Esame di Calcolo Numerico

Nome :

Cognome :

Matricola :

Gli esercizi sono di due tipologie:

- (T) indica un esercizio riguardante gli argomenti teorici del corso: tale esercizio va svolto sul foglio che verrà poi consegnato a mano.
- (M) indica un esercizio da svolgere in ambiente MatLab: il codice verrà sviluppato a computer, la consegna verrà effettuata tramite il portale `esami.lamping.unife.it`. **Tutti gli M–files devono contenere nelle prime due righe (commentate) il proprio numero di matricola e subito sopra il proprio cognome (pena l'esclusione dall'appello).**

Gli esercizi denotati con (M+T) riguarderanno argomenti sia teorici che pratici: la parte teorica dovrà essere svolta sul foglio, mentre la parte di programmazione MatLab verrà consegnata tramite il portale.

Istruzioni per il download dei codici forniti

1. collegarsi al sito `esami.lamping.unife.it`
2. accedere con le proprie credenziali di ateneo
3. inserire il codice (ad esempio, 7) dell'esame che verrà fornito in sede di esame
4. nella finestra ALLEGATI cliccare sui file per eseguire il download

Istruzioni per la consegna

1. collegarsi al sito `esami.lamping.unife.it`
2. accedere con le proprie credenziali di ateneo
3. inserire il codice (ad esempio, 7) dell'esame che verrà fornito in sede di esame
4. cliccare sulla voce CONSEGNA
5. caricare i file

Per una maggiore sicurezza, provvedere all'upload dei propri codici ogni venti minuti.

L'esame ha una durata di 4 ore.

Esercizio 1

1. (T) (3 punti) Rappresentare in "fixed point" (b=2, t+1=16) il numero $(-528)_{10}$. Rappresentare il numero $(-2.725)_{10}$ come numero finito in semplice precisione (4 byte) secondo le convenzioni dell'Ansi standard IEEE.

2. (M) (3 punti) Usando la M-function di Matlab `polyval`, scrivere una M-function che esegua la conversione di un numero intero con segno scritto come stringa da base b a base 10. Si assuma che la base sia al più uguale a 16 e che la stringa sia costituita da una combinazione di al più dai seguenti caratteri, oltre a eventuale segno '+' o '-':

   ```
   cifre=['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'];
   ```

   L'intestazione della function deve essere

   ```
   function num=convb10(stringa,b);
   ```

Esercizio 2

Si consideri la seguente matrice quadrata A, cosí definita:

$$A = \begin{pmatrix} 5 & -1 & -2 \\ -1 & 3 & 1 \\ -2 & 1 & 4 \end{pmatrix}$$

1. (T) (3 punti) Verificare che la matrice sia definita positiva, calcolando la fattorizzazione di Cholesky. Usare la fattorizzazione per calcolare il determinante. Quale complessità ha teoricamente la fattorizzazione di Cholesky?

2. (M) (2 punti) In un M–file denominato `es2.m` si definisca la matrice A, si verifichi tramite la function di Matlab `chol` che il fattore calcolato al punto precedente sia corretto. Dato il termine noto $b = (2, 3, 3)^T$, determinare e stampare la soluzione del sistema $Ax = b$, usando le funzioni `sollower.m` e `solupper.m` in allegato.

3. (M) (3 punti) Calcolare e stampare l'inversa X di A mediante l'utilizzo della fattorizzazione e delle funzioni `sollower.m` e `solupper.m`; calcolare la norma infinito del residuo $AX - I$, dove I è la matrice identità di dimensione opportuna.

Esercizio 3

1. (M) (3 punti) Dato il sistema $Ax = b$ associato alla matrice

$$A = \begin{pmatrix} \frac{26}{5} & 1 & 1 & 1 & 1 \\ -1 & \frac{12}{5} & -1 & \frac{1}{10} & 0 \\ \frac{1}{10} & -1 & \frac{12}{5} & -1 & \frac{1}{10} \\ 0 & \frac{1}{10} & -1 & \frac{12}{5} & -1 \\ 0 & 0 & \frac{1}{10} & -1 & \frac{12}{5} \end{pmatrix}$$

   calcolare il vettore b in modo che $sol = (1, 1, 1, 1, 1)^T$ sia la soluzione esatta.
   Determinare la fattorizzazione di A data da $PA = LR$ mediante l'algoritmo di Gauss con pivoting parziale, utilizzando la function `gauss2` in allegato. Risolvere il sistema $Ax = b$ con mediante la fattorizzazione ottenuta, usando le functions `sollower.m` e `solupper.m` in allegato. Calcolare il residuo normalizzato e una stima del numero di condizione rispetto alla norma infiito.

