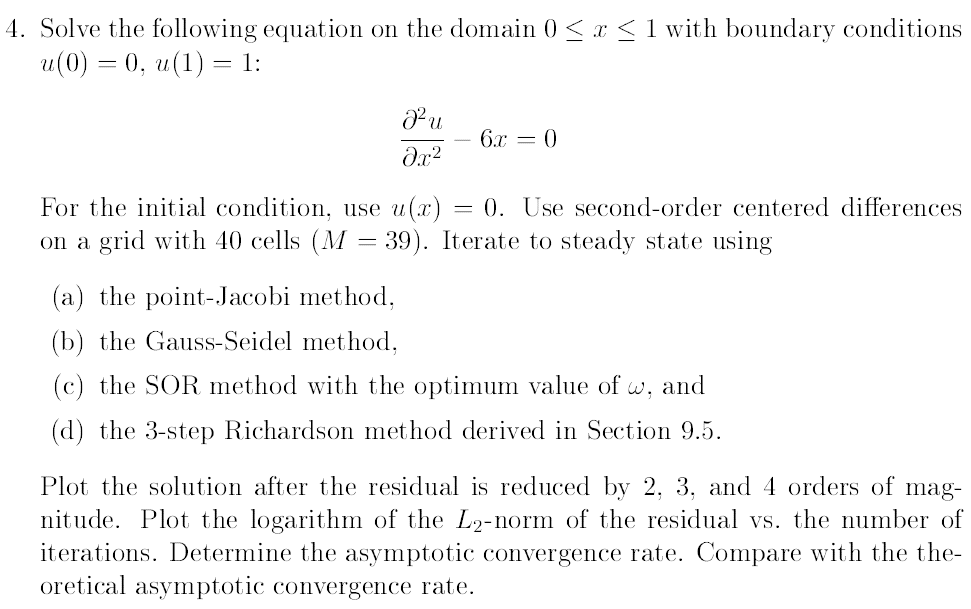
**CFD – HW #4**

**Banafsheh Zebhi**

**Chapter 9**



**Solution:** In order to iterate to steady state solution, we use relaxation method to remove transient solution from the general solution in the most efficient way possible. Relaxation (iterative) methods have no time accuracy and work the best with certain eigenstructures (eg: same sign eigenvalues and all real). Therefore, we precondition the process to prepare the algebraic system to guarantee this certain eigenstructure.

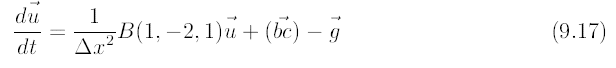
General form of a diffusion equation in one dimensional:

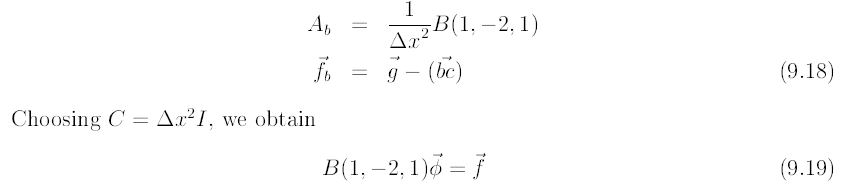


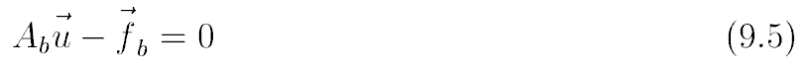
The steady state solution must satisfy:



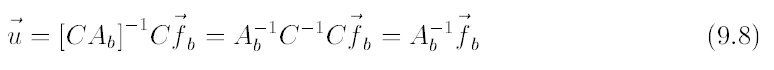
2nd-order centered differencing scheme for second derivative with bc:

, where g= 6x



General form of the equation at steady state:

Now we need to precondition the process:



To obtain convergence solution, the residual and the error:



The error at nth iteration is defined as:



And residual at nth iteration is defined as:



For part (a), **Point-Jacobi method** in one dimension is shown as:



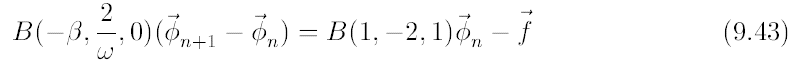
This operator comes about by choosing the value of ϕjn+1such that together with the old values of ϕj-1 and ϕj+1. In this method H= - D meets the criteria of equation 9.34.

