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) (2 punti) Convertire in numero decimale il seguente numero di macchina floating point, codificato secondo le convenzioni dell'Ansi standard IEEE, precisione semplice (4 byte):

   00111101100000001000001000000000

2. (M) (2 punti) Scrivere un M-script file Matlab che, dopo aver accettato un numero reale in base 10 e una base data da un intero positivo compreso tra 2 e 16, esegue la conversione nella base assegnata con k cifre nella parte frazionaria, usando le M-functions in allegato. k deve essere fornito in ingresso e occorre controllare la correttezza dei dati in ingresso (base data da intero compreso tra 2 e 16, k intero positivo).

Esercizio 2

Si consideri il sistema $Ax = b$, ove A è la matrice di Hilbert di ordine n generata con la function Matlab `hilb(n)` e b è un vettore di n elementi determinati in modo che la soluzione x del sistema sia il vettore $(1, 2, ..., n)$. Si ricorda che le matrici di Hilbert sono definite positive.

1. (T) (2 punti) Si enunci la condizione necessaria e sufficiente per l'esistenza e l'unicità della fattorizzazione LR di Gauss di una generica matrice $A \in \mathbb{R}^{n\times n}$. Si specifichino poi le condizioni solamente sufficienti. Per la matrice definita sopra, quale di queste condizioni sufficienti è soddisfatta?

2. (M) (3 punti) Si realizzi un M-script file che accetta in input la dimensione n del sistema, costruisce la matrice A e il termine noto $b \in \mathbb{R}^n$. Si risolva il sistema lineare $Ax = b$ usando la function di Matlab `chol` per calcolare la fattorizzazione di Choleski e le M-function `sollower` e `solupper` in allegato per le risoluzioni dei sistemi triangolari. Stampare l'errore relativo commesso in norma infinito.
   Calcolare e stampare il residuo normalizzato (in norma infinito). Fornire una stima del numero di condizione (in norma infinito) del sistema, per verificare se il residuo normalizzato risulta essere una buona stima dell'errore relativo. Per $n = 20$, la soluzione risulta accettabile?

Esercizio 3

1. (T) (3 punti) Risolvere con il metodo di eliminazione di Gauss con pivoting parziale il seguente sistema lineare:

$$\begin{aligned} 3x_1 + 5x_3 &= 13 \\ 7x_1 + x_2 + 2x_3 &= 11 \\ x_1 + 3x_2 + x_3 &= 3 \end{aligned}$$

   Indicare gli elementi delle matrici L,R e P della fattorizzazione $PA = LR$ usata per risolvere il sistema. Calcolare il determinante di A, usando la fattorizzazione.

2. (M) (1 punto) Realizzare un M-script file che controlli i risultati ottenuti nell'esercizio precedente, utilizzando la M–function `gauss2` per il calcolo di $PA = LR$ e le M-functions `sollower.m` e `solupper.m` per la risoluzione del sistema (vedere codici in allegato). Stampare la soluzione e il residuo normalizzato (in norma infinito).

3. (M) (3 punti) Usare la fattorizzazione $PA = LR$ calcolare per determinare l'inversa della matrice A (senza usare la function `inv` di Matlab). Stampare il risultato. Calcolare una stima dell'errore commesso, determinando la norma infinito di $I - AX$, ove X è l'inversa calcolata.
   L'output ottenuto deve essere:

```
 inversa di A
-0.0588235 0.176471 -0.0588235
-0.0588235 -0.0235294 0.341176
0.235294 -0.105882 0.0352941
stima dell'errore=2.498e-16
```

Esercizio 4

Si consideri la seguente matrice di ordine n, con $n \geq 2$:

$$A = \begin{pmatrix} 1 & \frac{1}{2}e^{\frac{2}{n}} & & & & & & \\ \frac{1}{2}e^{\frac{2}{n}} & 4 & \frac{1}{2}e^{\frac{3}{n}} & & & & & \\ & \frac{1}{2}e^{\frac{3}{n}} & 3^2 & \frac{1}{2}e^{\frac{4}{n}} & & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \frac{1}{2}e^{\frac{n-2}{n}} & (n-2)^2 & \frac{1}{2}e^{\frac{n-1}{n}} & \\ & & & & & \frac{1}{2}e^{\frac{n-1}{n}} & (n-1)^2 & \frac{1}{2}e^{1} \\ & & & & & & \frac{1}{2}e^{1} & n^2 \end{pmatrix}$$

1. (T) (4 punti) Dato il sistema $Ax = b$, ove A è la matrice sopra definita, enunciare con quali metodi iterativi è possibile ottenere la soluzione del sistema, essendo verificate le condizioni di convergenza (motivare). Quale metodo risulta asintoticamente più veloce? Motivare. Fornire la formula del valore ottimo del parametro di rilassamento per il metodo SOR.

2. (M) (3 punti) Realizzare un M-script file Matlab che accetta in input la dimensione n della matrice, costruisce mediante istruzioni di memorizzazione sparsa la matrice A e determinare il vettore $b \in \mathbb{R}^n$ tale che la soluzione di $Ax = b$ sia un vettore costituito da tutti 1. Utilizzando la function `sor` fornita in appendice, applicare il metodo SOR per la risoluzione del sistema $Ax = b$, utilizzando una tolleranza `tol = 1e-6` , un numero massimo di iterazioni `maxit = 1000` e vettore iniziale nullo. Si utilizzi il valore del parametro di rilassamento ottimale (usare la function di Matlab `eig` per calcolare gli autovalori della matrice di Iacobi, ricordando che la function richiede che la matrice abbia memorizzazione densa). Si stampi il numero di iterazioni impiegate dal metodo per risolvere il sistema per $n = 10$.
   L'output che deve essere ottenuto è il seguente:

```
omega=1.02811
n. iterazioni=8
```

Esercizio 5

Si consideri la seguente funzione

$$f(x) = \frac{1}{1 + 49x^2}$$

definita sull'intervallo $[-1, 1]$.

