PROVA PRATICA DI CALCOLO NUMERICO

per MATEMATICA APPLICATA E INFORMATICA MULTIMEDIALE

Prof. Stefano De Marchi

`Verona, 20 luglio 2007`

Il candidato dovrà scrivere su **ogni** foglio il cognome, nome, numero di matricola. **Consegnare fogli leggibili!**. Inviare una email a `stefano.demarchi@univr.it` contenente tutti i files e le figure prodotte. **NOTA: allegate immagini in formato .jpg o .eps**

1. Un proiettile viene lanciato in un tunnel di altezza h, con velocità iniziale v_0 e con un'inclinazione $\alpha \geqslant 0$. Se si vuole raggiungere la massima gittata, l'angolo α deve soddisfare la relazione
$$\cos(\alpha) = \sqrt{1 - \frac{2\,g\,h}{v_0^2}}$$
dove $g = 9.81\,m/s^2$. Determinare, con il metodo di Newton, il valore α quando $h = 2$ e $v_0 = 20\,m/s$ (usare come tolleranza $tol = 1.e-9$).

2. Si consideri la matrice $A \in \mathbb{R}^{10\times 10}$
$$A = \begin{bmatrix} 5 & -1 & & & \\ -1 & 5 & -1 & & \\ & -1 & 5 & -1 & \\ & & \ddots & \ddots & \\ & & & -1 & 5 \end{bmatrix},$$
e il vettore $b = \texttt{ones}(10,1)$.
   (a) Si dica (senza fare calcoli) se i metodi iterativi di Jacobi e Gauss-Seidel convergono alla soluzione del sistema $Ax = b$.
   (b) Si consideri ora il metodo iterativo di *Richardson stazionario* per la soluzione di $Ax = b$:
   $$x^{(k+1)} = (I - \alpha\,A)x^{(k)} + \alpha\,b$$
   dove $\alpha \in [0.01, 0.3]$ è un parametro di accelerazione. Si chiede di stimare il parametro ottimale α^* (quello per il cui il metodo di Richardson converge più rapidamente).
   (c) Produrre un grafico comparativo dell'errore assoluto, in funzione del numero delle iterazioni fatte, ottenuto con i metodi di Jacobi, Gauss-Seidel e Richardson stazionario con parametro α^*. Usare: $\texttt{x0} = \texttt{zeros}(10,1)$, $\texttt{tol} = \texttt{1.e-6}$, $\texttt{nmax} = \texttt{100}$.

3. Si consideri la funzione $f(x) = \cos(x^3)\,(x - 2\,\pi)\,e^{-x}$, $x \in [0, \pi]$. Sperimentalmente si determini il grado del polinomio d'interpolazione, costruito sui nodi di Chebsyhev in $[0, \pi]$, che approssima $f(x)$ in norma infinito a meno di $tol = 1.e-4$.

$\Diamond\Diamond$

Tempo: **3 ore**.

SOLUZIONI

Es. 1.

```
clear
%-------------------------------------------
% Esercizio 1 del compito del 20 luglio 2007
%-------------------------------------------

% Dati del problema
g=9.81; h=2; v0=20;

c=sqrt(1-2*g*h/v0^2);

% Domanda: dove si trova lo zero?
% Con i dati del problema sqrt(1-2gh/v0^2)=0.9497
% facendo il plot della funzione solo in [0.01,1]
% si vede che l'intervallo [0.01,1] e' l'intervallo
% separatore della radice che c'interessa.

x=linspace(0.01,1,100); y=cos(x)-c; plot(x,y,'r-') grid pause
% Uno zero sembra essere circa in x*=0.31

% Ora applichiamo Newton per vedere se converge ovvero se determina x*.
% Per esserne certi dobbiamo vedere se nell'intervallo separatore la
% funzione d'iterazione di Newton converge guardando al grafico
% della sua funzione derivata

% Plot della derivata della funzione d'iterazione

y1=1./(tan(x).^2)-c*cos(x)./sin(x).^2; plot(x,y1); grid pause

% C'e' convergenza per x>0.15 circa.

x0=0.2; %punto iniziale
tol=1.e-9; kmax=100;

% Metodo di Newton

[f,fp]=fun(x0); x1=x0-f/fp; iter=1;

while abs(1-x0/x1)> tol & iter<=kmax,
    x0=x1;
    [f,fp]=fun(x0);
     x1=x0-f/fp;
```

