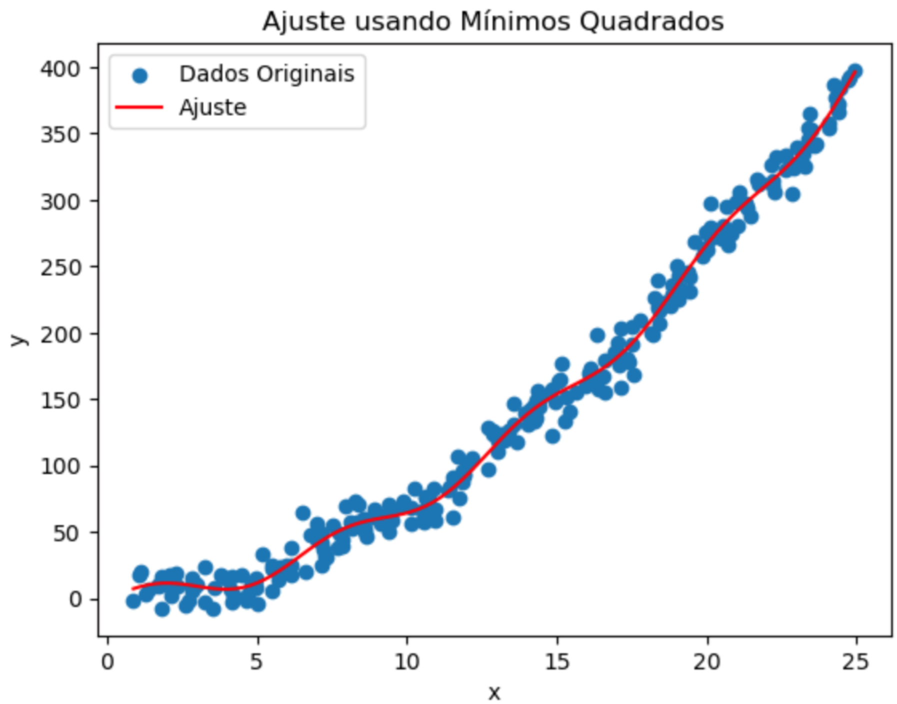
De seguida foi criada a variável **g funcs** , esta é onde o utilizador do código coloca a sua função de componentes à frente do lambda x. Podem ser colocadas as funções desejadas, sejam elas quais forem desde que no código sejam separadas por vírgulas e é feito o MMQ nas funções componentes obtendo a função ajuste e os coeficientes das funções componentes. Depois são calculados os y's ajustados .

Fizemos, de seguida, o plot dos dados onde colocamos a linha que define a função de ajustamento, fazendo com que apareça tudo no mesmo gráfico.

Finalizando é mostrado os coeficientes de ajustamento das funções componentes e o erro total. Essencialmente o y representa os valores reais dos dados originais e $y_{ajustado}$ representa os valores ajustados pelo modelo de mínimos quadrados. $(y - y_{ajustado})$ é uma operação de vetorização. Subtrai cada valor de $y_{ajustado}$ (valores previstos pelo modelo) pelo seu respectivo valor em y (valores reais dos dados originais). Isso dá-nos uma lista de diferenças entre os valores reais e os valores previstos pelo modelo. $(y - y_{ajustado})^2$ eleva ao quadrado cada uma dessas diferenças. Isso é feito para garantir que todas as diferenças sejam positivas e para enfatizar erros maiores, já que valores ao quadrado crescem mais rapidamente, np.sum(...) soma todas as diferenças ao quadrado. Isso representa a soma dos quadrados dos erros, que é uma medida comum do desempenho de um modelo de regressão. Quanto menor o valor de erro total, melhor o ajuste do modelo aos dados originais. Essencialmente, o erro total é uma medida do quão bem o modelo de ajuste se ajusta aos dados originais. Quanto menor o valor do erro total, melhor o modelo se ajusta aos dados.
Tal como na primeira parte do trabalho, o professor pediu nos para colocar dois exemplos no código, no qual um dos exemplos é dado pelo professor, tal como a base de dados. Vamos então começar por esse.

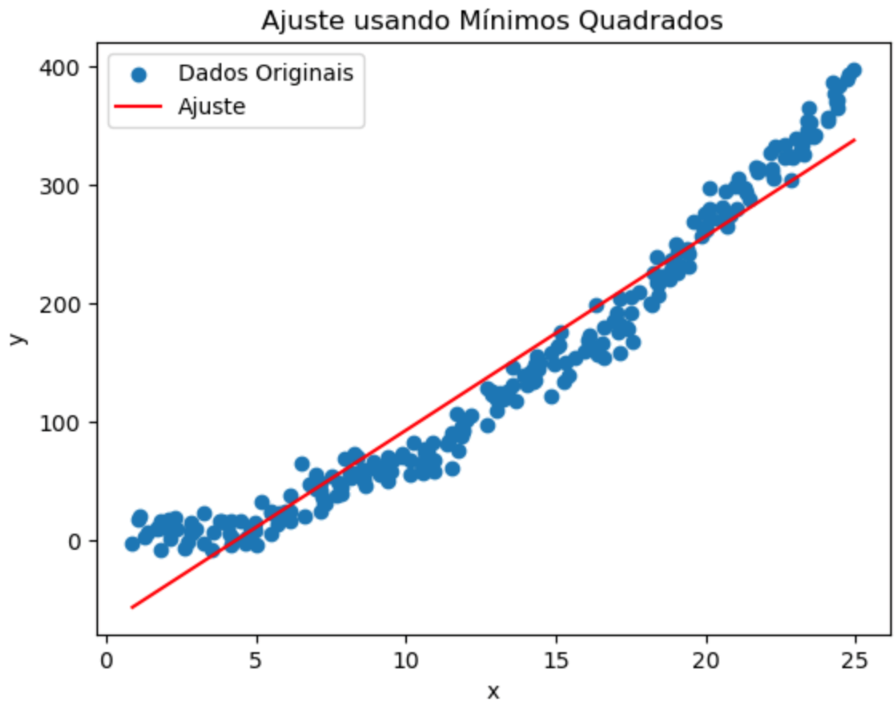
$$g(x) = a_0 \sin(x) + a_1 x + a_2 x^2 + a_3 \ln(x)$$

Valores dos coeficientes: [8.184838080218134, 0.6674366674695545, 0.6090383179372972, 0.5205612969352175]
Valor do Erro Total: 26584.942108982887




```
Coeficientes: [8.184838080218134, 0.6674366674695545, 0.6090383179372972, 0.5205612969352175]
Erro Total: 26584.942108982887
```

Segundo exemplo utilizando na mesma a base de dados fornecida pelo professor, mas alterando as funções componentes. Passamos a ter três em vez de quatro para provar que funciona para um qualquer x número de funções componentes.




```
Coeficientes: [-155.181994     28.39545069  106.27932961]
Erro Total: 198993.40161110822
```

Valores dos coeficientes: [-155.181994 28.39545069 106.27932961]
Valor do Erro Total: 198993.40161110822

Podemos comprovar que o valor do erro total é bastante significativo em ambos os exemplos, no entanto, no segundo caso, uma vez que todas as funções são de primeiro grau, isto faz com que a função ajuste seja menos eficiente, consequentemente, levando a um maior erro total.