- 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)

Consideriamo la seguente funzione definita sull'intervallo $[-1, 1]$.

$$f(x) = e^{-\frac{1}{x^2}}$$

(a) (3 punti) [M] Approssimare f mediante interpolazione polinomiale Lagrangiana su nodi equispaziati di grado $n = [5, 10, 20]$. 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?

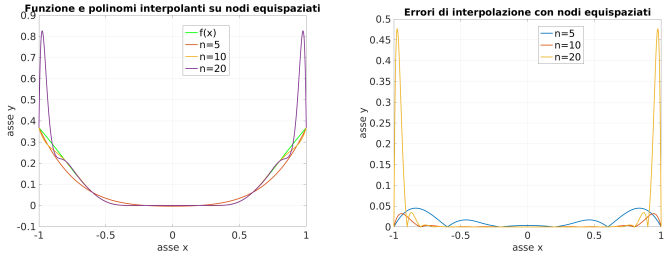
Soluzione. Possiamo costruire l'interpolazione polinomiale Lagrangiana su nodi equispaziati utilizzando il seguente script

```
clear
close all
clc

%% PUNTO 1
a = -1;  b = 1;
x_dis = linspace( a, b, 1000 );
```

```
fun = @(x) exp(-1./(x.^2));
f_dis = fun(x_dis);

grado = [5 10 20];
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=20','Location','best')
fontsize(fig, 24, "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=20','Location','best')
grid on
xlabel('asse x')
ylabel('asse y')
disp(err_max)
fontsize(fig, 24, "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:

```
0.0452
0.0319
0.4766
```

(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.

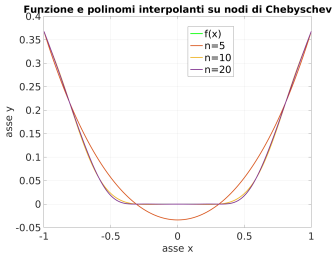
(c) (3 punti) [M] Ripetere quanto fatto al punto a per nodi di Chebychev-Gauss-Lobatto.

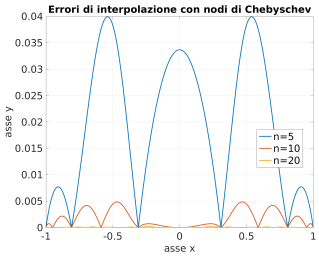
Soluzione. Possiamo costruire l'interpolazione polinomiale Lagrangiana su nodi di Chebychev-Gauss-Lobatto utilizzando il seguente script

```
%% 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=20','Location','best')
fontsize(fig, 24, "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=20','Location','best')
grid on
xlabel('asse x')
ylabel('asse y')
disp(err_max)
fontsize(fig, 24, "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:

```
0.0399
0.0049
0.0002
```

Esercizio 2 (punti 10)

Dato $n = 20$, si vuole risolvere il sistema lineare $A\boldsymbol{x} = \boldsymbol{b}$ dove A è tridiagonale, con 2 sulla diagonale principale e -1 sulle sotto/sopra-diagonali, mentre $\boldsymbol{b}$ è un vettore di 1. Entrambi potete generarli con i seguenti comandi

```
A = full(gallery('tridiag',n,-1,2,-1));
b = ones(n, 1);
```

Nei seguenti punti si fissi una tolleranza per il criterio di arresto pari a $1e-8$, un numero massimo di iterazioni pari a 1000 e il valore iniziale $x_0 = [0, \ldots, 0]^\top$.

(a) (4 punti) [T] Si introduca un generico metodo iterativo basato sulla matrice di iterazione B per la risoluzione di un sistema lineare. Si riporti e dimostri il risultato sulla condizione necessaria e sufficiente di convergenza per tale metodo.

Soluzione. Un metodo iterativo costruisce una successione $\{\boldsymbol{x}^k\}$, dove ogni $\boldsymbol{x}^k \in \mathbb{R}^n$, che converge alla soluzione $\boldsymbol{x}$ per $k \to \infty$. Abbiamo quindi

$$\boldsymbol{x}^0 \xrightarrow{\text{un passo}} \boldsymbol{x}^1 \xrightarrow{\text{un passo}} \boldsymbol{x}^2 \xrightarrow{\text{un passo}} \ldots \xrightarrow{\text{un passo}} \boldsymbol{x}^k \xrightarrow{k\to\infty} \boldsymbol{x}.$$