For part (b), **Gauss-Seidel method** in one dimension is shown as:

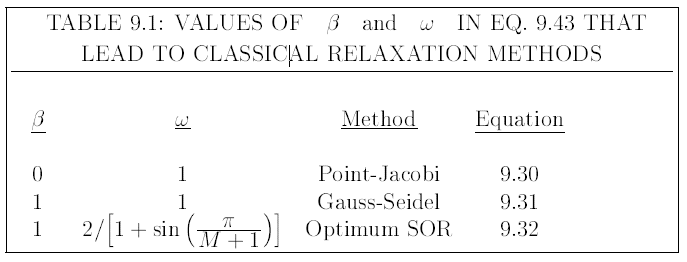


In this method the most recent updates of ϕj-1 is used. So the point operator is satisfied by using the new values of ϕj and ϕj-1 and oldvalue of ϕj+1. In this method H= - (L+ D) meets the criteria of equation 9.34.

The h constrain will lead us to the following general equation:



The values of Β and ω are identified according to table 9.1:



% CFD\_ HW#4\_Chapter 9\_Question 4

% Banafsheh Zebhi

%----------------------------------------- (a), (b), (c) ----------------------------------------------

clc

clear all

close all

M=39; % Grid of 40 cells = 39 nodes

dx=1/(M+1); % Spatial differencing

x=[0+dx:dx:1-dx];

% Boundary Condition: u(0)=0, u(1)=1

bc=zeros(M,1);

bc(M)=1;

bc=(1/dx^2)\*bc;

g=6\*x; % source function

fb=g.'-bc; % eqn 9.18

z=zeros(M,1);

% ------------------2nd-order central differencing matrix----------------------------------------

Ab = (1/dx^2)\*full(gallery('tridiag',M,1,-2,1)); % eqn 9.18

% Create pre-conditioning matrix C

C=eye(M)\*dx^2; % eqn 9.18

A=C\*Ab; % eqn 9.14

f=C\*fb; % eqn 9.14

%------B=Beta, w=omega in table 9.1------------------------------------------------------------

B=0; % Point-Jacobi method, table 9.1

w=1; % Point-Jacobi method, table 9.1

%-------------------------------------------------------------------------------------------------------

B=1; % Gauss-Seidel method, table 9.1

w=1; % Gauss-Seidel method, table 9.1

%-------------------------------------------------------------------------------------------------------

B=1; % SOR method, table 9.1

w=2/(1+sin(pi/(M+1))); % SOR method, table 9.1

%------------------------------------------------------------------------------------------------------

H=full(gallery('tridiag',M,-B,2/w,0)); % eqn 9.43

I=eye(M);

rn=norm(A\*z-f); % eqn 9.26

rn\_0=rn;

i=0;

%---------------------------------iteration to residual reduced by 2 orders-----------------------

while rn>rn\_0\*10^-2

phi2=(I+inv(H)\*A)\*z - inv(H)\*f; % eqn 9.23

rn=norm(A\*phi2-f); % eqn 9.26

i=i+1;

z=phi2;

end

rn2=rn;

i2=i;

CR2=(rn2/rn\_0)^(1/i2)

rn=rn\_0;

i=0;

z=zeros(M,1);

%----------------------------------iteration to residual reduced by 3 orders----------------------

while rn>rn\_0\*10^-3

phi3=(I+inv(H)\*A)\*z - inv(H)\*f; % eqn 9.23

rn=norm(A\*phi3-f); % eqn 9.26

i=i+1;

z=phi3;

end

rn3=rn;

i3=i;

