Calcolo Scientifico

Primo Progetto

Žana Ilić - 898373

20 aprile 2019.

Primo Esercizio

Il nostro problema è di derivare una approsimazione alle differenze finite di tipo one-sided (questo significa che dobbiamo prendere i valori solo sulla sinistra o sulla destra, in nostro caso sulla sinistra) della derivata prima di una funzione regolare $u = u(x)$ nel punto $x = x_0$. Utilizziamo la seguente schema:

$$D_s^- u(x_0) = au(x_0) + bu(x_0 - h) + cu(x_0 - 2h)$$

con h parametro di discretizzazione e a, b, c parametri reali. Vogliamo trovare i valori di a, b, c che massimizzano l'accutarezza dell'approssimazione. Usando sviluppo di Taylor abbiamo (con il resto di Lagrange) :

$$u(x_o - h) = u(x_0) - hu'(x_0) + \frac{h^2}{2}u''(x_0) - \frac{h^3}{6}u'''(x_0) + \frac{h^4}{24}u''''(\rho)$$

$$u(x_o - 2h) = u(x_0) - 2hu'(x_0) + 2h^2u''(x_0) - \frac{8h^3}{6}u'''(x_0) + \frac{16h^4}{24}u''''(\theta)$$

dove $\rho \in (x_0 - h, x_0)$ e $\theta \in (x_0 - 2h, x_0)$. $D_s^- u(x_0)$ è allora:

$$D_s^- u(x_0) \approx u(x_0)(a + b + c) - hu'(x_0)(b + 2c) + \frac{h^2}{2}u''(x_0)(b + 4c) - \frac{h^3}{6}u'''(x_0)(b + 8c)$$

Per ottenere la massima accuratezza cioè $D_s^- u(x_0) \approx u'(x_0)$, gli parametri reali a, b, c devono essere:

$$\begin{cases} a + b + c = 0 \\ b + 2c = -\frac{1}{h} \\ b + 4c = 0 \end{cases} \implies \begin{cases} a = \frac{3}{2h} \\ b = -\frac{2}{h} \\ c = \frac{1}{2h} \end{cases}$$

Infine, $D_s^- u(x_0)$ è:

$$D_s^- u(x_0) = \frac{1}{2h}(3u(x_0) - 4u(x_0 - h) + u(x_0 - 2h)) + \epsilon$$

Calcoliamo ora l'errore ϵ:

$$\epsilon = |D_s^- u(x_0) - u'(x_0)| = \left|\frac{h^3}{6}u'''(x_0)(b + 8c)\right| = \left|\frac{h^2}{3}u'''(x_0)\right| \leq \frac{Mh^2}{3} = Ch^2$$

