**Roteiro**

Letra (a), na letra a temos que interpolar esses dois valores usando o programa PoliLagrange:

Antes vamos entender um pouco o Método de Lagrange:

Polinômio de Lagrange é o polinômio de interpolação de um conjunto de pontos na forma de Lagrange.

Dado um conjunto de *k*+1 pontos:

{\displaystyle (x\_{0},y\_{0}),\ldots ,(x\_{k},y\_{k})}

Com todos *xj* distintos, o polinômio de interpolação de um conjunto de pontos na forma de Lagrange é a [combinação linear](https://pt.wikipedia.org/wiki/Combina%C3%A7%C3%A3o_linear) dos polinômios da base de Lagrange: {\displaystyle L(x):=\sum \_{j=0}^{k}y\_{j}l\_{j}(x)}

Com polinômios da base de Lagrange dados pela seguinte expressão:

EXPLICAR O CÓDIGO NO MATLAB

Aplicando no Matlab

{\displaystyle l\_{j}(x):=\prod \_{i=0,j\neq i}^{k}{\frac {x-x\_{i}}{x\_{j}-x\_{i}}}={\frac {x-x\_{0}}{x\_{j}-x\_{0}}}\cdots {\frac {x-x\_{j-1}}{x\_{j}-x\_{j-1}}}{\frac {x-x\_{j+1}}{x\_{j}-x\_{j+1}}}\cdots {\frac {x-x\_{k}}{x\_{j}-x\_{k}}}}

De início temos a Interpolação usando a função que implementa o método de Lagrange para o primeiro ponto fornecido, mostrando os coeficientes, o polinômio interpolador e o seguinte resultado:

Agora, temos a Interpolação usando a função que implementa o método de Lagrange para o segundo ponto fornecido (onde os coeficientes e o polinômio interpolador são os mesmos da aplicação anterior). Então, temos o seguinte resultado:

Agora, um pequeno resumo dos resultados:

Letra (b), na letra b temos que interpolar também os mesmo dois valores usando agora o programa PoliNewton:

O código PoliNewton já foi explicado na questão anterior.

Aplicando no Matlab

Novamente, temos a interpolação usando a função agora que implementa o método de Newton para o primeiro ponto fornecido, mostrando os coeficientes, o polinômio e as diferenças divididas calculadas, logo em seguida o resultado:

Agora, temos a interpolação usando a função que implementa o método de Newton para o segundo ponto fornecido (onde os coeficientes, o polinômio interpolador e as diferenças divididas são os mesmos da aplicação anterior). Então, agora temos o seguinte resultado:

**Como esperado, no resumo vemos que o mesmo resultado foi obtido, o que mostra a consistência dos métodos.**

**Agora, aqui temos o gráfico do polinômio interpolador.**

Letra (c) Resolver com as funções do Matlab o ajuste de curvas e a interpolação, justificando a escolha e comparando com as letras (a) e (b).

Utilizando a função *interp1* do MATLAB®, os resultados obtidos foram os seguintes:

Essa é uma função nativa do MATLAB® que implementa, dentre outros, o método das splines.

Percebe-se que os resultados obtidos por esse método diferiram um pouco dos resultados encontrados anteriormente, mas ainda assim são muito próximos (com erro na ordem de 10-3). Com relação ao tempo de processamento, temos a seguinte tabela:

Vemos que a função *interp1* tem melhor tempo de processamento, e que a função *PoliNewton* executou em menor tempo que a função *PoliLagrange*, porém também deve-se levar em consideração que a função *interp1* exibe apenas o resultado da interpolação, enquanto que as demais funções também exibem outros detalhes e também o gráfico. Assim, já era esperado que estas levassem mais tempo para executar que aquela.

Demonstrando a questão no Matlab, estamos aplicando polilagrange, polinewton e a função nativa para o primeiro ponto pedido, o segundo pode-se fazer da mesma forma.