CR3=(rn3/rn\_0)^(1/i3)

rn=rn\_0;

i=0;

z=zeros(M,1);

%-----------------------------------iteration to residual reduced by 4 orders---------------------

while rn>rn\_0\*10^-4

phi4=(I+inv(H)\*A)\*z - inv(H)\*f; % eqn 9.23

rn=norm(A\*phi4-f); % eqn 9.26

i=i+1;

z=phi4;

end

rn4=rn;

i4=i;

CR4=(rn4/rn\_0)^(1/i4)

%----------------plot of solution for residual reduction by 2, 3 and 4 orders------------------

figure(1)

plot(x,phi2,'g',x,phi3,'b--o',x,phi4,'r--\*')

xlabel('Domain 0<x<1')

ylabel('Solution u(x)')

legend('Residuals 2nd-Order','Residuals 3rd-Order','Residuals 4th-Order')

%-----------------Plots: logarithm of L2-norm of residuals vs iteration--------------------

figure(2)

rn\_vec=[log(rn2),log(rn3),log(rn4)];

i\_vec=[i2,i3,i4];

plot(i\_vec,rn\_vec,'-\*')

xlabel('Number of iterations')

ylabel('logarithm of L2-norm of residuals')

%---------------------- Theoretical Asymptotic Convergence Rates------------------

CR\_PointJacobi=cos(pi/(M+1));

CR\_GaussSeidel=CR\_PointJacobi^2;

w\_opt=2/(1+sin(pi/(M+1)));

CR\_SOR=(w\_opt-1);

(a) Point-Jacobi Method



The theoretical asymptotic convergence rate is calculated using equation 9.54:



|  |  |  |  |
| --- | --- | --- | --- |
| **Theoretical** | **2 Order** | **3 Order** | **4 Order** |
| 0.993 | 0.979 | 0.992 | 0.994 |

(b) Gauss-Seidel Method





The theoretical asymptotic convergence rate is calculated using equation 9.63:



|  |  |  |  |
| --- | --- | --- | --- |
| **Theoretical** | **2 Order** | **3 Order** | **4 Order** |
| 0.996 | 0.923 | 0.979 | 0.987 |

(c) SOR Method





The theoretical asymptotic convergence rate is calculated using equation 9.63:



|  |  |  |  |
| --- | --- | --- | --- |
| **Theoretical** | **2 Order** | **3 Order** | **4 Order** |
| 0.854 | 0.862 | 0.878 | 0.882 |

In non-stationary process, conditioning matrices H and C are varied at each time step (h). This doesn’t change the steady state solution (egn 9.5) but it will affect the convergence rate. In a simpler way, we keep the H and C matrices fixed and vary the step size (h). This process changes the Point-Jacobi method to Richardson’s method.

%---------------------------------- (d)3-step Richardson Method------------------------------

clc

clear all

close all

M=39; % Grid of 40 cells = 39 nodes

dx=1/(M+1); % Spatial differencing

x=[0+dx:dx:1-dx];

% Boundary Condition: u(0)=0, u(1)=1

bc=zeros(M,1);

bc(M)=1;

bc=(1/dx^2)\*bc;

g=6\*x; % source function

fb=g.'-bc; % eqn 9.18

z=zeros(M,1);

%-----------------------------2nd-order central differencing matrix--------------------------

Ab = (1/dx^2)\*full(gallery('tridiag',M,1,-2,1)); % eqn 9.18

% Create pre-conditioning matrix C

C=eye(M)\*dx^2; % eqn 9.18

A=C\*Ab; % eqn 9.14

f=C\*fb; % eqn 9.14

%------B=Beta, w=omega in table 9.1-------3-step Richardson------------------------------

B=0; w=1;

h1=4/(6-sqrt(3));

h2=4/6;

h3=4/(6+sqrt(3));

