Contents


```
%%***************MULTIGIRD PROJECT******************
%%******SUBMITTED BY GHANESH NARASIMHAN*************
%%*******NUMERICAL METHODS (FALL 2018)*************

clear
```

Multigrid parameters

```
Nlvlmax=5;            %Number of multigrid levels
Ncycle=50;           %Number of MG cycles
%tot_iter=2500;       %total iterations=Ncycle*maxiter*Nlvlmax;
maxiter=5;%tot_iter/(Nlvlmax*Ncycle);%Maximum iterations in iterative solver

omg=1.00;            %SOR parameter
Vcycle=1;            %Flag for V-Cycle
Wcycle=0;            %Flag for W-Cycle


%Type of smoother
itype=3;
%
%itype=1 Point Gauss-Seidel red black + SOR + MG
%itype=2 Point Gauss-Seidel normal    + SOR + MG
%itype=3 Line  Gauss-Seidel (ADI)     + SOR + MG
%itype=4 Line  Gauss-Seidel (ADI)     + SOR + Non-MG
%itype=5 Point Gauss-Seidel           + SOR + Non-MG
```

Grid definition for each level

```
Lx=2*pi;Ly=2*pi;
Nx=257;Ny=257;
nx(1)=Nx;
ny(1)=Ny;

%Number of grid points at other levels
```

```
for i=2:Nlvlmax
    ny(i)=((ny(1)-1)/2^(i-1))+1;
    nx(i)=((nx(1)-1)/2^(i-1))+1;
end
```

Initialise arrays on each level

```
error(1:Ncycle)=0;
for i=1:Nlvlmax
    uin{i}(1:ny(i),1:nx(i))=0;
    uout{i}(1:ny(i),1:nx(i))=0;
    uoutnew{i}(1:ny(i),1:nx(i))=0;
    eps{i}(1:ny(i),1:nx(i))=0;
    epsnew{i}(1:ny(i),1:nx(i))=0;
    RHS{i}(1:ny(i),1:nx(i))=0;
end
```

Initial condition (random noise+ boundary condition)

```
for n=1:Nlvlmax
  dx=Lx/(nx(n)-1);dy=Ly/(ny(n)-1);
  x=0:dx:Lx;
  y=0:dy:Ly;
  if (n==1)
   %uin{n}= -1 + (1+1)*rand(ny(1),nx(1));
   u=load('init_cond.mat');
   uin{n}(1:ny(n),1:nx(n))=u.u(1:ny(n),1:nx(n));
   uin{n}(1:ny(n),1)=sin(4*y);
   uin{n}(1:ny(n),nx(n))=0;
   uin{n}(1,1:nx(n))=sin(4*x);
   uin{n}(ny(n),1:nx(n))=0;
  end
end
```

Calculate co-efficients

```
a(1:Nlvlmax)=0;b(1:Nlvlmax)=0;c(1:Nlvlmax)=0;
for n=1:Nlvlmax
    A{n}=0;B{n}=0;invA{n}=0;invB{n}=0;
    [invA{n},invB{n},A{n},B{n},a(n),b(n),c(n)]=coeff(n,Nx,Ny,Lx,Ly);
end
```

Solving Lu=R using Non-MG

```
if (itype>=4)
  n=1;tol=1e-5;counternMG=0;
```

```
  [uout{n},counternMG,tnonMG,errornMG]=NonMG(uin{n},RHS{n},invA{n},invB{n}...
                                              ,a(n),b(n),c(n),Nx,Ny,itype,omg,tol);
  semilogy(errornMG)
end
```

Solving Lu=R using MG (V-cycle)

```
if (Vcycle==1 && itype<4)
   tsV=cputime;
   for Ncyl=1:Ncycle
       Ncyl
      [uout]=fine_to_coarse(uin,RHS,maxiter,invA,invB,A,B,a,b,c,...
       nx,ny,itype,omg,Nlvlmax,1,uout,eps,epsnew);
      [uout]=coarse_to_fine(Nlvlmax,uout,uoutnew,2);
      uin{1}=uout{1};
      %residue for testing convergence
      error(Ncyl)=norm(residual(uout{1},RHS{1},a(1),b(1),c(1),nx(1),ny(1)));
      if (error(Ncyl)<1e-5)
          Ncyl
          teV=cputime-tsV;
          break
      end
   end
end
```

