- Non si possono consultare libri, note, ed ogni altro materiale o persone durante l'esame ad eccezione delle funzioni Matlab fornite.
- Risolvere i seguenti esercizi con l'ausilio di Matlab.
- La durata del compito è di 90 minuti.
- Questo esame ha 3 domande, per un totale di 30/30 punti.
- Svolgere gli esercizi su fogli protocollo, indicando: nome, cognome, codice persona e data
- Per ciascun esercizio consegnare su webeep un file nominato, ad esempio, "esercizio1.m" con il codice Matlab sviluppato (caricare le function fornite per l'esame solo se modificate).
- Per utilizzare le funzioni Matlab sviluppate durante il corso e fornite per l'esame, è necessario aggiungere la cartella con il comando `addpath functions2023`.

Esercizio 1 (punti 10)

Consideriamo la seguente funzione definita sull'intervallo $(-1, 1)$.

$$f(x) = \frac{1}{1 + 25x^2}$$

(a) (3 punti) [M] Approssimare f mediante interpolazione polinomiale Lagrangiana su nodi equispaziati di grado $n = [5, 10, 15]$. Riportare graficamente i polinomi interpolanti ottenuti sovrapposti alla funzione f e l'errore in spazio; stampare a schermo il massimo dell'errore nei tre casi,

$$err_n = \max_{x \in (-1,1)} |f(x) - \Pi_n f(x)|$$

Cosa osserviamo?

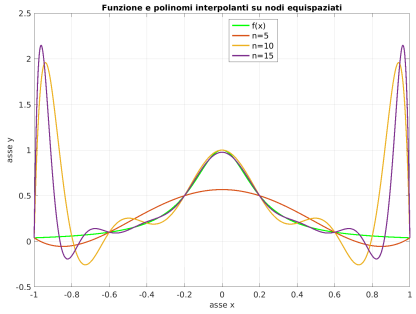
Soluzione. Possiamo costruire l'interpolazione polinomiale Lagrangiana su nodi equispaziati utilizzando il seguente script

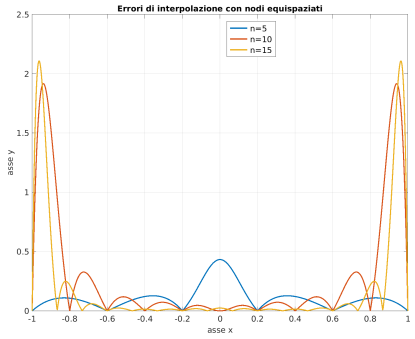
```
clear
close all
clc

%% PUNTO 1
a = -1;  b = 1;
```

```
x_dis = linspace( a, b, 1000 );
fun = @(x) 1./(1+25*x.^2);
f_dis = fun(x_dis);

grado = [5 10 15];
PP_dis = [];
EE_dis = [];
err_max = [];

fig = figure();
plot( x_dis, f_dis,'g','Linewidth',2)
title ('Funzione e polinomi interpolanti su nodi
   equispaziati')
xlabel('asse x')
ylabel('asse y')
grid
hold on

for n = grado
    x_nod = linspace(a,b,n+1);
    f_nod = fun(x_nod);
    P = polyfit( x_nod, f_nod, n );
    poly_dis = polyval( P, x_dis );
    err_dis = abs( poly_dis - f_dis );
    err_max = [err_max; max( err_dis )];
    PP_dis = [ PP_dis; poly_dis ];
    EE_dis = [ EE_dis; err_dis ];
end

plot(x_dis,PP_dis,'Linewidth',2)
legend('f(x)','n=5','n=10','n=15','Location','best')
fontsize(fig, 14, "points")
saveas(gcf, "es1_sol_a.png")

fig = figure();
plot(x_dis,EE_dis,'Linewidth',2)
title('Errori di interpolazione con nodi equispaziati')
legend('n=5','n=10','n=15','Location','best')
grid on
xlabel('asse x')
ylabel('asse y')
disp(err_max)
```

```
fontsize(fig, 14, "points")
saveas(gcf, "es1_sol_a_err.png")
```