1. (T) (2 punti) Si scriva l'espressione analitica del polinomio di interpolazione di Newton della funzione relativo ai nodi $-1, 0, 1$.

2. (M) (4 punti) Si disegni il grafico della funzione $f(x)$ basato su un vettore z di 100 punti di tabulazione equispaziati nell'intervallo considerato. In nuove finestre grafiche si disegni il grafico dei polinomi di interpolazione di grado 5 e di grado 20 della funzione assegnata relativo a nodi equispaziati, usando la M-function `pol_lagrange` in allegato. Inserire i titoli nei grafici. Aprire una nuova finestra grafica e fare il grafico dell'errore dei due casi nella stessa finestra grafica. Calcolare l'errore massimo connesso nei due casi, considerando i punti z.

3. (T) (2 punti) Come si può giustificare teoricamente l'andamento dell'errore riscontrato al punto precedente? Si propongano possibili rimedi a questo ben noto fenomeno.

Esercizio 6

1. (M) (4 punti) Considerati le seguenti coppie di dati $(-5,-7),(-3,-5),(0,1),(4,4),(5,3)$, determinare i parametri della retta dei minimi quadrati che meglio approssima questo insieme di dati, usando la function di Matlab `polyfit`. Stampare i coefficienti della retta e la somma dei quadrati dei residui. In una finestra grafica, fare il grafico della retta in $[-4.9, 5.1]$ e segnare i 5 punti.

2. (T) (2 punti) Scrivere cosa si intende per sistema delle equazioni normali nel caso in cui si vuole il polinomio di miglior approssimazione secondo il criterio dei minimi quadrati di grado 2.

3. (T) (2 punti) Se invece di un polinomio si considera la funzione $f(x) = a_1 x + a_2 e^{-x}$, come si possono determinare i parametri a_1 e a_2?

Esercizio 7

Si consideri la funzione

$$f(x) = x^3 - 2(e+4)x^2 + e(e+16)x - 8e^2$$

definita nell'intervallo $[-2, 10]$.

1. (M) (4 punti) Realizzare un M-script file che realizza il grafico della funzione. Dopo aver ottenuto il grafico, con la function `bisez` in allegato, determinare quello dei due zeri della funzione per cui il metodo di bisezione risulta convergere. Stampare il valore ottenuto e il numero di iterazioni. Per l'altro zero (multiplo), modificare la function `newton` in allegato realizzando una function `newton_mod`, in modo da ottenere convergenza (localmente quadratica). Applicare la function a partire dal punto iniziale 0 per ottenere una approssimazione dello zero multiplo. Stampare il risultato ottenuto e il numero di iterazioni.

2. (T) (2 punti) Se si usano come estremi dell'intervallo di ricerca dello zero semplice 5 e 10, quante iterazioni di bisezione sono necessarie per ottenere uno zero con errore 10^{-6}?

3. (T) (2 punti) Quale altro metodo può essere usato per calcolare una stima di uno zero multiplo senza conoscere la molteplicità? Scrivere la formula del metodo.

1 Allegati

```
function [st]=primo(n,b);
%
% algoritmo delle divisioni successive
%
if b<=16
cifre=['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'];
d=n; st='';
while d~=0
   q=fix(d/b);
   r=rem(d,b);
   st=[cifre(r+1),st];
   d=q;
end;
else
    error('base non consentita');
end;

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

function [st]=secondo(n,b,k);
%
% algoritmo delle moltiplicazioni successive
%
cifre=['0','1','2','3','4','5','6','7','8','9',...
    'A','B','C','D','E','F'];
if b<=16
    p=n; st='.';cont=0;
while p~=0 & cont<k
   q=p*b;
   r=fix(q);
   st=[st,cifre(r+1)];
   p=q-r;cont=cont+1;
end;
else
    error('base non consentita');
end;




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


function [L,R,P]=gauss2(A);
% fattorizzazione di Gauss con pivoting parziale - II versione
% P e' un vettore!!
%
```

```
[m,n]=(size(A));
temp=zeros(1,n);
P=1:m;
tol=eps*norm(A,inf);
r=min([m-1,n]);
for k=1:r
   [amax,ind]=max(abs(A(k:m,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;
   end;
   if abs(A(k,k))>tol
      A(k+1:m,k)=A(k+1:m,k)/A(k,k);
      %  operazione di base: aggiornamento mediante diadi
      A(k+1:m,k+1:n)=A(k+1:m,k+1:n)-A(k+1:m,k)*A(k,k+1:n);
   end;
end;
R=zeros(m,n);L=eye(m);
R=triu(A);
L=L+tril(A(1:m,1:m),-1);

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


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;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,k]=sor(A,b,x, maxit,tol,omega);
 % metodo di sor - Gauss Seidel 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,it]=newton(fname,fpname,x0,tolx,tolf,maxit);
% metodo di Newton
x=x0;fx=feval(fname,x);


for it=1:maxit
   d=fx/feval(fpname,x);
   x=x-d;
   fx=feval(fname,x);

   if (abs(fx)<tolf & abs(d)<tolx+eps*abs(x))| fx==0
     break;
   end;
end;
if it>=maxit
    fprintf('raggiunto massimo numero di iterate \n');
end;
```

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