Solving Lu=R using MG (W-cycle)

```
if (Wcycle==1 && itype<4 && Nlvlmax>3)
   tsW=cputime;
   for Ncyl=1:Ncycle
       Ncyl
      [uout]=fine_to_coarse(uin,RHS,maxiter,invA,invB,A,B,a,b,c,nx,ny,itype,...
                            omg,Nlvlmax,1,uout,eps,epsnew);
      [uout]=coarse_to_fine(Nlvlmax,uout,uoutnew,Nlvlmax-1);
      uin{Nlvlmax-2}=uout{Nlvlmax-2};
      [uout]=fine_to_coarse(uin,RHS,maxiter,invA,invB,A,B,a,b,c,nx,ny,itype,omg,...
                            Nlvlmax,Nlvlmax-2,uout,eps,epsnew);
      [uout]=coarse_to_fine(Nlvlmax,uout,uoutnew,2);
      uin{1}=uout{1};
      %residue for testing convergence
      error(Ncyl)=norm(residual(uout{1},RHS{1},a(1),b(1),c(1),nx(1),ny(1)));
   end
   teW=cputime-tsW;
end
```

Analysis

```
figure(3)
Lx=2*pi;Ly=2*pi;
dx=Lx/(nx(1)-1);dy=Ly/(ny(1)-1);
x=0:dx:2*pi;
y=0:dy:2*pi;
[X,Y]=meshgrid(x,y);
surf(x,y,uout{1},'linestyle','none')
set(gca, 'CameraPosition', [2*pi 2*pi 0.25]);
xlabel('$x$','interpreter','latex','fontsize',16)
ylabel('$y$','interpreter','latex','fontsize',16)
title('$u$ (Line GS)','interpreter','latex','fontsize',16)
set(gcf,'Color','w')
set(gca,'fontsize',16,'fontname','times')
colorbar
colormap(flipud(gray))

figure(1)
semilogy(error);hold on
xlabel('Number of cycles','interpreter','latex','fontsize',16)
ylabel('$\epsilon =|\!|\nabla^2u-R|\!|$','interpreter','latex','fontsize',16)
title('Point \ GS','interpreter','latex','fontsize',16)
set(gcf,'Color','w')
set(gca,'fontsize',16,'fontname','times')
```

Subroutines

Iterative Solve

1. `function [uout]=iterative_solve(uin,RHS,maxiter,...`
   `invA,invB,A,B,a,b,c,Nx,Ny,itype,omg)`

Point Gauss-Seidel + MG (red-black+SOR)

```
if (itype==1)
uout(1:Ny,1:Nx)=0;
utemp(1:Ny,1:Nx,1:2)=0;
utemp(1:Ny,1:Nx,1)=uin(1:Ny,1:Nx);
utemp(1,:,2)=utemp(1,:,1);
utemp(Ny,:,2)=utemp(Ny,:,1);
utemp(:,1,2)=utemp(:,1,1);
utemp(:,Nx,2)=utemp(:,Nx,1);

for iter=1:maxiter
```

```
for i=2:Nx-1
for j=2:Ny-1
if (mod(i+j,2)~=0)
utemp(j,i,2)=RHS(j,i)*(1/b)-(a/b)*(utemp(j,i-1,1)+utemp(j,i+1,1))...
-(c/b)*(utemp(j-1,i,1)+utemp(j+1,i,1));
end
end
end

for i=2:Nx-1
for j=2:Ny-1
if (mod(i+j,2)==0)
utemp(j,i,2)=RHS(j,i)*(1/b)-(a/b)*(utemp(j,i-1,2)+utemp(j,i+1,2))...
-(c/b)*(utemp(j-1,i,2)+utemp(j+1,i,2));
end
end
end
%SOR
utemp(2:Ny-1,2:Nx-1,2)=utemp(2:Ny-1,2:Nx-1,2)*omg+...
utemp(2:Ny-1,2:Nx-1,2)*(1-omg);
utemp(1:Ny,1:Nx,1)=utemp(1:Ny,1:Nx,2);
end
uout(1:Ny,1:Nx)=utemp(1:Ny,1:Nx,2);%utemp copied with boundary values
end
```