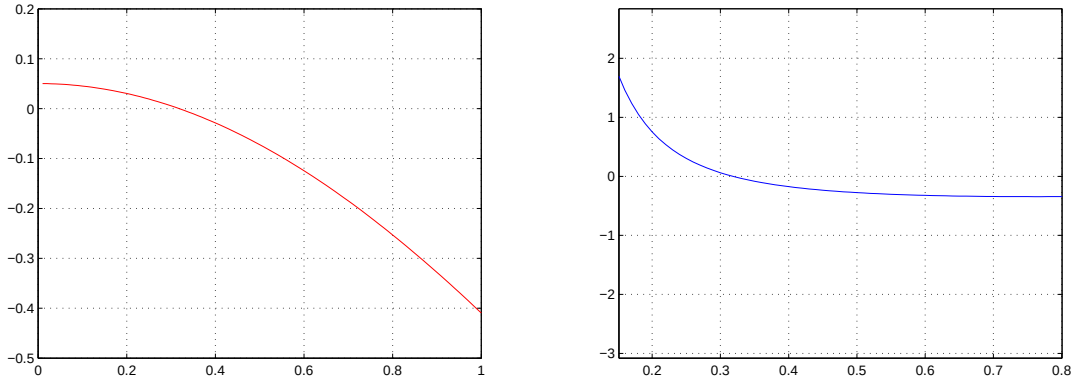

Figure 1: Sx:La fuzione $f(x) = cos(x) - \sqrt{1 - 2gh/v_0^2}$ in [0.01,1]. Dx: La derivata della funzione d'iterazione del metodo di Newton per $x > 0.15$

```
    iter=iter+1;
end iter=iter-1 x1


% Il metodo di Newton converge alla radice
% in 5 iterazioni a x1 =0.3186.


%---------------------------------
function [y,yp]=fun(x)

 % Funzione del 20/7/07
 g=9.81; h=2; v0=20;
 y=cos(x)-sqrt(1-2*g*h/v0^2);
 yp=-sin(x);

return
```

Es. 2. (a) Per quanto rigurda i metodi di Jacobi e Gauss-Seidel, osserviamo che entrambi convergono poiché la matrice è simmetrica e diagonalmente dominante in senso stretto.

(b) Implementazione

```
clear

%-----------------------------------------
% Esercizio 2 del compito del 20 luglio 2007
%-----------------------------------------

n=10; %dimensione matrice
tol=1.e-6; kmax=100;

% Costruisco la matrice e il termine noto
a=-ones(n-1,1); c=5*ones(n,1); A=diag(a,-1)+diag(c)+diag(a,1);

b=ones(n,1);


x0=zeros(n,1);

% Metodo di Richardson stazionario
alf=linspace(0.02,0.3,100); for i=1:length(alf)
 [x1,iter_alf(i),errRS]=RichardsonStazionario(A,b,x0,alf(i),tol,kmax);
end plot(alf,iter_alf,'r.') xlabel('\alpha'); ylabel('nr.
iterazioni');

[m,imin]=min(iter_alf); alfmin=alf(imin) disp('Premi un tasto per
continuare');

pause


[solRS,iterRS,errRS]=RichardsonStazionario(A,b,x0,alfmin,tol,kmax);
[solGS,iterGS,errGS]=GaussSeidel(A,b,x0,tol,kmax);
[solJ,iterJ,errJ]=Jacobi(A,b,x0,tol,kmax);
%

plot(1:length(errRS),errRS,'r-',1:length(errGS),errGS,'b-',1:length(errJ),errJ,'g-');
legend('Richardson','Gauss-Seidel','Jacobi');

%-------------------------------------------------------------
% Di seguito forniamo solo la funzione che implementa il metodo
```

