- 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.
- 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)

Si consideri la seguente funzione

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

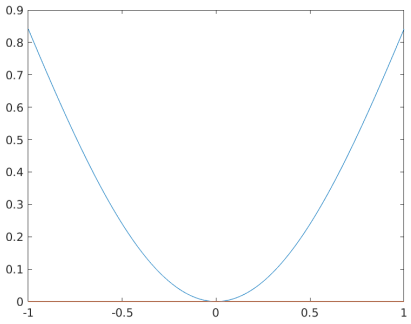
(a) (1 punto) [M] Si rappresenti f in Matlab e si identifichi il valore α tale per cui $f(\alpha) = 0$.

Soluzione. Ecco un possibile script per rappresentare f

```
clear all
close all
clc

addpath functions2023

%% a)

f = @(x) x.*sin(x);

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

```
plot(x_val, 0*x_val)
hold off

saveas(gcf, "es1_sol_a.png")
```

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

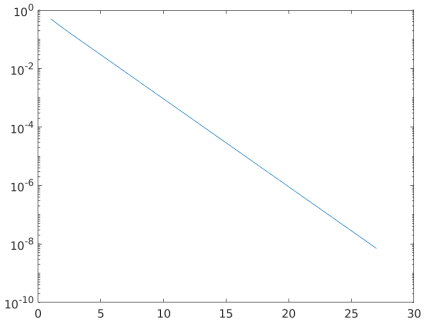
$$\lim_{k\to\infty} \frac{x^{k+1}-\alpha}{(x^k-\alpha)^2} = C = \frac{f''(\alpha)}{2f'(\alpha)}.$$

(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

```
%% b)

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

df = @(x) sin(x) + x*cos(x);

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

figure(1)
semilogy(abs(xvect))

saveas(gcf, "es1_sol_c.png")
```

Il metodo si arresta dopo 26 iterazione e il valore finale calcolato è dato da 0.0000000070302. 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

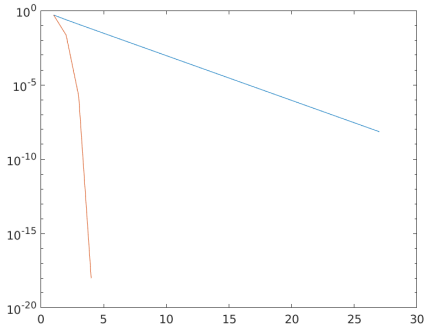
```
function [xvect,it] = newtmod(x0,nmax,toll,fun,dfun,
   mol)
%
% [xvect,it] = newtmod(x0,nmax,toll,fun,dfun,mol)
%
% Metodo di Newton modificato per la ricerca degli
   zeri
% della funzione fun. Test d'arresto basato sul
   controllo
% della differenza tra due iterate successive.
%
% Parametri di 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

```
%% c)
mol = 2;
[xvect_mod,it_mod] = newtmod(x0,nmax,tol,f,df,mol);

hold on
semilogy(abs(xvect_mod))

saveas(gcf, "es1_sol_e.png")
```

Il metodo si arresta dopo 4 iterazione e il valore finale calcolato è 0. L'andamento dell'errore è ora dato dal seguente grafico



Notiamo come la nuova versione converge molto più rapidamente.

Esercizio 2 (punti 10)

Si consideri la seguente matrice

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 4 & 10 & 20 & 35 \\ 1 & 5 & 15 & 35 & 70 \end{bmatrix}$$

che si può costruire con il comando Matlab `pascal(n)` dove n è la dimensione della matrice.

(a) (3 punti) [M+T] Enunciare la condizione necessaria e sufficiente per l'esistenza e unicità della fattorizzazione LU e verificare (con opportuni comandi) che è soddisfatta per la matrice A di dimensione $n = 5$.