Graficamente otteniamo





Notiamo che aumentando il grado di approssimazione l'interpolante risulta meno accurato agli estremi dell'intervallo $(-1, 1)$, tipico del fenomeno di Runge associato all'interpolazione polinomiale Lagrangiana su nodi equispaziati. Infine il massimo dell'errore che otteniamo è il seguente:

```
0.4327
1.9156
2.1069
```

(b) (4 punti) [T] Si discuta la stabilità e la convergenza dell'interpolazione polinomiale Lagrangiana su nodi equispaziati e su nodi di Chebychev-Gauss-Lobatto.

Soluzione. Data una funzione f definita su I e dati $n+1$ nodi x_i, considero la sua interpolazione Lagrangiana data da

$$\Pi_n f(x) = \sum_{i=0}^{n} f(x_i)\mathcal{L}_i(x) = \sum_{i=0}^{n} y_i \mathcal{L}_i(x),$$

dove Π_n è l'operatore di interpolazione che data una funzione f restituisce il polinomio interpolatore π_n negli $n+1$ punti x_i. Considero ora una funzione $\widetilde{f}$ ottenuta perturbando f: il suo interpolato è dato dalla seguente espressione

$$\Pi_n \widetilde{f}(x) = \sum_{i=0}^{n} \widetilde{f}(x_i)\mathcal{L}_i(x).$$

Possiamo calcolare la differenza tra l'interpolata di f e la sua perturbata per capire come si propagano le perturbazioni

$$\left\| \Pi_n f - \Pi_n \widetilde{f} \right\|_\infty \leq \max_{i=0,\dots,n} \left| f(x_i) - \widetilde{f}(x_i) \right| \max_{x \in I} \left| \sum_{i=0}^{n} \mathcal{L}_i(x) \right|$$

L'ultimo termine, che non dipende da f ma solo dai valori dei $\mathcal{L}_i$ nei nodi x_i, è detto costante di Lebesgue e, per nodi equispaziati, è data da

$$\Lambda_n = \max_{x\in I} \left| \sum_{i=0}^{n} \mathcal{L}_i(x) \right| \approx \frac{2^{n+1}}{en\log(n+\gamma)}$$

tale valore cresce molto all'aumentare di n rendendo quindi non stabile l'interpolazione polinomiale Lagrangiana su nodi equispaziati. Nel caso in cui considerassimo i nodi di Chebychev-Gauss-Lobatto tale costante risulterebbe

$$\Lambda_n < \frac{2}{\pi}\log n$$

ha una crescita di tipo logaritmico in n.

Per la convergenza abbiamo che nel caso di nodi uniformi non ci viene garantito che si abbia convergenza, ovvero in generale

$$\lim_{n\to\infty} \max_{x\in I} |E_n f(x)| \neq 0.$$

Mentre per nodi di Chebychev-Gauss-Lobatto il polinomio interpolatore $\Pi_n f$ è tale che $\Pi_n f \to f$ per $n \to \infty$, ovvero abbiamo convergenza all'aumentare del grado polinomiale.

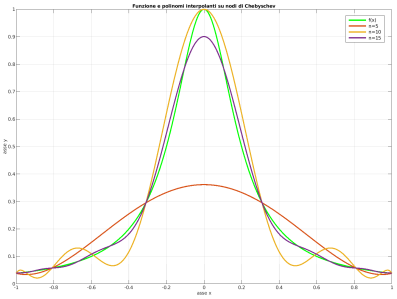
(c) (3 punti) [M] Ripetere quanto fatto al punto a per nodi di Chebychev-Gauss-Lobatto.