Una forma generale per i metodi iterativi è data dall'espressione seguente

$$\boldsymbol{x}^{k+1} = B\boldsymbol{x}^k + \boldsymbol{g},$$

dove $B \in \mathbb{R}^{n\times n}$ è detta matrice di iterazione, che generalmente può dipendere da A, e $\boldsymbol{g}$ un vettore che viene costruito partendo da A e $\boldsymbol{b}$. Il metodo iterativo è detto consistente se B e $\boldsymbol{g}$ sono tali che

$$\boldsymbol{x} = B\boldsymbol{x} + \boldsymbol{g},$$

ossia se la soluzione esatta soddisfa esattamente il metodo numerico.
Un metodo iterativo consistente della forma

$$\boldsymbol{x}^{k+1} = B\boldsymbol{x}^k + \boldsymbol{g},$$

è convergente se e solo se

$$\rho(B) < 1,$$

Mostriamo l'implicazione che se il raggio spettrale soddisfa $\rho(B) < 1$ allora il metodo è convergente. Essendo per ipotesi un metodo consistente, allora vale

$$\boldsymbol{e}^{k+1} = B\boldsymbol{e}^k,$$

inoltre, ripetendo il ragionamento per $k-1$, $k-2$ ... si ottiene che

$$\boldsymbol{e}^{k+1} = B^{k+1}\boldsymbol{e}^0.$$

Dato che $\rho(B) < 1$ sappiamo che $\lim_{k\to\infty} B^k = 0$, quindi otteniamo che

$$\lim_{k\to\infty} \boldsymbol{e}^{k+1} = \lim_{k\to\infty} B^{k+1}\mathbf{e}^0 = \mathbf{0}.$$

Mostriamo ora l'implicazione inversa, ovvero vogliamo mostrare che se il raggio spettrale è maggiore di 1 allora il metodo non è convergente. Supponiamo che $\rho(B) \geq 1$, cioè esiste almeno un autovalore λ di B tale che $|\lambda| \geq 1$. Scelgo $\boldsymbol{x}^0$ tale che l'errore iniziale, $\boldsymbol{e}^0 = \boldsymbol{x} - \boldsymbol{x}^0$, sia uguale all'autovettore associato a λ. Abbiamo quindi

$$B\boldsymbol{e}^0 = \lambda\boldsymbol{e}^0$$

e otteniamo che l'errore al passo k si esprime come

$$\boldsymbol{e}^k = B\boldsymbol{e}^{k-1} = B^2\boldsymbol{e}^{k-2} = \ldots = B^k\boldsymbol{e}^0. = \lambda^k\boldsymbol{e}^0$$

Quest'ultima espressione non può convergere a zero per $k \to \infty$ dato che $|\lambda| > 1$. Quindi il raggio spettrale della matrice di iterazione per un metodo convergente deve necessariamente essere minore di 1.

(b) (3 punti) [M+T] Si introduca la matrice di iterazione B_J di Jacobi, si verifichi che tale metodo soddisfa la condizione necessaria e sufficiente per la convergenza e si risolva il sistema lineare usando la funzione `jacobi.m`. Calcolando il residuo normalizzato su $\boldsymbol{b}$, cosa osserviamo?

Soluzione. Possiamo definire la matrice di iterazione B_J e il vettore $\boldsymbol{g}$ associati al metodo di Jacobi come

$$\boldsymbol{g} = D^{-1}\boldsymbol{b}. \quad \text{e} \quad B_J = D^{-1}(D - A) = I - D^{-1}A$$

dove $D = \text{diag}(A)$. Per verificare la convergenza e calcolare la soluzione corrispondente utilizziamo il seguente script

```
clear
close all
clc

addpath functions2023

n = 20;
A = full(gallery('tridiag',n,-1,2,-1));
b = ones(n, 1);

%% PUNTO 2
```

```
% definizione della matrice di iterazione di Jacobi
D = diag(diag(A));
Bj = eye(n) - D\A;

% test sulla convergenza
disp(max(abs(eig(Bj))))

% soluzione con Jacobi
nmax = 1000;
toll = 1e-8;
x0 = ones(n, 1);
[xj, kj] = jacobi(A, b, x0, toll, nmax);

% calcolo errore
err = norm(b - A*xj) / norm(b);
disp(kj)
disp(err)
```

Ottenendo il seguente risultato

```
0.9888
1000
1.1922e-05
```

Ovvero il raggio spettrale $\rho(B_j)$ risulta molto prossimo a 1 e quindi ci aspettiamo una convergenza piuttosto lenta. Otteniamo infatti che l'algoritmo non raggiunge la tolleranza specificata in 1000 iterazioni ma ne richiederebbe di più.