Soluzione. Data una matrice $A \in \mathbb{R}^{n\times n}$ non singolare la fattorizzazione LU esiste ed è unica se e solo se tutte le sotto-matrici principali A_i di ordine $i = 1 \dots n-1$ sono non singolari. Possiamo calcolare il determinante delle sottomatrici con il seguente ciclo:

```
close all
clear all
clc
addpath functions2023

%% a)
n=5;
A=pascal(n);

for i=1:n
   D(i)=det(A(1:i,1:i));
end

if (all(D~=0))
    disp('la fattorizzazione LU (senza pivolting)
       esiste')
end
```

Chiedendo che tutti i valori salvati nel vettore D siano diversi da zero verifichiamo che tutte le sotto-matrici principali e la matrice A stessa siano non-singolari (condizione verificata).

(b) (3 punti) [M] Data la soluzione esatta $\boldsymbol{x} = [1, 1 \dots, 1]^T$ costruire il termine noto $\boldsymbol{b}$ e risolvere il sistema lineare $A\boldsymbol{x} = \boldsymbol{b}$ utilizzando i) la fattorizzazione LU ii) i metodi di sostituzione in avanti e all'indietro implementati nelle funzioni fornire. Verificare se è stato effettuato il pivoting.

Soluzione. Definita la soluzione esatta il termine noto si calcola come $\boldsymbol{b} = A\boldsymbol{x}$:

```
%% b)

x=ones(n,1);
b=A*x;
[L,U,P]=lu(A);
if any(any(P-eye(n)))
    disp('il pivoting e stato effettuato')
end
y=fwsub(L,P*b);
xn=bksub(U,y)
```

Ottenendo i seguenti risultati

```
la fattorizzazione LU (senza pivolting) esiste
il pivoting e stato effettuato
xn =
     1
     1
     1
     1
     1
```

In cui abbiamo effettuato la fattorizzazione chiedendo in output la matrice di permutazione; confrontandola con la matrice identità si pu'ødedurre che il pivoting, anche se non necessario, è stato effettuato. Infine, abbiamo risolviamo i problemi triangolari, ottenendo la soluzione numerica $\boldsymbol{x}_n$. In questo caso la soluzione numerica è identica alla soluzione esatta.

(c) (4 punti) [M+T] Ripetere i passaggi al punto precedente per matrici di dimensione $n = 10$, 15 e 20. Per ognuno dei casi calcolare la norma dell'errore *relativo* e il condizionamento della matrice e rappresentarli su due grafici in scala logaritmica. Commentare i risultati alla luce della teoria.

Soluzione. Ripetiamo i passi precedenti per diversi valori di n:

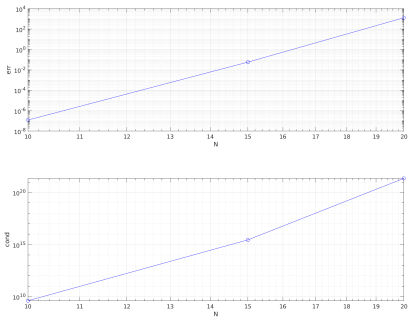
```
%% c)

N=[10 15 20];
err=[];
K=[];

for n=N
    A=pascal(n);
```

```
    x=ones(n,1);
    b=A*x;
    [L,U,P]=lu(A);
    y=fwsub(L,P*b);
    xn=bksub(U,y);
    err=[err norm(x-xn)/norm(x)];
    K=[K cond(A)];
end
err
K
subplot(2,1,1)
loglog(N,err,'bo-')
grid on
xlabel('N')
ylabel('err')
subplot(2,1,2)
loglog(N,K,'bo-')
grid on
xlabel('N')
ylabel('cond')
saveas(gcf, "es2_sol_c.png")
```

Ottenendo i seguenti risultati

```
err =
   1.0e+03 *
    0.0000    0.0001    1.2977
K =
   1.0e+21 *
    0.0000    0.0000    2.2465
```

Ed il seguente grafico



Dal grafico ottenuto osserviamo che per matrici di dimensioni maggiori l'errore cresce fino a superare il 100% e questo si spiega con l'aumento del numero di

condizionamento della matrice $K(A) > 10^{20}$. Non è quindi garantito che gli errori di arrotondamento commessi durante la fattorizzazione e il calcolo della soluzione vengono tenuti bassi, infatti in questo caso sono amplificati inquinando la soluzione numerica.