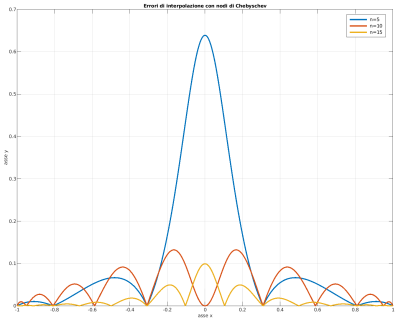
Soluzione. Possiamo costruire l'interpolazione polinomiale Lagrangiana su nodi di Chebychev-Gauss-Lobatto utilizzando il seguente script

```
%% PUNTO 3

fig = figure();
plot( x_dis, f_dis,'g','Linewidth',2);
title ('Funzione e polinomi interpolanti su nodi di
   Chebyschev')
xlabel('asse x')
ylabel('asse y')
grid
hold on

PP_dis = [];
EE_dis = [];
err_max = [];
```

```
for n = grado;
    k = [ 0 : n ];
    t = - cos( pi * k / n );
    x_nod = ( a + b ) / 2 + ( ( b - a ) / 2 )* t;
    f_nod = fun(x_nod);
    P = polyfit( x_nod, f_nod, n );
    poly_dis = polyval( P, x_dis );
    err_dis = abs( poly_dis - f_dis );
    err_max = [err_max; max( err_dis )];
    PP_dis = [ PP_dis; poly_dis ];
    EE_dis = [ EE_dis; err_dis ];
end

plot(x_dis,PP_dis,'Linewidth',2)
legend('f(x)','n=5','n=10','n=15','Location','best')
fontsize(fig, 8, "points")
saveas(gcf, "es1_sol_c.png")

fig = figure();
plot(x_dis,EE_dis,'Linewidth',2)
title('Errori di interpolazione con nodi di Chebyschev')
legend('n=5', 'n=10','n=15','Location','best')
grid on
xlabel('asse x')
ylabel('asse y')
disp(err_max)
fontsize(fig, 8, "points")
saveas(gcf, "es1_sol_c_err.png")
```

Graficamente otteniamo





Notiamo che aumentando il grado di approssimazione l'interpolante risulta più

accurata su tutto l'intervallo $(-1, 1)$, non abbiamo più il fenomeno di Runge associato all'interpolazione polinomiale Lagrangiana su nodi equispaziati. Infine il massimo dell'errore che otteniamo è il seguente:

```
0.6386
0.1322
0.0993
```

Esercizio 2 (punti 10)

Si consideri la seguente funzione

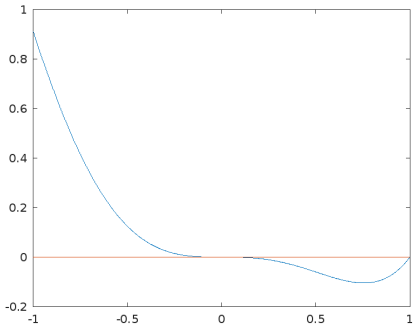
$$f(x) = x^3 \sin(x-1) \quad \text{per} \quad x \in (-1, 1)$$

(a) (1 punto) [M] Si rappresenti f in Matlab e si identifichi il valore α tale per cui $f(\alpha) = 0$.

Soluzione. Ecco un possibile script per rappresentare f

```
clear all
close all
clc

addpath functions2023

%% a)

f = @(x) x.^3.*sin(x-1);

a = -1;
b = 1;
x_val = linspace(a, b, 1000);
figure(1)
plot(x_val, f(x_val))
hold on
plot(x_val, 0*x_val)
hold off
```

Il cui grafico è il seguente



Possiamo notare che lo zero α della funzione f è situato in 0.

(b) (3 punti) [T] Derivare il metodo di Newton per la ricerca degli zeri di una funzione, riportando anche le sue proprietà di convergenza.