H=full(gallery('tridiag',M,-B,2/w,0));

I=eye(M);

rn=norm(A\*z-f);

rn\_0=rn;

i=0;

phi2=z;

%---------------------------------iteration to residual reduced by 2 orders---------------------

while rn>rn\_0\*10^-2

phi2=(I+inv(H/h1)\*A)\*phi2 - inv(H/h1)\*f;

phi2=(I+inv(H/h2)\*A)\*phi2 - inv(H/h2)\*f;

phi2=(I+inv(H/h3)\*A)\*phi2 - inv(H/h3)\*f;

rn=norm(A\*phi2-f);

i=i+1;

end

rn2=rn;

i2=i;

rn=rn\_0;

i=0;

phi3=z;

%----------------------------------iteration to residual reduced by 3 orders--------------------

while rn>rn\_0\*10^-3

phi3=(I+inv(H/h1)\*A)\*phi3 - inv(H/h1)\*f;

phi3=(I+inv(H/h2)\*A)\*phi3 - inv(H/h2)\*f;

phi3=(I+inv(H/h3)\*A)\*phi3 - inv(H/h3)\*f;

rn=norm(A\*phi3-f);

i=i+1;

end

rn3=rn;

i3=i;

rn=rn\_0;

i=0;

phi4=z;

%-----------------------------------iteration to residual reduced by 4 orders-----------------

while rn>rn\_0\*10^-4

phi4=(I+inv(H/h1)\*A)\*phi4 - inv(H/h1)\*f;

phi4=(I+inv(H/h2)\*A)\*phi4 - inv(H/h2)\*f;

phi4=(I+inv(H/h3)\*A)\*phi4 - inv(H/h3)\*f;

rn=norm(A\*phi4-f);

i=i+1;

end

rn4=rn;

i4=i;

%----------------plot of solution for residual reduction by 2, 3 and 4 orders---------------

figure(1)

plot(x,phi2,'g',x,phi3,'b--o',x,phi4,'r--\*')

xlabel('Domain 0<x<1')

ylabel('Solution u(x)')

legend('Residuals 2nd-Order','Residuals 3rd-Order','Residuals 4th-Order')

%-----------------Plots: logarithm of L2-norm of residuals vs iteration--------------------

figure(2)

rn\_vec=[log(rn2),log(rn3),log(rn4)];

i\_vec=[i2,i3,i4];

plot(i\_vec,rn\_vec,'-\*')

xlabel('Number of iterations')

ylabel('logarithm of L2-norm of residuals')

%---------------------- Theoretical Asymptotic Convergence Rate--------------------------

CR\_Richardson=cos(pi/(M+1))\*h1;

(d) 3-step Richardson Method

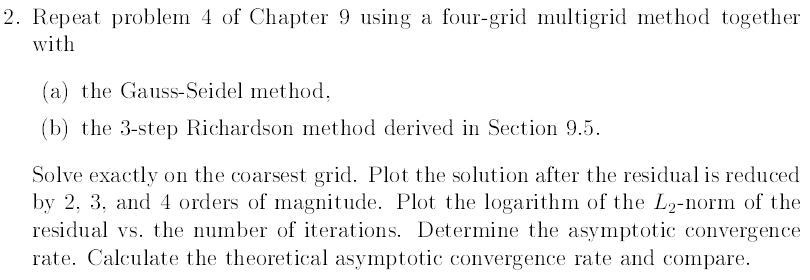




The theoretical asymptotic convergence rate:

|  |  |  |  |
| --- | --- | --- | --- |
| **Theoretical** | **2 Order** | **3 Order** | **4 Order** |
| 0.934 | 0.911 | 0.976 | 0.985 |

**Chapter 10**



Many iterative methods reduce error components (from transient modes) corresponding to eigenvalues of large amplitude more effectively than those corresponding to eigenvalues of small amplitudes. In the other words, reducing error components corresponding to eigenvalues of small amplitude is more difficult.