2. (2 punti) Per quali classi di matrici si può eseguire la fattorizzazione $A = LR$ senza necessità di pivoting?

Esercizio 4

Si consideri la matrice tridiagonale A di ordine n data da

$$A = \begin{pmatrix} 2 & -\sin(n-1) & & & & & & \\ -\sin(n-1) & 4 & -\sin(n-2) & & & & & \\ & -\sin(n-2) & 6 & -\sin(n-3) & & & & \\ & & -\sin(n-3) & 8 & -\sin(n-4) & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & -\sin(3) & 2(n-2) & -\sin(2) & & \\ & & & & -\sin(2) & 2(n-1) & -\sin(1) & \\ & & & & & -\sin(1) & 2n \end{pmatrix}$$

con $n = 20$ e il sistema lineare $Ax = b$ dove b è scelto in modo che la soluzione esatta sia $sol = (1, 1, \ldots, 1)^T \in \mathbb{R}^{20}$.

1. (T) (3 punti) Con quali metodi iterativi è possibile risolvere il sistema? Motivare verificando le condizioni di convergenza. Specificare la matrice di iterazione di Iacobi.

2. (M+T) (2+2) Si realizzi un M-script file che costruisca la matrice A, memorizzandola secondo la modalità sparsa, ed il termine noto b. Si risolva il sistema tramite il metodo SOR (utilizzando la funzione `sor` in allegato), partendo da un vettore iniziale nullo, usando una tolleranza pari a $1e-6$, dopo aver determinato e stampato il valore di ω ottimale. Usare la function `eig` di Matlab per calcolare gli autovalori della matrice di Iacobi (ricordare che `eig` ha come argomento una matrice densa). Stampare il numero di iterazioni necessarie a ottenere la soluzione e l'errore commesso in norma 2.

Esercizio 5

Si consideri la funzione

$$f(x) = x + e^{\left(\frac{1}{1+x^2}\right)}$$

definita sull'intervallo $[-2, 2]$.

1. (M) (3 punti) Realizzare un M-script file che costruisca il grafico della funzione usando 100 punti nell'intervallo di definizione. Costruire mediante la function `pol_lagrange` in allegato i polinomi interpolanti di Lagrange $p_n(x)$ di grado $n = 6, 8, 10$ relativo ad una distribuzione di nodi equispaziati nell'intervallo e realizzare i grafici dei polinomi nella stessa finestra grafica, usando gli stessi 100 punti. Costruire in una seconda finestra grafica, i grafici degli errori nei 100 punti considerati per ogni polinomio. Specificare la legenda.

2. (T) (2 punti) Si fornisca la stima teorica dell'errore di interpolazione per una generica funzione $f \in \mathcal{C}^{(n+1)}([a, b])$. Data la funzione

   $$f(x) = \cos(x + 1), \quad x \in \left[0, \frac{\pi}{2}\right]$$

   calcolare il grado del polinomio affinchè l'errore commesso non sia superiore a 10^{-3} nel caso di nodi di Chebyshev.

Esercizio 6

1. (M) (3 punti) Scrivere un M–file denominato `es6.m` in cui:
   - usando la function di Matlab `polyfit` si determinano i coefficienti della retta che onora nel modo migliore possibile i seguenti punti del piano cartesiano $(1.0, 1.18)$, $(1.2, 1.26)$, $(1.4, 1.23)$, $(1.6, 1.37)$, $(1.8, 1.37)$, $(2.0, 1.45)$, $(2.2, 1.42)$, $(2.4, 1.46)$, $(2.6, 1.53)$, $(2.8, 1.59)$, $(3.0, 1.50)$;
   - si stampino i coefficienti del polinomio e la somma dei quadrati dei residui
   - si visualizzino le coppie di dati e, utilizzando opportunamente `polyval`, si costruisca nella stessa finestra grafica il grafico di tale polinomio tabulando l'intervallo con 100 punti equispaziati.
2. (T) (2 punto) Scrivere la marice di regressione lineare nel caso dell'esercizio precedente e la formula del sistema delle equazioni normali.

Esercizio 7

Si consideri la funzione $f(x) = (x-1)\cos(x)$ definita nell'intervallo $[0, \pi]$.

