Esercitazione 7 Maggio

Interpolazione e approssimazione

Routine nuove utilizzate:

`linspace max abs eval`

`plot figure subplot title`

`polyfit polyval interp1 spline interpft`

Comandi nuovi:

x(*end*) , `.^` , `.*`

Parte 1 - Plot e subplot

```
x=[0:0.01:1]; y=x.^2;
plot(x,y)
plot(x,y,'r')
plot(x,y,'.')
f='x.^2';
valutazf=eval(f);
max(valutazf-y)
x=[0:0.2:1]
valutazf=eval(f);
plot(x,valutazf,'.')
plot(x,valutazf)
plot(x,valutazf,'-x'), title('prova')
figure
subplot(2,2,1), plot(x,valutazf), title('semplice')
subplot(2,2,2), plot(x,valutazf,'--'), title('tratteggiato')
subplot(2,2,3), plot(x,valutazf,'.'), title('a punti')
```

```
subplot(2,2,4), plot(x,valutazf,'-x'), title('prova')
```

Parte 2 - Approssimazione di una funzione polinomiale. Valutazione della routine perturbando i dati, polinomio con e senza 'rumore':

```
x=[0:0.05:2]; %41 punti
y1=eval('3*x.^3-4.*x+1');
z=rand(size(x));
y2=z+y1;
plot(x,y1,'b',x,y2,'y')
figure
plot(x,y1,'b*',x,y2,'y.')

gradopol=3;

coefex1=polyfit(x,y1,gradopol);
coefex2=polyfit(x,y2,gradopol);

yapprox1=polyval(coefex1,x);
yapprox2=polyval(coefex2,x);

figure
subplot(2,3,1), plot(x,y1,'b',x,y2,'y'), title('le due funzioni')
subplot(2,3,2), plot(x,z), title('errore introdotto')

subplot(2,3,3), plot(x,yapprox1,'r',x,yapprox2,'g.'), title('i due polinomi')
subplot(2,3,4), plot(x,abs(yapprox1-yapprox2),'m'), title('diff tra polinomi')
```

```
subplot(2,3,5), plot(x,abs(y1-yapprox1),'m'), title('diff tra non perturb')
subplot(2,3,6), plot(x,abs(y2-yapprox2),'m'), title('diff tra perturb')
```

Parte 3 - Interpolazione di due funzioni oscillanti: fatto in aula

```
clear all
f1='x.*(1-x)+sin(20*pi*x)./30'
f2='sin(x*(2*pi))+sin(5*(x)*(2*pi))'
x=linspace(0,1,100);
plot(x,eval(f1))
figure
plot(x,eval(f2))

close all
x=linspace(0,1,3);
pp11=polyfit(x,eval(f1),2);
ff11=eval(f1);
pp21=polyfit(x,eval(f2),2);
x=linspace(0,1,200);
plot(x,polyval(pp11,x),'r',x,eval(f1),'y');
figure
plot(x,polyval(pp21,x),'r',x,eval(f2),'y');

close all
x=linspace(0,1,4);
ff12=eval(f1);
pp12=polyfit(x,eval(f1),3);
```

```
x=linspace(0,1,200);
plot(x,polyval(pp12,x),'r',linspace(0,1,4),ff12,'or',x,eval(f1),'y',...
    x,polyval(pp11,x),'--b',linspace(0,1,3),ff11,'xb');
figure

x=linspace(0,1,14);
ff13=eval(f1);
pp13=polyfit(x,eval(f1),13);
x=linspace(0,1,200);
plot(x,polyval(pp12,x),'r',linspace(0,1,4),ff12,'or',x,eval(f1),'y',...
    x,polyval(pp11,x),'--b',linspace(0,1,3),ff11,'xb',...
    x,polyval(pp13,x),'-.m',linspace(0,1,14),ff13,'xm');
figure

grad=8;%fare 6,7,8
x=linspace(0,1,grad);
ff2=eval(f2);
pp2=polyfit(x,eval(f2),grad-1);
x=linspace(0,1,200);
plot(x,polyval(pp2,x),'r',x,eval(f2),'y',linspace(0,1,grad), ff2, 'x');


%GRADO ALTISSIMO
x=linspace(0,1,100);
pp=polyfit(x,eval(f1),99);
%Warning: Polynomial is badly conditioned. Remove repeated data points
%         or try centering and scaling as described in HELP POLYFIT.
```

```
%> In C:\matlabR12\toolbox\matlab\polyfun\polyfit.m at line 74
x=linspace(0,1,200);
plot(x,polyval(pp,x)), axis([0 1 0 10])
hold on, plot(x,eval(f1),'r')
axis([0 1 0 1])


%CHEBY
n=20; %fare 10,20
x=0.5+0.5*(-cos(pi*[0:n]/(n) ) );
xx=x;
fc1=eval(f1);
fc2=eval(f2);
pc1=polyfit(x,eval(f1),n);
pc2=polyfit(x,eval(f2),n);
x=linspace(0,1,200);
plot(x,polyval(pc1,x),'r',x,eval(f1),'y',xx,fc1,'o');
figure
plot(x,polyval(pc2,x),'r',x,eval(f2),'y',xx,fc2,'o');


%SPLINE
x=linspace(0,1,100);
yy = INTERP1(x,eval(f1),linspace(0,1,200),'spline');
plot(x,eval(f1),'b',linspace(0,1,200),yy,'m');

x=linspace(0,1,100);
```

```
yy = INTERP1(x,eval(f2),linspace(0,1,200),'spline');
plot(x,eval(f2),'b',linspace(0,1,200),yy,'m');


%Intepolazione trigonometrica
x=linspace(0,1,15); %fare 3,5,15,55
x=x(1:end-1);
xx=x;
y=eval(f1);
z=interpft(y,100);
x=linspace(0,1,100);
plot(x,z,'m',x,eval(f1),'b',xx,y,'xr')

dim=5;%fare 3,5,15
x=linspace(0,1,dim);
x=x(1:end-1);
xx=x;
y=eval(f2);
z=interpft(y,100);
x=linspace(0,1,100);
plot(x,z,'m',x,eval(f2),'b',xx,y,'xr')
```

