**SCRIPTS DOS MÉTODOS (Teste 1)**

**EGPP**

* Inversa = **inv(A)**
* Determinante = **det(A)**
* **x = A\b**

**Gaus Seidel**

* **1ª Condição**

2\*diag(abs(A))> sum(abs(A),2)

* **2ª Condição**

A == A'

e

det(A(1,1))>0

det(A(1:2,1:2))>0

det(A(1:3,1:3)) ou det(A)>0

* **3ª Condição**

D = diag(diag(A))

L = -tril(A,-1)

U = -triu(A,1)

então, C = inv(D-L)\*U ou C = (D-L)\U

norm(C,1)<1 ou norm(C,inf)<1

* **Iterações**

x2 = C\*x1+inv(D-L)\*b

norm(x2-x1,2)/norm(x2,2) <=

**Polinómio interpolador de Newton**

* x = [n+1 pontos];
* f = [imagens de x];
* pn = polyfit(x,f,n)
* val = polyval(pn, ponto interpolador)

**Splines**

**Splines do MatLab**

* x = [todos os pontos];
* f = [imagens de x];
* s3 = spline(x,f)
* s3.coefs
* val = spline (x, f, ponto interpolador)

**Splines cúbicas completas**

* d0 = (y0-y1)/(x0-x1)
* dn = (yn-1 – yn) / (xn-1 – xn)
* x = [todos os pontos menos x1 e xn-1];
* f = [imagens de x menos y1 e yn-1];
* s3c = spline(x, [d0 f dn])
* s3c.coefs
* val = spline (x, [d0 f dn], ponto interpolador)

**Mínimos quadrados**

**Modelo polinomial:**

* x = [todos os pontos]
* f = [imagens de x]
* [pn,r] = polyfit(x,f,n)
* Spn = r.normr^2
* val = polyval(pn, ponto interpolador)

**Modelo no polinomial:**

* Função:
  1. New -> function
  2. Nos parâmetros de saída colocar a letra que corresponde ao modelo (ex:M)
  3. Onde diz untilted colocar mqLETRA DO MODELO (ex: mqM)
  4. No primeiro parâmetro de entrada colocar c e no segundo parâmetro de entrada colocar x (dentro de parênteses, separados por vírgulas como lá esta)
  5. Por baixo do texto verde escrever a relação:

LETRA DO MODELO = c(1)\*/... + c(2)\*/... + c(n)\*/...

(pode ser mais menos, divisão, multiplicação...)

NÃO ESQUECER que os números dos coeficientes são sempre entre parênteses e se queremos dividir um escalar por um vetor x temos de por um ponto (1./x).

* 1. Deixa estar o ponto e vírgula, apagar a outra linha de condição e deixar estar o end.
  2. Gravar a função no save com o nome que colocamos no untilted.
* x = [todos os pontos]
* f = [imagens de x]
* [c,S] = lsqcurvefit (@o\_que\_colocamos\_no\_untilted , [1 1 1] , x , f)

Ter atenção ao nº de 1’s no vetor porque depende do nº de coeficientes a determinar e eles podem já nos dar um valor inicial. Daqui obtemos o S já diretamente.

* val = o\_que\_colocamos\_no\_untilted (c, ponto interpolador)