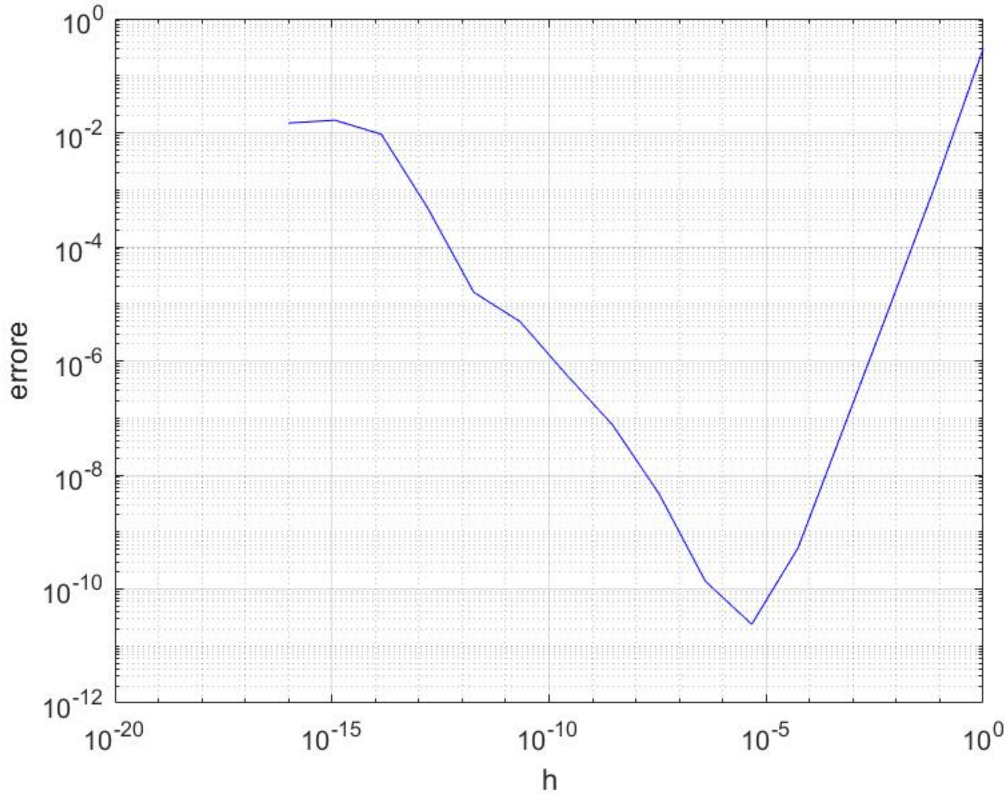

Figura 1: Ordine di convergenza al variare di h

In Matlab calcoliamo l'ordine di convergenza trovato al variare di h nel caso $u(x) = sin(x)$ con $x_0 = 1$.

Secondo Esercizio

Consideriamo l'equazione:

$$\begin{cases} u''(x) = \sqrt{1 + u'(x)^2} & x \in (-1, 1) \\ u(-1) = u(1) = 1 \end{cases}$$

Usando il sviluppo di Taylor, troviamo le differenze finite centrate del secondo ordine della derivata prima e seconda, con un passo di discretizzazione h:

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \frac{h^3}{6}u'''(\rho)$$

$$u(x_{i-1}) = u(x_i) - hu'(x_i) + \frac{h^2}{2}u''(x_i) - \frac{h^3}{6}u'''(\theta)$$

da cui troviamo

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_{i-1})}{2h} - \varepsilon$$

dove ε è l'errore locale. Analogmente,

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \frac{h^3}{6}u'''(x_i) + \frac{h^4}{24}u''''(\rho)$$

$$u(x_{i-1}) = u(x_i) - hu'(x_i) + \frac{h^2}{2}u''(x_i) - \frac{h^3}{6}u'''(x_i) + \frac{h^4}{24}u''''(\theta)$$

da cui troviamo

$$u''(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - \tau$$

dove τ è l'errore locale.

Dal condizioni al bordo abbiamo che $u(x_1) = u_1 = 1$ e $u(x_n) = u_n = 1$. Usando le differenze finite centrate del secondo ordine della derivata prima e seconda appena calcolate, la discretizzazione del nostro problema con applicati condizioni al bordo diventa:

$$A \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} 1 \\ \sqrt{1 + (\frac{u_3 - u_1}{2h})^2} \\ \vdots \\ \sqrt{1 + (\frac{u_n - u_{n-2}}{2h})^2} \\ 1 \end{bmatrix} - \mathbf{b}$$

Cioè dobbiamo risolvere il seguente sistema non lineare:

$$F(\mathbf{u}) = A\mathbf{u} - \sqrt{1 + (B\mathbf{u})^2} - \mathbf{b} = 0$$

dove $\mathbf{u} = [u_1, u_2, \ldots, u_{n-1}, u_n]^T$, $\mathbf{b} = [\frac{1}{h^2}, 0, \ldots, 0, \frac{1}{h^2}]^T$, $\mathbf{b}$ è vettore che completa la differenza dei nodi sul bordi, e i matrici A e B sono:

$$A = \frac{1}{h^2}\begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ 1 & -2 & 1 & 0 & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 & -2 & 1 \\ 0 & \ldots & 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \frac{1}{2h}\begin{bmatrix} 0 & 0 & 0 & 0 & \ldots & 0 \\ -1 & 0 & 1 & 0 & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & -1 & 0 & 1 \\ 0 & \ldots & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ora vogliamo usare il metodo di Newton per trovare lo zero del nostra problema non lineare. Prima calcoliamo jacobiano di $F(\mathbf{u})$:

$$J_F(\mathbf{u}^{(k)}) = A - d_{ij}(\mathbf{u}^{(k)})B$$

dove

$$d_{ij}(\mathbf{u}^{(k)}) = \begin{cases} \frac{B(\mathbf{u}^{(k)})}{\sqrt{1+(\mathbf{u}^{(k)})^2}} & i = j \\ 0 & i \neq j \end{cases}$$