Parte 4 - Esempio aggiuntivo: funzione di Runge e nodi di Chebyshev

La seguente procedura genera polinomi interpolanti di grado sempre maggiore in punti equispaziati e di Chebychev, andando poi a memorizzare l'errore commesso in un vettore:

```
clear all
```

```
close all
rungef='1./(1+x.^2)';
app=[5:40];
for j=app
    i=j-app(1)+1;
    pti=j;
    grado=j-1;
    x=linspace(-5,5,pti);
    y=eval(rungef);
    x2=linspace(-5,5,200);
    y2=polyval(polyfit(x,y,grado),x2);
    x=x2;
    ErrEq(i)=max(abs(eval(rungef)-y2));
    clear x x2 y y2
    a=-5;
    b=5;
    for ii=1:pti+1
        x(ii)=(a+b)/2 -((b-a)/2)*cos(pi*(ii-1)/(pti));
    end
    y=eval(rungef);
    x2=linspace(-5,5,200);
    y2=polyval(polyfit(x,y,grado),x2);
    x=x2;
    ErrChb(i)=max(abs(eval(rungef)-y2));
    clear x x2 y y2
end
```

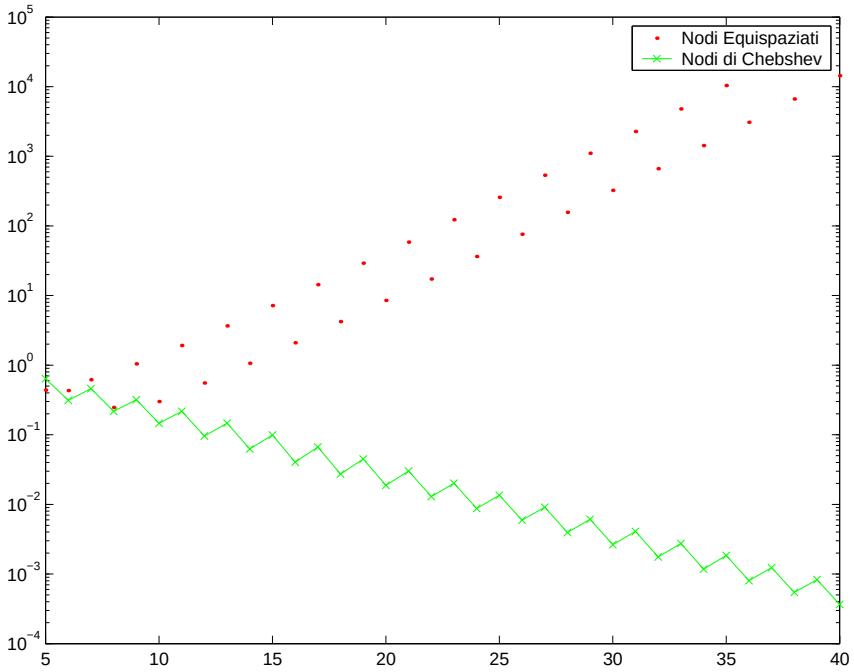

Figura 1: Errori commessi

Come viene valutato l'errore?

Il comando:

```
semilogy(app,ErrEq,'r.',app,ErrChb,'-gx'),
legend('Nodi Equispaziati','Nodi di Chebshev')
```

genera il plot in figura: **Si valuti l'andamento dell'errore**, considerando attentamente il modo in cui si è generato il plot (fare `help semilogy`).