The concept of multigrid is that the error components corresponding to high frequencies (eigenvalues of large amplitude) will be reduced more quickly than those corresponding to low space frequencies. Although working with a coarse mesh is better for damping the error, the resolution is undesirable and we need to transfer the error to a fine mesh. Multigrid enables us to

* Solve the error on a coarse mesh
* And transfer it to a fine mesh

In this problem we follow the process in the book. We restrict/transfer the residual to the coarse mesh:



Solve the problem on a coarse mesh:



Prolong/transfer the error back to the fine grid and update the solution:



The basic iteration matrix for G12:





Substitute  into G12 for a three grid method:



Substitute  into G23 for a four-grid method:



% CFD\_ HW#4\_Chapter 10\_Question 2

% Banafsheh Zebhi

%----------------------------------(a) Gauss-Seidel Method----------------------------------

clc

clear all

close all

grids=4;

M\_0=39; % Grid of 40 cells = 39 nodes

dx=1/(M\_0+1); % Spatial differencing

x=[0+dx:dx:1-dx];

% Boundary Condition: u(0)=0, u(1)=1

bc=zeros(M\_0,1);

bc(M\_0)=1;

bc=(1/dx^2)\*bc;

g=6\*x; % source function

fb=g.'-bc; % eqn 9.18

phi1=zeros(M\_0,1);

% ------------------2nd-order central differencing matrix-----------------------------------

Ab = (1/dx^2)\*full(gallery('tridiag',M\_0,1,-2,1));

% Create pre-conditioning matrix C

C=eye(M\_0)\*dx^2;

A\_0=C\*Ab;

f=C\*fb;

%-------------B=Beta, w=omega in table 9.1----Gauss-Seidel method----------------------

beta=1; omega=1;

% ------------------------------------Create M, A and H matrices-----------------------------

for i=1:grids

if i==1

M(i)=39;

else

M(i)=(M(i-1)-1)/2;

end

A{i}=(1/4)^(i-1)\*full(gallery('tridiag',M(i),1,-2,1));

H{i}=full(gallery('tridiag',M(i),-beta,2/omega,0));

G{i}=eye(M(i))+H{i}\A{i};

end

% ---------------------Create restriction and prolongation matrices --------------------

for i=1:1:grids-1

R{i}=zeros(M(i+1),M(i));

I{i}=zeros(M(i),M(i+1));

for j=1:M(i)

if mod(j,2)==0

R{i}(j/2,j)=1;

I{i}(j,j/2)=1;

I{i}(j-1,j/2)=1/2;

I{i}(j+1,j/2)=1/2;

end

end

end

%---------I{1}=I\_21, I{2}=I\_2, I{3}=I\_43--------------------

%---------R{1}=R\_12, R{2}=R\_23, R{3}=R\_34-------------

% --------------Create M, A and H matrices-------------------

for i=grids-1:-1:1

if i==grids-1

Asub=1;

else

Asub=eye(M(i+1))-G\_iters{i+1};

end

G\_iters{i}=(eye(M(i))-I{i}\*Asub\*inv(A{i+1})\*R{i}\*A{i})\*G{i};

end

%-----------------------------G{3}=G\_43, G{2}=G\_32, G{1}=G\_21-------------------------

rn\_0=norm(A{1}\*zeros(M(1))-f);

rn=rn\_0;

i=0;

phi2=zeros(M(1));

%----------------------------------iteration to residual reduced by 2 orders--------------------

while rn>rn\_0\*10^-2

phi2=G\_iters{1}\*phi2-G\_iters{1}\*inv(A{1})\*f+inv(A{1})\*f;

rn=norm(A{1}\*phi2-f);

i=i+1;

end

rn2=rn;

i2=i;

rn=rn\_0;

i=0;

phi3=zeros(M(1));

