**- Sistemas lineares**

Ax = b

A\b

inv(x); det(x)

**- Sistemas de equações nao lineares**

**f(x) = 0 !!!**

x -> valor final (newton)

f -> valor da funcão f no ponto final x

exitflag -> >0 converge, = 0 pode convergir, <0 nao converge´

x1 -> valor inicial da primeira iteração

options = optimset('Jacobian', 'on', 'Tolx', E2, 'Tolfun', E1)

format long

[x, f, exitflag, output] = fsolve('funcao', x1)

function [f,d] = funcao ( x )

f = [x1; x2^2 + x2; exp(x3)-1]

if nargout > 1

j = [1 0 0; 0 (2\*x2+1) 0; 0 0 exp(x3)]

end

end

**-Polinomio interpolador**

grau N => N+1 pontos!!!

pn = polyfit(x,f,N) !!!! N -> grau do polinomio

pn = ...x^N + ....x^N-1 + ...K ESCREVER POLINOMIO

polyval(pn, K) valor do polinomio pn no ponto k

**-Splines**

USAR todos os pontos

s3 = spline (x,f)

s3.coefs

/\*surge lista de coeficientes.

segmento N escolher linha N\*/

sN3= ...\*(x-xN)^3 +...(x-xN)^2 +...\*(x-xN) + k onde N tem o valor do indice do segmento

sN3 = spline(x,f,k) -> valor da spline no ponto k do segmento N

spline(x, [d0 f dn]) quando deve ser natural/ dão derivada nos extremos !!!!!!!!!!!

**-Minimos quadrados**

Usar polyfit e polyval, mas com todos os pontos!!!!!

[P, s] = polyfit(x,f,N) --> N = grau do polinimio

Residuo = s^2

**-Integração numérica**

x = [x1 ... xn]

f = [f0 .... fn]

integral x1 a xn =

z = trapz(x,f)

[I, npts] = quad('funcao', limInf, limSup, erro/tolerância)

quad só quando se tem a expressão da funcao!!!!