1. (M) (3 punti) Si rappresenti il grafico della funzione sull'intervallo $[0, \pi]$ e si determinino graficamente gli intervalli ove è possibile applicare il metodo di bisezione. Calcolare, ove possibile, le radici della funzione f con l'algoritmo di bisezione utilizzando la function `bisez`, con una tolleranza di 10^{-8}. Determinare per ogni radice il numero di iterazioni necessarie per calcolare l'approssimazione con la tolleranza prefissata.

2. (T) (2 punti) Descrivere il metodo della secante, scrivendo la sua iterazione generica e fornire la sua velocità di convergenza.

1 Allegati

```
function [L,R,P,deter]=gauss2(A);
% fattorizzazione di Gauss con pivoting parziale - II versione
[n]=size(A,1);
temp=zeros(1,n);
P=1:n;
deter=1;
tol=eps*norm(A,inf);
for k=1:n-1
   [amax,ind]=max(abs(A(k:n,k)));
   ind=ind+k-1;
   if k~= ind
      aux=P(k);
      P(k)=P(ind);
      P(ind)=aux;
      temp=A(ind,:);
      A(ind,:)=A(k,:);
      A(k,:)=temp;
      deter=-deter;
   end;
   deter=deter*A(k,k);
   if abs(A(k,k))>tol
      A(k+1:n,k)=A(k+1:n,k)/A(k,k);
      %  operazione di base: aggiornamento mediante diadi
      A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-A(k+1:n,k)*A(k,k+1:n);
   end;
end;
deter=deter*A(n,n);
R=zeros(n);L=eye(n);
R=triu(A);
L=L+tril(A(1:n,1:n),-1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 function [x,k]=sor(A,b,x, maxit,tol,omega);
 % metodo di sor - Gauss Seidell estrapolato
 n = max(size(A));

 for k=1:maxit
     xtemp=x;
     for i=1:n
         x(i)= (-A(i,[1:i-1, i+1:n])*x([1:i-1, i+1:n])+b(i))/A(i,i);
         x(i)=(1-omega)*xtemp(i)+omega*x(i);
     end;


     if norm(xtemp-x,inf)<tol*norm(x,inf)
```

```
        break;
    end;
end;
if k == maxit
    fprintf('convergenza non raggiunta\n');
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [p,coeff]=pol_lagrange(x,y,punti);
% x: vettore dei nodi o punti di osservazione
% y: vettore delle osservazioni
% punti: punti in cui calcolare il polinomio di Lagrange
%
n1=length(y); coeff=zeros(size(x)); p=zeros(size(punti));
for i=1:n1
    coeff(i)=y(i)/prod(x(i)-x([1:i-1,i+1:n1]));
end;
for k=1:length(punti)
    ij=find(punti(k)==x);
    if isempty(ij)
        % calcolo del polinomio di Lagrange
        temp=prod(punti(k)-x); %
        p(k)=temp*sum(coeff./(punti(k)-x));
    else
        p(k)=y(ij(1));
    end;
end;


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,it]=bisez(fname,a,b,tol,maxit);

fa = feval(fname,a);
fb = feval(fname,b);
if sign(fa)*sign(fb) >=0
    error('intervallo non corretto');
else
    % bisezione
    it = 0;
    while abs(b-a)>=tol+eps*max([abs(a) abs(b)]) & it<=maxit
        it = it+1;
        pm = a+(b-a)*0.5;
%         fprintf('it=%g x=%g\n',it,pm);
        fpm = feval(fname,pm);
        if fpm == 0
            break;
        end;
```

```
        if sign(fpm)*sign(fa) >0
            a  = pm;
            fa = fpm;
        else
            b  = pm;
            fb = fpm;
        end;
    end;
    if it>maxit
        error('Raggiunto max limite di iterazioni');
    else
        x = pm;
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%

function x = solupper(r,b);
% orientato per righe
n = length(b);
x = b;
x(n) = x(n)/r(n,n);
for i = n-1:-1:1
% SDOT
    x(i) = x(i)-r(i,i+1:n)*x(i+1:n);
    x(i) = x(i)/r(i,i);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%

function x = sollower(l,b);
% orientato per righe
n = length(l);
x = b;
x(1) = x(1)/l(1,1);
for i = 2:n
%SDOT
    x(i) = x(i)-l(i,1:i-1)*x(1:i-1);
    x(i) =x(i)/l(i,i);
end;
```