Soluzione. Dato, all'iterazione k, un valore di tentativo per lo zero esatto x^k, la retta tangente a f in x^k è data da

$$\frac{f(x) - f(x^k)}{x - x^k} = f'(x^k) \quad \Rightarrow \quad f(x) = f(x^k) + f'(x^k)(x - x^k).$$

Quindi dato x^k, il punto x^{k+1} è trovato come punto di intersezione della retta tangente con l'asse x ovvero lo zero della retta tangente approssimante f. Abbiamo che

$$f(x^{k+1}) = f(x^k) + f'(x^k)(x^{k+1} - x^k) = 0 \quad \Rightarrow \quad x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

Per la convergenza ci basiamo sul seguente risultato

Teorema 1. *Sia $f : [a, b] \to \mathbb{R}$ una funzione di classe C^2 in $[a, b]$. Sia α tale che $f(\alpha) = 0$ e $f'(\alpha) \neq 0$, ovvero di molteplicità algebrica pari a 1. Allora esiste $\eta > 0$ tale che se scelgo x^0 in modo che $|x^0 - \alpha| < \eta$ allora si ha*

$$\forall k \in \mathbb{N} \quad \left|x^k - \alpha\right| < \eta,$$

inoltre il metodo risulta convergente, ovvero

$$\lim_{k \to \infty} x^k = \alpha,$$

infine il metodo di Newton ha una convergenza quadratica, abbiamo

$$\lim_{k \to \infty} \frac{x^{k+1} - \alpha}{(x^k - \alpha)^2} = C = \frac{f''(\alpha)}{2f'(\alpha)}.$$

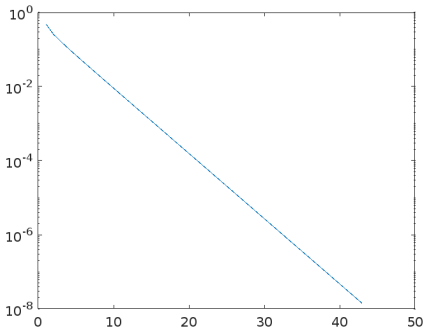
(c) (2 punti) [M] Applicare il metodo di Newton per il calcolo di α, partire da un valore iniziale pari a $x_0 = 0.5$ e impostare una tolleranza pari a 10^{-8}. Rappresentare in scala semilogy l'errore ottenuto e commentare alla luce della teoria.

Soluzione. Ecco un possibile script per il calcolo dello zero

```
x_ex=0;

%% b)

x0 = 0.5;
tol = 1e-8;
nmax = 1000;
```

```
df = @(x) 3*x.^2.*sin(x-1) + x.^3.*cos(x-1);

[xvect,it] = newton(x0,nmax,tol,f,df);

figure(2)
semilogy(abs(xvect-x_ex))
```

Il metodo si arresta dopo 42 iterazione e il valore finale calcolato è dato da 0.0000000139135. Assumendo $\alpha = 0$ la soluzione esatta, allora l'errore calcolato è rappresentato dal seguente grafico



Possiamo notare l'andamento lineare che indica una convergenza di tipo lineare.

(d) (2 punti) [T] Si proponga una modifica al metodo di Newton per il calcolo degli zeri a molteplicità algebrica maggiore di 1.

Soluzione. Introducendo m come molteplicità algebrica dello zero α, possiamo quindi presentare il metodo di Newton modificato come

$$x^{k+1} = x^k - m\frac{f(x^k)}{f'(x^k)}$$

Se tale metodo converge allora converge quadraticamente, come nel caso del metodo di Newton la convergenza è solo locale, cioè per un x^0 sufficientemente vicino al valore di α.

(e) (2 punti) [M] Si estenda opportunamente la function `newton.m` in modo da implementare quanto proposto al punto precedente. Utilizzando questa nuova funzione ripetere quanto fatto al punto c, sovrapponendo gli errori sullo stesso grafico. Commentare i risultati ottenuti.