Allora sviluppiamo $F(\mathbf{u})$ in serie di Taylor scegliendo $\mathbf{u}^{(k)}$ come punto iniziale, per ottenere la sistema lineare, e abbiamo:

$$\begin{aligned} & F(\mathbf{u}) = F(\mathbf{u}^{(k)}) + J_F(\mathbf{u}^{(k)})(\mathbf{u} - \mathbf{u}^{(k)}) + ... \\ \Longrightarrow \quad & F(\mathbf{u}^{(k+1)}) \approx F(\mathbf{u}^{(k)}) + J_F(\mathbf{u}^{(k)})(\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}) = 0 \\ & \Longrightarrow \quad J_F(\mathbf{u}^{(k)})(\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}) = -F(\mathbf{u}^{(k)}) \end{aligned}$$

In Matlab troviamo la soluzione numerica del nostro problema, prendendo soluzione esatta del problema, che soddisfa le condizioni al bordo. Tale funzione è $u(x) = cosh(x) - cosh(1) + 1$.

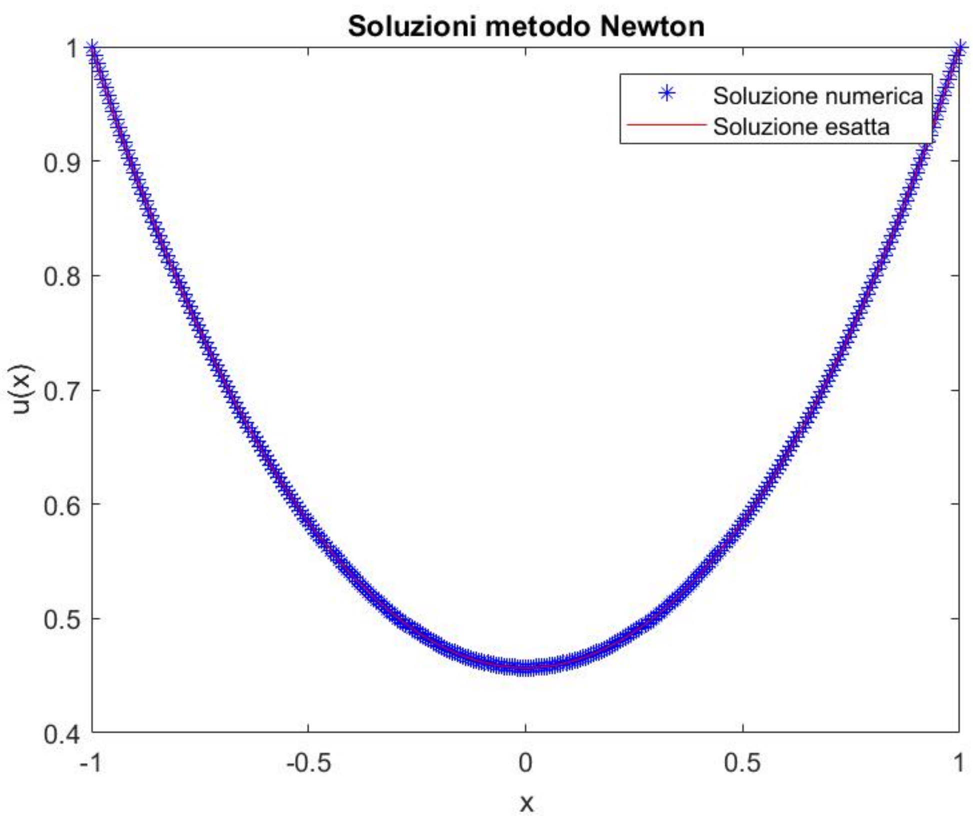

Figura 2: Soluzioni del problema non lineare tramite metodo di Newton

Ora utilizziamo il metodo di shooting per la risoluzione dello stesso problema non lineare. Il nostro problema ai limiti:

$$\begin{cases} u''(x) = f(x, u(x), u'(x)) = \sqrt{1 + u'(x)^2} & x \in (-1, 1) \\ u(-1) = 1 \\ u(1) = 1 \end{cases}$$

si può trasformare in un sistema differenziale del primo ordine con problema ai valori iniziali, con $u(x, s)$ soluzione del problema ai valori iniziali, dove $s \in \mathbb{R}$, che varia fino a che $u(1) = 1$:

$$\begin{cases} u'(x) = f(x, u(x)) & x \in (-1, 1) \\ u(-1) = 1 \\ u'(-1) = s \end{cases}$$

Dobbiamo trovare s^* tale che $u(1, s^*) = 1$, cioè una funzione F in s tale che $F(s^*) = u(1, s^*) - 1 = 0$. Questo ultimo si può fare in Matlab con funzione `fzero`, e $u(1, s)$ si calcolo in Matlab con funzione `ode45` con $u(-1) = 1$ e $u'(-1) = s$.