Esercizio 3 (punti 10)

Si consideri il seguente problema già affrontato nella parte 1 dell'esame

$$\begin{cases} \partial_t u - \partial_{xx} u = 1 & 0 < x < \pi,\ t > 0 \\ u(t,0) = u(t,\pi) = 0 & t > 0 \\ u(0,x) = 0 & 0 < x < \pi. \end{cases}$$

(a) (5 punti) [T] Introdurre brevemente l'approssimazione del problema con il metodo degli elementi finiti in spazio, e un generico theta-metodo in tempo. Derivarne l'espressione matriciale (senza dettagliare i singoli elementi delle matrici).

Soluzione. Consideriamo lo spazio funzionale $V = H_0^1(\Omega)$ e prendiamo delle funzioni test $v \in V$ che sono indipendenti da t; la forma debole del problema definita dall'equazione del calore è data da: trovare, per ogni $t \in (0,T)$, $u(x,t) \in V$ tale che

$$\int_a^b \partial_t u v dx + \int_a^b \partial_x u \partial_x v dx = \int_a^b v dx \qquad \forall v \in V.$$

dove $a = 0$ e $b = \pi$. La formulazione discreta del problema è derivata fissando il tempo ed operando prima in spazio. Scegliendo $V_h \subset V$ sotto-spazio finito dimensionale otteniamo che la semi-discretizzazione in spazio è: trovare, per ogni $t \in (0,T)$, $u_h(t) \in V_h$ tale che

$$\int_a^b \partial_t u_h v_h dx + \int_a^b \partial_x u_h \partial_x v_h dx = \int_a^b v_h dx \qquad \forall v_h \in V_h.$$

Introduciamo ora le forme a e m e il funzionale F definiti come

$$a(u_h, v_h) = \int_a^b \partial_x u_h \partial_x v_h \quad \text{e} \quad m(u_h, v_h) = \int_a^b u_h v_h \quad \text{e} \quad F(v_h) = \int_a^b v_h dx$$

il precedente problema può quindi essere scritto come:

$$m(\partial_t u_h(t), v_h) + a(u_h, v_h) = F(v_h) \qquad \forall v_h \in V_h.$$

Anche in questo caso consideriamo una base $\{\phi_j\}$ per V_h di funzioni linearmente indipendenti, consideriamo lo sviluppo di v_h e u_h sulla base $\{\phi_j\}$ di V_h otteniamo

$$v_h(x) = \sum_{i=1}^{N_h} \alpha_i \phi_i(x) \quad \text{e} \quad u_h(x,t) = \sum_{j=1}^{N_h} u_j(t) \phi_j(x)$$

Sfruttando la bi-linearità della forma a, il fatto che il dominio non dipende dal tempo e la linearità del funzionale F otteniamo la seguente espressione

$$\partial_t m\left(\sum_{j=1}^{N_h} u_j(t)\phi_j, \phi_i\right) + a\left(\sum_{j=1}^{N_h} u_j(t)\phi_j, \phi_i\right) = F(\phi_i) \qquad \forall i = 1, \dots, N_h$$

$$\sum_{j=1}^{N_h} \partial_t u_j(t) m(\phi_j, \phi_i) + \sum_{j=1}^{N_h} u_j(t) a(\phi_j, \phi_i) = F(\phi_i) \qquad \forall i = 1, \dots, N_h,$$

introduciamo la matrice di rigidezza A e di massa M date rispettivamente da

$$A \in \mathbb{R}^{N_h \times N_h}: \quad a_{ij} = a(\phi_j, \phi_i)$$
$$M \in \mathbb{R}^{N_h \times N_h}: \quad m_{ij} = m(\phi_j, \phi_i)$$

Il problema semi-discreto risulta quindi di determinare il vettore tempo dipendente $\boldsymbol{u}(t)$ tale che

$$M d_t \boldsymbol{u}(t) + A\boldsymbol{u}(t) = \boldsymbol{f}.$$

(b) (3 punti) [M] Si consideri una griglia di ampiezza uniforme $h = \frac{\pi}{20}$. Si risolva il problema con il metodo di Eulero Esplicito, usando un passo temporale $\Delta t = 0.25$, quindi si ripeta il calcolo con $\Delta t = 0.0025$. Si utilizzi la function `heatsolve` fornita. Si rappresentino le soluzioni utilizzando la function `xtplot` fornite e si commenti il risultato alla luce della teoria.