%----------------------------------iteration to residual reduced by 3 orders--------------------

while rn>rn\_0\*10^-3

phi3=G\_iters{1}\*phi3-G\_iters{1}\*inv(A{1})\*f+inv(A{1})\*f;

rn=norm(A{1}\*phi3-f);

i=i+1;

end

rn3=rn;

i3=i;

rn=rn\_0;

i=0;

phi4=zeros(M(1));

%----------------------------------iteration to residual reduced by 2 orders--------------------

while rn>rn\_0\*10^-4

phi4=G\_iters{1}\*phi4-G\_iters{1}\*inv(A{1})\*f+inv(A{1})\*f;

rn=norm(A{1}\*phi4-f);

i=i+1;

end

rn4=rn;

i4=i;

%---------------------------------- Convergence Rates--------------------------------------------

CR2=(rn2/rn\_0)^(1/i2)

CR3=(rn3/rn\_0)^(1/i3)

CR4=(rn4/rn\_0)^(1/i4)

figure(1)

plot(x,phi2,'g',x,phi3,'b--o',x,phi4,'r--\*')

xlabel('x')

ylabel('Solution u(x)')

legend('Residuals 2nd-Order','Residuals 3rd-Order','Residuals 4th-Order')

figure(2)

rn\_vec=[log(rn2),log(rn3),log(rn4)];

i\_vec=[i2,i3,i4];

plot(i\_vec,rn\_vec,'-\*')

xlabel('Number of iterations')

ylabel('logarithm of L2-norm of residuals')

1. Gauss-Seidel Method



Convergence rate:

|  |  |  |  |
| --- | --- | --- | --- |
| **Theoretical** | **2 Order** | **3 Order** | **4 Order** |
| 0.996 | 0.841 | 0.887 | 0.902 |

%---------------------------------- (b) 3-step Richardson Method------------------------------

clc

clear all

close all

grids=4;

M\_0=39; % Grid of 40 cells = 39 nodes

dx=1/(M\_0+1); % Spatial differencing

x=[0+dx:dx:1-dx];

% Boundary Condition: u(0)=0, u(1)=1

bc=zeros(M\_0,1);

bc(M\_0)=1;

bc=(1/dx^2)\*bc;

g=6\*x; % source function

fb=g.'-bc; % eqn 9.18

phi1=zeros(M\_0,1);

% ------------------2nd-order central differencing matrix-----------------------------------

Ab = (1/dx^2)\*full(gallery('tridiag',M\_0,1,-2,1));

% Create pre-conditioning matrix C

C=eye(M\_0)\*dx^2;

A\_0=C\*Ab;

f=C\*fb;

%-------------B=Beta, w=omega in table 9.1--- 3-step Richardson method----------------

beta=0; omega=1;

h1=4/(6-sqrt(3));

h2=4/6;

h3=4/(6+sqrt(3));

h=full(gallery('tridiag',M\_0,-beta,2/omega,0));

% ------------------------------------Create M, A and H matrices-----------------------------

for i=1:grids

if i==1

M(i)=39;

else

M(i)=(M(i-1)-1)/2;

end

A{i}=(1/4)^(i-1)\*full(gallery('tridiag',M(i),1,-2,1));

H{i}=full(gallery('tridiag',M(i),-beta,2/omega,0));

G{i}=eye(M(i))+H{i}\A{i};

end

% ---------------------Create restriction and prolongation matrices -------------------------

for i=1:1:grids-1

R{i}=zeros(M(i+1),M(i));

I{i}=zeros(M(i),M(i+1));

for j=1:M(i)

if mod(j,2)==0

R{i}(j/2,j)=1;

I{i}(j,j/2)=1;

I{i}(j-1,j/2)=1/2;