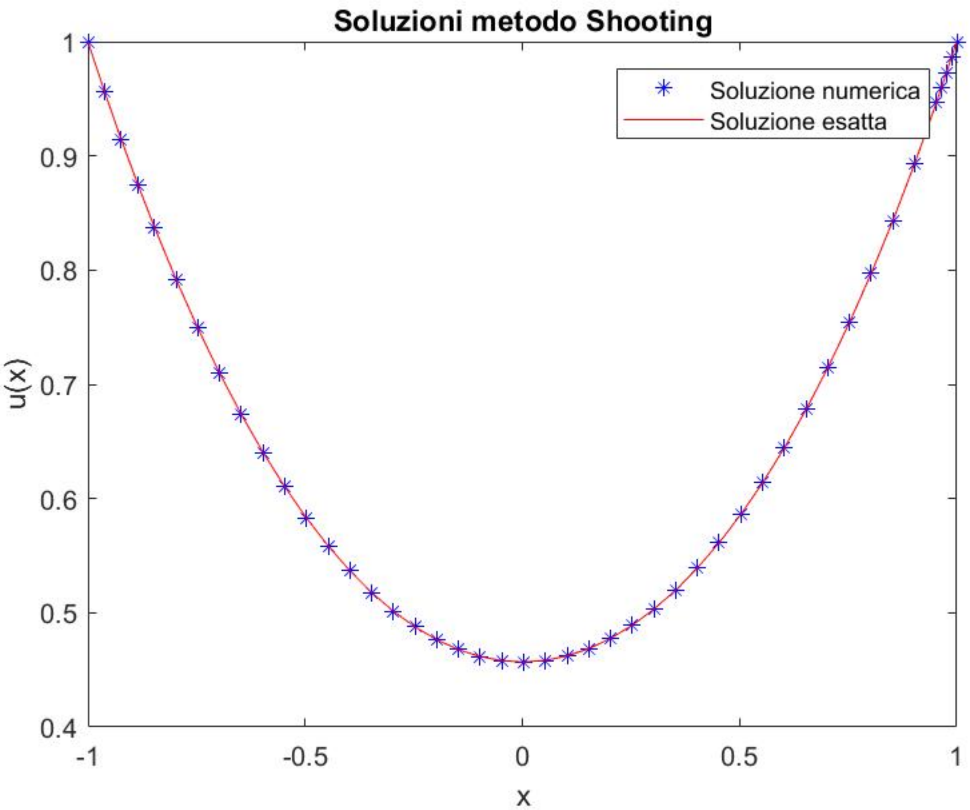

Figura 3: Soluzioni del problema non lineare tramite metodo di shooting

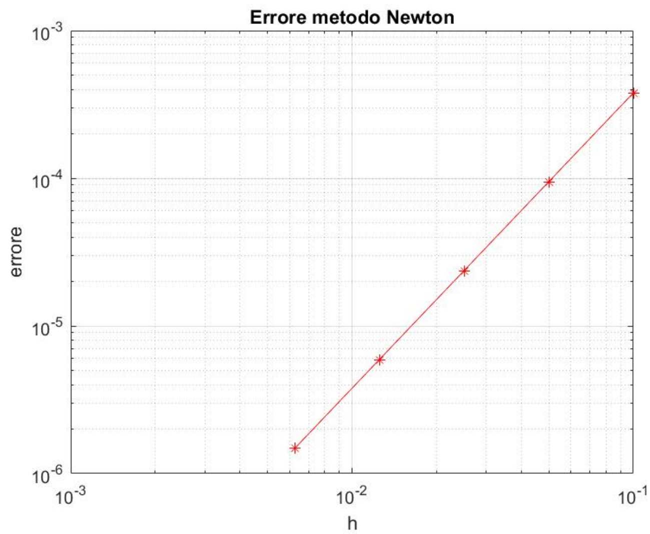

Figura 4: Errore in norma infinito della soluzione al variare di h con metodo di Newton