(c) (3 punti) [M+T] Si introduca la matrice di iterazione B_{GS} di Gauss-Seidel, si verifichi che tale metodo è convergente e si risolva il sistema lineare usando la funzione `gs.m`. Calcolando il residuo normalizzato su $\boldsymbol{b}$, cosa osserviamo?

Soluzione. Scomponiamo la matrice A come $A = D - E - F$ dove

$$D = \begin{bmatrix} a_{11} & 0 & \dots & & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ \dots & & & & \\ 0 & & \dots & 0 & a_{nn} \end{bmatrix} \quad E = \begin{bmatrix} 0 & 0 & \dots & & 0 \\ -a_{21} & 0 & 0 & \dots & 0 \\ \dots & & & & \\ -a_{n1} & & \dots & -a_{n,n-1} & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & -a_{12} & \dots & & -a_{1n} \\ 0 & 0 & -a_{23} & \dots & -a_{2n} \\ \dots & & & & \\ 0 & & \dots & 0 & 0 \end{bmatrix}.$$

La matrice di iterazione di Gauss-Seidel è data da

$$B_{GS} = (D - E)^{-1}F$$

Per verificare la convergenza e calcolare la soluzione corrispondente utilizziamo il seguente script

```
%% PUNTO 3

% definizione della matrice di iterazione di Gauss-
   Seidel
D = diag(diag(A));
E = -tril(A);
F = -triu(A);
Bgs = (D - E)\F;

% test sulla convergenza
disp(max(abs(eig(Bgs))))

% soluzione con Gauss-Seidel
[xgs, kgs] = gs(A, b, x0, toll, nmax);

% calcolo errore
err = norm(b - A*xgs) / norm(b);
disp(kgs)
disp(err)
```

Ottenendo il seguente risultato

```
0.5000
817
9.8676e-09
```

In questo caso il raggio spettrale è minore che per il metodo di Jacobi, ci aspettiamo una convergenza più rapida alla soluzione. Infatti l'algoritmo terma entro il numero massimo di iterazioni impostate. Tale risultato è in linea con quanto predetto dalla teoria.

Esercizio 3 (punti 10)

Si consideri il seguente problema

$$\begin{cases} -u'' = 2\pi^2 \sin(2\pi x) \quad 0 < x < 1, \\ u(0) = 0 \\ u'(1) = \pi \end{cases}$$

(a) (1 punto) [T] Determinare il valore di A tale per cui $u(x) = A\sin(2\pi x)$ è soluzione esatta del problema.

Soluzione. Verifichiamo innanzitutto che $-u'' = -4\pi^2 A(-\sin(Ax))$ è pari alla forzante data se $A = \frac{1}{2}$; inoltre $u(0) = 0\ \forall A$. Calcoliamo $u'(1)$ e verifichiamo se la condizione di Neumann è soddisfatta:

$$2\pi\frac{1}{2}\cos(2\pi) = \pi$$
$$\pi = \pi$$

(b) (4 punti) [T] Introdurre brevemente l'approssimazione del problema con il metodo degli elementi finiti in spazio e derivarne l'espressione matriciale considerando le condizioni al contorno del problema in esame.

Soluzione. Dato che abbiamo una condizione di Dirichlet nel bordo sinistro, in $x = 0$, consideriamo lo spazio funzionale $V = \{v \in H^1(\Omega) : v(0) = 0\}$ e prendiamo delle funzioni test $v \in V$; la forma debole del problema è: trovare $u(x) \in V$ tale che

$$a(u, v) = F(v) \qquad \forall v \in V.$$

dove $a(u, v) = \int_0^1 u'v'dx$ e $F(v) = \int_0^1 fv - gv(1)$. Nel nostro caso $f = 2\pi^2 \sin(2\pi x)$ e $g = \pi$. Notiamo che il termine di bordo è nullo sul bordo sinistro mentre la condizione di Neumann sul bordo destro è inclusa nel funzionale.