Soluzione. Definiamo innanzitutto i dati del problema e della discretizzazione spaziale:

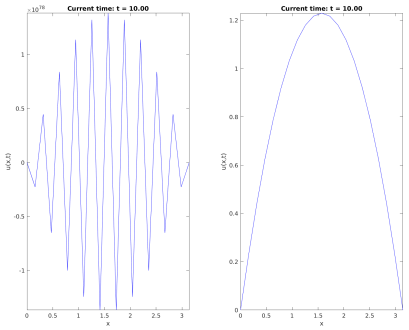
```
f=@(x,t) 1+x*0;
u0=@(x) 0*x;
L=pi;
T=10;
D=1;
N=20;
h=L/N;
```

Quindi scegliamo $\theta = 0$ (Eulero Esplicito) e risolviamo usando il primo Δt richiesto:

```
%% a)
%soluzione con Eulero implicito
theta=0;
%primo passo temporale
dt=.25;
[x, t, u_EE] = heatsolve(D, f, L, h, u0, T, dt, theta)
   ;
figure(1)
subplot(1,2,1)
xtplot(x,t,u_EE,'animation')

%secondo passo temporale
dt=.0025;
```

```
[x, t, u_EE] = heatsolve(D, f, L, h, u0, T, dt, theta)
   ;
subplot(1,2,2)
xtplot(x,t,u_EE,'animation')

saveas(gcf, "es3_sol_b.png")
```

Il cui grafico è il seguente



Notiamo che per $\Delta t = 0.25$ la soluzione risulta instabile e presenta oscillazione, mentre riducendolo a $\Delta t = 0.0025$ tale fenomeno non accede.

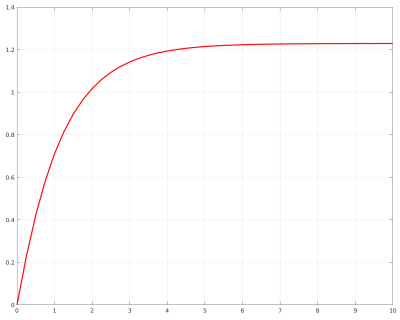
(c) (2 punti) [M] Si risolva ora il problema con il metodo di Eulero Implicito, scegliendo $\Delta t = 0.25$. Si rappresenti il valore della soluzione numerica al centro del dominio, in tempo: $u_h(t, \frac{\pi}{2})$ (da estrarre con un metodo a scelta). Cosa si osserva?

Soluzione. Scegliamo $\theta = 1$ (Eulero Implicito) e risolviamo usando il primo Δt richiesto:

```
%% b)
%soluzione con Eulero implicito
dt=0.25;
theta=1;
[x, t, u_EI] = heatsolve(D, f, L, h, u0, T, dt, theta)
   ;
figure(2)
plot(t, u_EI(N/2+1,:),'linewidth',2,'color','r')
grid on
saveas(gcf, "es3_sol_c1.png")

%metodo alternativo
u_middle=interp1(x,u_EI,pi/2)
figure(3)
plot(t, u_middle,'linewidth',2,'color','r')
```

```
grid on

saveas(gcf, "es3_sol_c2.png")
```

Il grafico che otteniamo per la soluzione al centro dell'intervallo è il seguente



Notiamo che la soluzione non presenta oscillazioni anche se il $\Delta t = 0.25$ nel caso di Eulero Esplicito non funzionava. Inoltre, possiamo notare che la soluzione tende ad uno stato stazionario con l'avanzare del tempo. La soluzione analitica stazionaria è infatti data da $u(x) = \frac{1}{2}x(\pi - x)$.