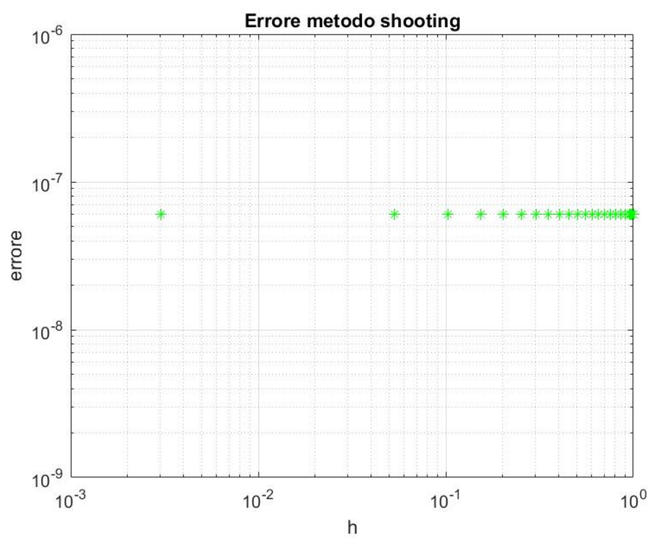

Figura 5: Errore in norma infinito della soluzione al variare di h con metodo di shooting

Infine, in Matlab calcoliamo il grafico dell'errore in norma infinito della soluzione al variare di h per due metodi.

Terzo Esercizio

Consideriamo il problema differenziale:

$$\begin{cases} u''(x) + u'(x) = f(x) & x \in (0, \pi/2) \\ u(0) = u(\pi/2) = 0 \end{cases}$$

con $f(x) = x$. Calcoliamo la soluzione esatta tramite integrazione classica. Prima calcoliamo soluzione particolare dell'equazione omogenea associata:

$$u''(x) + u'(x) = 0$$

$$\lambda^2 + 1 = 0 \implies \lambda_1 = i \quad \lambda_2 = -i$$

Poichè il polinomio caratteristico ha due radici complesse coniugate $\alpha_1 + i\beta_1$ e $\alpha_2 - i\beta_2$ allora $u_1(x) = e^{\alpha_1 x} cos(\beta_1 x)$ e $u_2(x) = e^{\alpha_2 x} sin(\beta_2 x)$ e la soluzione dell'equazione omogenea è allora:

$$u_{om}(x) = c_1 sin(x) + c_2 cos(x)$$

Poichè $f(x) = x$ con radice $\lambda = 0$, che non coincide nè con λ_1 nè con λ_2, soluzione particolare $\bar{u}(x)$ della non omogenea sarà del tipo $\bar{u}(x) = e^{\lambda x} g(x) = g(x)$ dove $g(x)$ è polinomio dello stesso grado di $f(x)$, cioè $g(x) = Ax + B$ con A e B parametri reali. E quindi:

$$\bar{u}(x) = Ax + B$$

Deriviamo ora fino all'ordine due e sostituiamo nell'equazione in partenza:

$$\begin{cases} \bar{u}'(x) = A \\ \bar{u}''(x) = 0 \end{cases} \implies \begin{cases} A = 1 \\ B = 0 \end{cases}$$

La soluzione particolare sarà $\bar{u}(x) = x$ e la soluzione generale dell'equazione differenziale è data da:

$$u(x) = u_{om}(x) + \bar{u}(x) = x + c_1 sin(x) + c_2 cos(x)$$

Infine, troviamo i parametri c_1 e c_2, calcolando $u(0) = 0$ e $u(\pi/2) = 0$:

$$\begin{cases} u(0) = c_2 = 0 \\ u(\pi/2) = \pi/2 + c_1 = 0 \end{cases} \implies \begin{cases} c_1 = -\pi/2 \\ c_2 = 0 \end{cases}$$

Soluzione esatta della nostra problemma è:

$$u(x) = x - \frac{\pi}{2} sin(x)$$

Per trovare la soluzione generale del problema in forma alternativa

$$u(x) = \int_0^{\pi/2} G(x - x') f(x') \, \mathrm{d}x$$

dobbiamo determinare la funzione di Green associata al problema, che è soluzione della problema ai valori iniziali

$$G'' + G = \delta(x - x')$$

con $G(0, x') = 0$ e $G(\pi/2, x') = 0$. Per trovare la soluzione dividiamo il problema in due domini: $x \in [0, x')$ e $x \in (x', \pi/2]$. Poichè $x \neq x'$, $\delta(x-x') = 0$, e allora $G'' + G = 0$. Questa equazione ha soluzioni del tipo $G(x, x') = A(x')sin(x) + B(x')cos(x)$, cioè:

$$\begin{cases} G_1(x, x') = A_1(x')sin(x) + B_1(x')cos(x), & 0 \leq x < x' \\ G_2(x, x') = A_2(x')sin(x) + B_2(x')cos(x), & x' < x \leq \pi/2 \end{cases}$$