I{i}(j+1,j/2)=1/2;

end

end

end

%---------I{1}=I\_21, I{2}=I\_2, I{3}=I\_43--------------------

%---------R{1}=R\_12, R{2}=R\_23, R{3}=R\_34-------------

% --------------Create M, A and H matrices-------------------

for i=grids-1:-1:1

if i==grids-1

Asub=1;

else

Asub=eye(M(i+1))-G\_iters{i+1};

end

G\_iters{i}=(eye(M(i))-I{i}\*Asub\*inv(A{i+1})\*R{i}\*A{i})\*G{i};

end

%-----------------------------G{3}=G\_43, G{2}=G\_32, G{1}=G\_21-------------------------

rn\_0=norm(A{1}\*zeros(M(1))-f);

rn=rn\_0;

i=0;

phi2=zeros(M(1));

%----------------------------------iteration to residual reduced by 2 orders--------------------

while rn>rn\_0\*10^-2

phi2=(eye(M\_0)+inv(h/h1)\*A\_0)\*phi2 - inv(h/h1)\*f;

phi2=(eye(M\_0)+inv(h/h2)\*A\_0)\*phi2 - inv(h/h2)\*f;

phi2=(eye(M\_0)+inv(h/h3)\*A\_0)\*phi2 - inv(h/h3)\*f;

phi2=G\_iters{1}\*phi2-G\_iters{1}\*inv(A{1})\*f+inv(A{1})\*f;

rn=norm(A{1}\*phi2-f);

i=i+1;

end

rn2=rn;

i2=i;

rn=rn\_0;

i=0;

phi3=zeros(M(1));

%----------------------------------iteration to residual reduced by 3 orders--------------------

while rn>rn\_0\*10^-3

phi3=(eye(M\_0)+inv(h/h1)\*A\_0)\*phi3 - inv(h/h1)\*f;

phi3=(eye(M\_0)+inv(h/h2)\*A\_0)\*phi3 - inv(h/h2)\*f;

phi3=(eye(M\_0)+inv(h/h3)\*A\_0)\*phi3 - inv(h/h3)\*f;

phi3=G\_iters{1}\*phi3-G\_iters{1}\*inv(A{1})\*f+inv(A{1})\*f;

rn=norm(A{1}\*phi3-f);

i=i+1;

end

rn3=rn;

i3=i;

rn=rn\_0;

i=0;

phi4=zeros(M(1));

%----------------------------------iteration to residual reduced by 4 orders--------------------

while rn>rn\_0\*10^-4

phi4=(eye(M\_0)+inv(h/h1)\*A\_0)\*phi4 - inv(h/h1)\*f;

phi4=(eye(M\_0)+inv(h/h2)\*A\_0)\*phi4 - inv(h/h2)\*f;

phi4=(eye(M\_0)+inv(h/h3)\*A\_0)\*phi4 - inv(h/h3)\*f;

phi4=G\_iters{1}\*phi4-G\_iters{1}\*inv(A{1})\*f+inv(A{1})\*f;

rn=norm(A{1}\*phi4-f);

i=i+1;

end

rn4=rn;

i4=i;

%---------------------------------- Convergence Rates--------------------------------------------

CR2=(rn2/rn\_0)^(1/i2)

CR3=(rn3/rn\_0)^(1/i3)

CR4=(rn4/rn\_0)^(1/i4)

figure(1)

plot(x,phi2,'g',x,phi3,'b--o',x,phi4,'r--\*')

xlabel('x')

ylabel('Solution u(x)')

legend('Residuals 2nd-Order','Residuals 3rd-Order','Residuals 4th-Order')

figure(2)

rn\_vec=[log(rn2),log(rn3),log(rn4)];

i\_vec=[i2,i3,i4];

plot(i\_vec,rn\_vec,'-\*')

xlabel('Number of iterations')

ylabel('Logarithm of L2-norm of residuals')

(b) 3-step Richardson Method



Convergence rate:

|  |  |  |  |
| --- | --- | --- | --- |
| **Theoretical** | **2 Order** | **3 Order** | **4 Order** |
| 0.934 | 0.574 | 0.679 | 0.730 |