Soluzione. La function `newtonmod.m` che implementa il metodo di Newton modificato è la seguente

```
function [xvect,it] = newtmod(x0,nmax,toll,fun,dfun,mol)
```

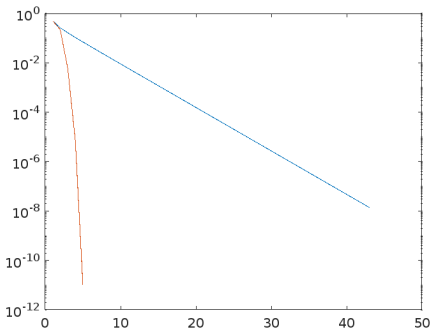
```
%
% [xvect,it] = newtmod(x0,nmax,toll,fun,dfun,mol)
%
% Metodo di Newton modificato per la ricerca degli zeri
% della funzione fun. Test d'arresto basato sul
   controllo
% della differenza tra due iterate successive.
%
% Parametri di ingresso:
%
% x0          Punto di partenza
% nmax        Numero massimo di iterazioni
% toll        Tolleranza sul test d'arresto
% fun dfun    Function handle contenenti la funzione e la
    sua derivata
% mol         Molteplicita' dello zero cercato
%
% Parametri di uscita:
%
% xvect       Vett. contenente tutte le iterate calcolate
%             (l'ultima componente e' la soluzione)
% it          Iterazioni effettuate


err = toll+1;
it = 0;
xvect = [x0];

while (it< nmax && err>= toll)
   xv = xvect(end);
   if abs(dfun(xv)) < eps % epsilon macchina - errore
      macchina
      disp(' Arresto per azzeramento di dfun');
      it = it + 1;
      break
   else
      xn = xv - mol * fun(xv) / dfun(xv);
      err = abs(xn - xv);
      xvect = [xvect; xn];
      it = it + 1;
   end
end
```

```
fprintf(' \n Numero di Iterazioni : %d \n',it);
fprintf(' Zero calcolato       : %-12.13f \n',xvect(end)
   );

return
```

Possiamo quindi calcolare lo zero con la nuova funzione nel seguente modo

```
saveas(gcf, "es2_sol_c.png")

%% c)
mol = 3;
[xvect_mod,it_mod] = newtmod(x0,nmax,tol,f,df,mol);

hold on
```

Il metodo si arresta dopo 5 iterazione e il valore finale calcolato è -0.0000000000103.
L'andamento dell'errore è ora dato dal seguente grafico



Notiamo come la nuova versione converge molto più rapidamente.

Esercizio 3 (punti 10)

Assumendo $c > 0$, si consideri la seguente equazione di conservazione

$$\begin{cases} \partial_t c + \partial_x(0.5c^2) = 0 & x \in (0,5),\ t \in (0,T] \\ c(0,x) = \begin{cases} 3 & x \le 2.5 \\ 5.5 - x & x > 2.5 \end{cases} & x \in (0,5) \\ c(t,0) = 0 & t \in (0,T] \end{cases}$$

dove il tempo finale è pari a $T = 1$.

(a) (3 punti) [T] Verificare se il flusso numerico Upwind è applicabile per la discretizzazione dell'equazione proposta con il metodo dei volumi finiti. Scegliere il valore $h = 0.125$ costante per l'ampiezza delle celle. Calcolare quindi il massimo Δt ammesso e chiamare Δt_{max} tale valore.

Soluzione. Nel caso in esame abbiamo $f(c) = \frac{1}{2}c^2$ e $f'(c) = c$. In base ai dati iniziali forniti il valore minimo e massimo di c sono rispettivamente $c_m = c(5) = 0.5$, $c_M = 3$ e, per tali valori, la derivata prima della funzione flusso è sempre non negativa qiundi il flusso upwind è applicabile. Il massimo della derivata $f'(c)$ si ottiene in corrispondenza di $c = 3$ e vale $\max_{c\in(c_m,c_M)} |f'(c)| = 3$. Per soddisfare la condizione CFL dobbiamo quindi garantire che $\Delta t \max_{c\in(c_m,c_M)} |f'(c)| < h$. Con i valori scelti si ottiene $\Delta t_{max} = 0.125/3 = 0.041\overline{6}$.

(b) (3 punti) [M] Sia $N = T/\Delta t_{max}$, risolvere il problema utilizzando la function `fvsolve` utilizzando il metodo di Upwind per N, $N-2$ e $2N$ passi temporali (attenzione: calcolare i Δt corrispondenti!) e rappresentare le soluzioni ottenute usando la function `xtplot`. Commentare cosa si osserva.