Applicando $G_1(0, x') = 0$ e $G_2(\pi/2, x') = 0$ concludiamo che $B_1 = 0$ e $A_2 = 0$:

$$\begin{cases} G_1(x, x') = A_1(x')sin(x), & 0 \leq x < x' \\ G_2(x, x') = B_2(x')cos(x), & x' < x \leq \pi/2 \end{cases}$$

$G(x, x')$ deve essere continua in $x = x'$ che significa che

$$A_1(x')sin(x) = B_2(x')cos(x)$$

Ora integriamo l'equazione tra x'_- e x'_+:

$$\int_{x'_-}^{x'_+} G'' \, \mathrm{d}x + \int_{x'_-}^{x'_+} G \, \mathrm{d}x = \int_{x'_-}^{x'_+} \delta(x - x') \, \mathrm{d}x = 1$$

e la condizione "jump" è $G'|_{x'_-}^{x'_+} = 1$. Troviamo le derivate prime:

$$\begin{cases} G'_1(x, x') = A_1(x')cos(x), & 0 \leq x < x' \\ G'_2(x, x') = -B_2(x')sin(x), & x' < x \leq \pi/2 \end{cases}$$

e concludiamo che $-B_2(x')sin(x) - A_1(x')cos(x) = 1$. Alla fine, combinando due equazioni con due incognite, troviamo A_1 e B_2. La funzione di Green si può scrivere così:

$$G(x,x') = H(x'-x)(-cos(x')sin(x)) + H(x-x')(-sin(x')cos(x))$$

dove $H(x,x')$ è la funzione di Heaviside: $H(x-x') = \begin{cases} 1, & x > x' \\ 0, & x < x' \end{cases}$

Mettiamo tutto insieme:

$$\int_0^x -x'sin(x')cos(x)\,\mathrm{d}x' + \int_x^{\pi/2} -x'cos(x')sin(x)\,\mathrm{d}x' = \cdots = x - \frac{\pi}{2}sin(x)$$

Codici Matlab

Codice di primo esercizio

```matlab
% Primo Progetto Calcolo Scientifico
% Primo Esercizio

clear all, close all

f=@(x)sin(x);
df=@(x)cos(x); %derivata esatta

h=logspace(0,-16,16);
x0=1;

for i=1:numel(h)
    derf=(3*(f(x0))-4*(f(x0-h(i)))+f(x0-2*h(i)))/(2*h(i));
    err(i)=abs(derf-df(x0));
end

loglog(h,err,'b');
xlabel('h');
ylabel('errore');
hold on
grid
```

Codice di secondo esercizio-metodo di Newton

```matlab
% Primo Progetto Calcolo Scientifico
% Secondo Esercizio
% Metodo di Newton con errore in norma infinito

clear all, close all

rang=[20 40 80 160 320];
cont=0;

for n=rang+1
    cont=cont+1;
```

```
x=linspace(-1,1,n)'; % a=-1 & b=1
h=2/(n-1); % lunghezza dell'intervallo , h=b-a/(n-1)
uex=@(x)(cosh(x)-cosh(1)+1); % soluzione esatta

% Matrice A
A=zeros(n);
d=-2*ones(n,1);
d1=ones(n-1,1);
A=diag(d)+diag(d1,-1)+diag(d1,1);
A(1,1)=1; A(1,2)=0; % cambio nodi sul bordo
A(n,n)=1; A(n,n-1)=0;
A=(1/(h^2))*A;

% Matrice B
B=zeros(n);
db=ones(n-1,1);
B=diag(-db,-1)+diag(db,1);
B(n,n-1)=0; B(1,2)=0; % cambio nodi sul bordo
B=(1/(2*h))*B;

% Vettore b
b=zeros(n,1);
b(1)=b(1)+1/h^2-1;
b(end)=b(end)+1/h^2-1;

% Metodo di Newton
F=@(u)A*u-sqrt(1+(B*u).^2)-b;

% Jacobiano di F
JF=@(u)A-diag((B*u)./sqrt(1+(B*u).^2))*B;

u0=ones(n,1);
%u0=linspace(-1,1,n)';
u=u0;

% Calcolo di errore
tol=h^2/n;
delta=-JF(u)\F(u);
```

```
    while (norm(delta,inf)>tol)
        u=u+delta;
        delta=-JF(u)\F(u);
    end
    u=u+delta;
    err(cont)=norm(u-uex(x),inf);
    p(cont)=2/(n-1); % =h
end

figure(1)
plot(x,u,'b*',x,uex(x),'r-');
title('Soluzioni metodo Newton')
legend('Soluzione numerica','Soluzione esatta')
xlabel('x')
ylabel('u(x)')

figure(2)
loglog(p,err,'r-*');
%loglog(rang,err,'r-*');
title('Errore metodo Newton');
xlabel('h')
ylabel('errore')
grid on
```