La formulazione discreta del problema è ottenuta scegliendo un sotto-spazio finito dimensionale $V_h \subset V$ per cui possiamo considerare una base $\{\phi_j\}$ per V_h di funzioni linearmente indipendenti. Nel nostro caso, data una griglia di $N+2$ nodi in cui $x_0 = 0$, $x_{N+1} = 1$, e spaziatura $h = \frac{b-a}{N+1}$ la base è formata dalle funzioni "a capanna" ϕ_j con $j = 1, \dots N+1$; infatti, considerando lo sviluppo

$$u_h(x) = \sum_{j=1}^{N_h} u_j \phi_j(x)$$

le incognite sono i valori u_j in tutti i nodi eccetto il primo, per un totale di $N+1$ incognite. Sfruttando la bi-linearità della forma a, il fatto che il dominio

non dipende dal tempo e la linearità del funzionale F otteniamo un sistema di $N+1$ equazioni

$$A\boldsymbol{u} = \boldsymbol{f}$$

dove la matrice di rigidezza A

$$A \in \mathbb{R}^{N+1 \times N+1}: \quad a_{ij} = a(\phi_j, \phi_i)$$

In particolare otteniamo la seguente struttura:

$$A = \frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ \dots & & & & & & \\ 0 & \dots & 0 & 0 & -1 & 2 & -1 \\ 0 & \dots & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

mentre il termine noto $\boldsymbol{f}$ contiene, nell'ultima riga, il contributo della condizione di Neumann.

(c) (2 punti) [M] Si consideri una griglia di ampiezza uniforme $h = \frac{1}{10}$. Si risolva il problema con il metodo degli elementi finiti lineari, utilizzando la function `dirNeusolve` fornita. Si rappresenti in un grafico la soluzione numerica sovrapposta alla soluzione esatta.

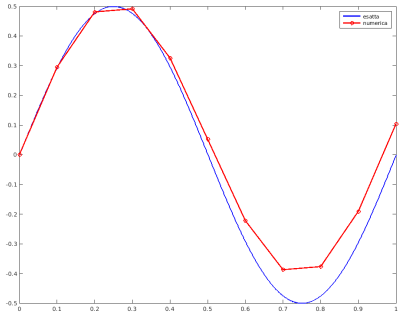
Soluzione. Definiamo i dati e rappresentiamo la soluzione esatta:

```
A = 1/2;
f = @(x) 2*pi*pi*sin(2*pi*x); % forzante
L = 1; % lunghezza intervallo
gN = pi; % condizione di Neumann in x = L

uex = @(x) A*sin(2*pi*x); % soluzione esatta

figure(1)
xplot = linspace(0, L, 1000);
plot(xplot, uex(xplot),'b-','linewidth',2)
hold on
```

Quindi risolviamo il problema numericamente con $h = 1/10$ e sovrapponiamo il grafico ottenuto:

```
[x, u] = dirNeusolve(f, L, gN, L/10);
plot(x, u, 'ro-','linewidth',2)
legend('esatta','numerica')
```

Ottenendo il seguente grafico



(d) (3 punti) [M+T] Si ripeta il calcolo della soluzione numerica per valori di al variare di h con $h = \frac{1}{N}$, $N = 20, 40, 80$. Calcolare l'errore e rappresentarne l'andamento in scala logaritmica. Commentare il risultato alla luce della teoria. (*Suggerimento*: per il calcolo dell'errore definire l'integranda e utilizzare la funzione `simpcomp` fornita.)

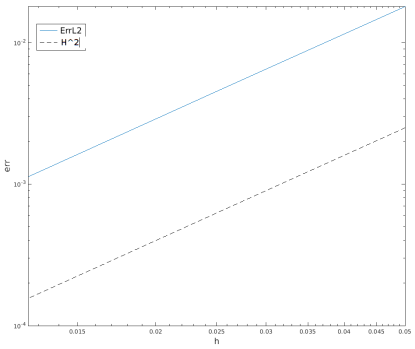
Soluzione. Definiamo un vettore per i passi d. griglia e un vettore, della stessa lunghezza, per il salvataggio dell'errore, quindi impostiamo un ciclo `for`

```
N = [20 40 80];
errL2 = 0*N;
for i = 1:length(N)
    n = N(i);
    h = L/n;
    [x, u] = dirNeusolve(f, L, gN, h);

    integrandaL2 = @(t) (uex(t) - interp1(x, u, t)).^2;
    errL2(i) = sqrt(simpcomp(0, L, 1000, integrandaL2));
end
```

Il calcolo dell'errore è effettuato definendo come funzione integranda il quadrato della differenza fra la soluzione esatta e quella numerica in un generico punto t e integrando numericamente con il metodo di Simpson. Per ottenere il grafico in scala logaritmica utilizziamo i seguenti comandi:

```
figure(2)
H = L./N;
```

```
loglog(H, errL2, H, H.^2, 'k--')
legend('ErrL2', 'H^2','FontSize',14)
xlabel('h','FontSize',14)
ylabel('err','FontSize',14)
```



Notiamo che l'errore diminuisce quadraticamente come previsto dalla teoria.