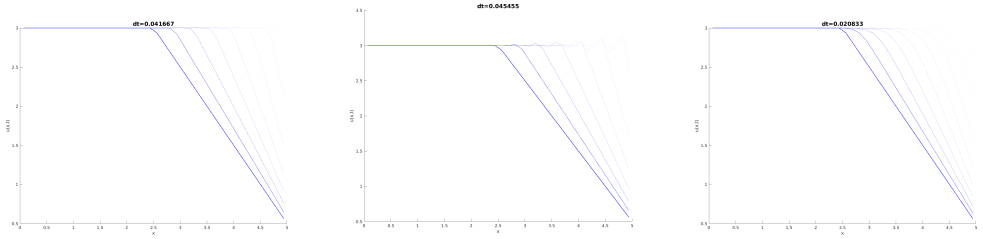
Soluzione. Il numero di intervalli N si ottiene come $N = 1/0.125 = 8$. Definiamo i dati necessari per la soluzione numerica,

```
h=0.125;
dtlim=h/df(3);
N=T/dtlim;

cont=1;
for n=[N, N-2, N*2]
 [xc, t, u] = fvsolve(u0, f, df, L, T, h, T/n, 'UPWIND')
    ;
 figure
 xtplot(xc,t,u,'animation')
 pause
```

Impostiamo quindi un ciclo per calcolare la soluzione con il numero di intervalli richiesto:

```
cont=1;
for n=[N, N-2, N*2]
 [xc, t, u] = fvsolve(u0, f, df, L, T, h, T/n, 'UPWIND')
    ;
 figure
 xtplot(xc,t,u,'animation')
```

ottenendo i tre grafici riportati:



Osserviamo che per il passo temporale "limite" la soluzione è corretta e si osserva la formazione di uno shock; per un passo temporale leggermente superiore al massimo consentito dalla CFL si ha un andamento non fisico; per passi temporali minori si ha una soluzione corretta, con una maggiore diffusine numerica.

(c) (2 punti) [T] Si descriva il flusso numerico di Godunov e se ne discutano le proprietà.

Soluzione. L'idea per introdurre il flusso di Godunov è quella di partire da una soluzione costante a tratti al tempo t_n, e di risolvere problemi di Riemann ad ogni interfaccia fra due celle vicine. Per calcolare il flusso di Godunov fra le celle i e $i+1$ consideriamo che, a seconda dei valori di c_i^n e c_{i+1}^n e della "forma" del flusso, possiamo avere uno shock, una rarefazione o una combinazione dei due; questo determina il valore della soluzione c^* nel nodo $i+\frac{1}{2}$ nell'intervallo di tempo $[t_n, t_{n+1}]$. Utilizzando c^* possiamo valutare il flusso numerico all'interfaccia.
Il flusso di Godunov risultante si può esprimere in modo sintetico con la seguente definizione:

$$F^G_{i+\frac{1}{2}}(c_i, c_{i+1}) = \begin{cases} \min f(\xi),\ \xi \in [c_i, c_{i+1}] & \text{se } c_i \leq c_{i+1} \\ \max f(\xi),\ \xi \in [c_{i+1}, c_i] & \text{se } c_i \geq c_{i+1} \end{cases}$$

Osserviamo che se f è monotono questa definizione coincide con il flusso upwind.

(d) (2 punti) [M] Si calcoli la soluzione del problema proposto utilizzando ora il flusso numerico di Godunov e un opportuno passo temporale.

Soluzione. Scegliamo ad esempio un passo $\Delta t = 0.9\Delta t_{max}$:

```
 title(strcat('dt=',num2str(T/n)),'FontSize',14)
saveas(gcf, strcat("es3_sol_", num2str(cont),".png"))
cont=cont+1;
```

La soluzione ottenuta è simile a quella ottenuta con il flusso upwind.