Codice di secondo esercizio-metodo di shooting

```
% Primo Progetto Calcolo Scientifico
% Secondo Esercizio
% Metodo di Shooting con ode45 e fzero
% Sto usando i funzioni: odefunz.m e shoot.m

clear all, close all

global a b xa xb
xa=1; xb=1;
a=-1; b=1;
xex=@(t)(cosh(t)-cosh(1)+1); % soluzione esatta

s0=-0.05;
```

```
s=fzero(@shoot,s0);

[t,x]=ode45(@odefunz,[a,b],[xa,s]);

figure(1)
plot(t,x(:,1),'b*',t,xex(t),'r-')
title('Soluzioni metodo Shooting')
legend('Soluzione numerica','Soluzione esatta')
xlabel('x')
ylabel('u(x)')

err=norm((x(:,1)-xex(t)), inf);
figure(2)
loglog(t,err,'g-*');
title('Errore metodo shooting');
xlabel('h')
ylabel('errore')
grid on
```

```
% funzione per secondo esercizio-metodo di shooting
function F=shoot(s)

global a b xa xb

[t,x]=ode45(@odefunz,[a,b],[xa,s]);

F=x(end,1)-xb;

return
```

```
% funzione per secondo esercizio-metodo di shooting
function f=odefunz(t,y)

t=y(1);
z=y(2);

f=zeros(2,1);
f(1)=z;                    %u'=z;
```

```
f(2)=sqrt(1+f(1).^2);   %u''=z'=sqrt(1+u'^2)=sqrt(1+z^2)

return
```

Codice di terzo esercizio

```
% Calcolo Scientifico
% Terzo esercizio
% Metodo del punto medio composito

clear all, close all

a=0; b=pi/2;

uex=@(x)(x-(pi/2)*sin(x)); % u esatto
alphaex=@(x)(sin(x)-x.*cos(x));    % alpha=integrale da 0 a x esatto
betaex=@(x)(-sin(x)-cos(x)+(pi/2));  % beta=integrale da x a pi/2 esatto
%alpha=@(y)sin(y);
%beta=@(y)cos(y);
f=@(x)sin(x);
g=@(x)cos(x);
%u=@(x)(-cos(x).*alphaex(x)-sin(x).*betaex(x));
x(1)=0;
u(1)=0;
alpha(1)=0;
beta(1)=0;
u(end)=pi/2;
z=linspace(0,pi/2);
nn=[20 40 80 160 320];
for k=1:numel(nn)
    n=nn(k);
    h=(b-a)/n;
    hh(k)=h;
    for i=1:n
        x(i+1)=x(i)+h;
        x12=x(i)+h/2;
        %diff(alpha(x12))
```

```
            %alpha(i+1)=alpha(i)-h*diff(alpha(x12));
            %beta(i+1)=beta(i)-h*diff(beta(x12));
            alpha(i+1)=alpha(i)-h*(x(i)*f(x(i))+2*x12*f(x12)+x(i+1)*f(x(i+1)));
            beta(i+1)=beta(i)-h*(x(i)*g(x(i))+2*x12*g(x12)+x(i+1)*g(x(i+1)));
            u(i+1)=-g(x(i+1))*alpha(i+1)-f(x(i+1))*beta(i+1);
        end
        figure(1)
        plot(z,uex(z),'r',x,u,'g-.')
        legend('Soluzione esatta','Soluzione numerica')
        err(k)=norm(uex(x)-u,inf);
    end
%close all
figure(2)
loglog(hh,err,'-*')
xlabel('h')
ylabel('errore')
```