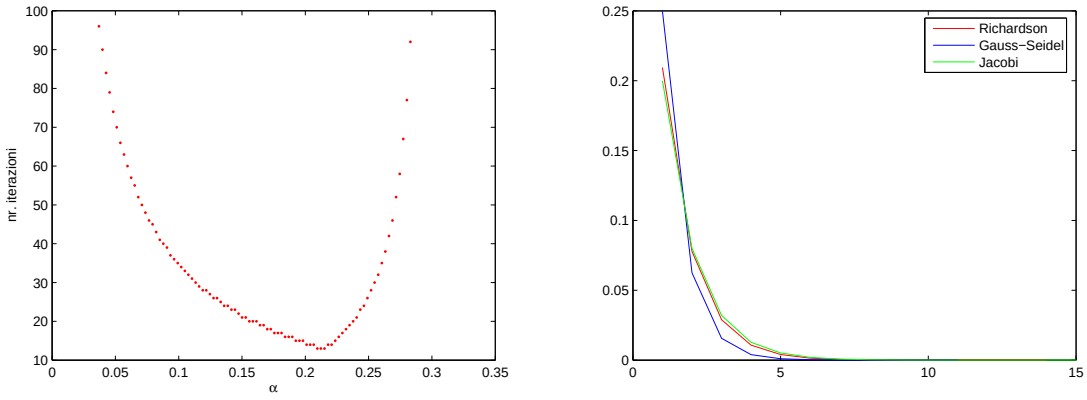

Figure 2: sx: ricerca del parametro α ottimale. dx: andamento dell'errore dei metodi di Jacobi, Gauss-Seidel e Richardoson con α ottimale.

```
% di Richardson stazionario, poich le funzioni dei metodi di
% Jacobi e Gauss-Seidel sono gi state viste durante il corso
% nelle esercitazioni di laboratorio
%---------------------------------------------------------------------

function [x1,iter,err]=RichardsonStazionario(A,b,x0,alf,tol,kmax)

n=length(b); I=eye(n);

 x1=(I-alf*A)*x0+alf*b;
 k=1;
 err(k)=norm(x1-x0,inf);
while(norm(x1-x0)> tol & k<=kmax)
    x0=x1;
    x1=(I-alf*A)*x0+alf*b;
    k=k+1;
    err(k)=norm(x1-x0,inf);
end
iter=k-1;

return


%
```

Es. 3 Si tratta semplicemente di implementare l'interpolazione polinomiale su nodi di Chebsy-

hev, con grado che varia finchè la condizione sull'errore non sarà soddisfatta.

```
clear
%-----------------------------------------
% Esercizio 3 del compito del 20 luglio 2007
%-----------------------------------------

a=0;
b=pi;


n=3;
tol=1.e-4;
err=2;
xx=a:0.01:b;     % target points
yy=funInt(xx);   % i valori della funzione sui targets

while err> tol

%nodi di Chebyshev
xc=-(b-a)/2*cos((2*[1:n+1]-1)*pi/(2*(n+1)))+(b+a)/2;
% Polinomio su nodi di Chebyshev
   yc=funInt(xc);
   dc=diffDivise(xc,yc);
   pc=horner(xc,dc,xx);
   err=norm(yy-pc,inf);
  n=n+1;
end

   plot(xc,zeros(length(xc),1),'o',xx,yy,'-',xx,pc,'--')
   legend('nodi di Chebyshev','funzione','polinomio su nodi di Chebyshev')

%---------------------------------------
function y=funInt(x)
  y=cos(x.^3).*(x-2*pi).*exp(-x);
return


>> esame20luglio2007esIII
>> n

n =
```

39

Le funzioni `diffDivise.m` e `horner.m` sono state implementate durante il corso. Come si vede il minimo grado per soddisfare le richieste dell'esercizio è $n = 39$. Di seguito il grafico corrispondente della funzione e del polinomio interpolante di grado 39.