Point Gauss-Seidel + MG (normal)

```
if (itype==2)
utemp(1:Ny,1:Nx,1:2)=0;
utemp(1:Ny,1:Nx,1  )=uin(1:Ny,1:Nx);
utemp(1   ,:   ,2  )=utemp(1 ,: ,1);
utemp(Ny  ,:   ,2  )=utemp(Ny,: ,1);
utemp(:   ,1   ,2  )=utemp(: ,1 ,1);
utemp(:   ,Nx  ,2  )=utemp(: ,Nx,1);
uout(1:Ny,1:Nx     )=0;
for iter=1:maxiter
for i=2:Nx-1
for j=2:Ny-1
utemp(j,i,2)=RHS(j,i)*(1/b)-(a/b)*(utemp(j,i-1,2)+utemp(j,i+1,1))...
-(c/b)*(utemp(j-1,i,2)+utemp(j+1,i,1));
end
end
%SOR
utemp(2:Ny-1,2:Nx-1,2)=utemp(2:Ny-1,2:Nx-1,2)*omg...
+utemp(2:Ny-1,2:Nx-1,2)*(1-omg);
```

```
utemp(1:Ny,1:Nx,1)=utemp(1:Ny,1:Nx,2);
end
uout(1:Ny,1:Nx)=utemp(1:Ny,1:Nx,2);%utemp copied with boundary values

end
```

Line-SOR ADI + MG

```
if (itype==3)
utemp(1:Ny,1:Nx,1:3)=0;
utemp(1:Ny,1:Nx,1  )=uin(1:Ny,1:Nx);
utemp(1   ,:   ,2  )=utemp(1 ,: ,1);
utemp(Ny  ,:   ,2  )=utemp(Ny,: ,1);
utemp(:   ,1   ,2  )=utemp(: ,1 ,1);
utemp(:   ,Nx  ,2  )=utemp(: ,Nx,1);

utemp(1   ,:   ,3  )=utemp(1 ,: ,1);
utemp(Ny  ,:   ,3  )=utemp(Ny,: ,1);
utemp(:   ,1   ,3  )=utemp(: ,1 ,1);
utemp(:   ,Nx  ,3  )=utemp(: ,Nx,1);

for iter=1:maxiter
for j=2:Ny-1
BCmatx(1,1:Nx-2)=0;
BCmatx(1,   1)=-a*utemp(j ,1 ,1);
BCmatx(1,Nx-2)=-a*utemp(j ,Nx,1);
Atemp(1:Nx-2,1)=RHS(j,2:Nx-1)-c*(utemp(j+1,2:Nx-1,1)...
+utemp(j-1,2:Nx-1,2))+BCmatx(1,1:Nx-2);
utemp(j,2:Nx-1,2)=mtimes(invB,Atemp);
end

for i=2:Nx-1
BCmaty(1:Ny-2,1)=0;
BCmaty(1,   1)=-c*utemp(1 ,i,1);
BCmaty(Ny-2,1)=-c*utemp(Ny,i,1);
Btemp(1:Ny-2,1)=RHS(2:Ny-1,i)-a*(utemp(2:Ny-1,i-1,3)...
+utemp(2:Ny-1,i+1,2))+BCmaty(1:Ny-2,1);
utemp(2:Ny-1,i,3)=mtimes(invA,RHS(2:Ny-1,i)...
-a*(utemp(2:Ny-1,i-1,3)+utemp(2:Ny-1,i+1,2))+BCmaty(1:Ny-2,1));

%SOR
utemp(2:Ny-1,2:Nx-1,3)=utemp(2:Ny-1,2:Nx-1,1)*(1-omg)+utemp(2:Ny-1,2:Nx-1,3)*omg;
utemp(1:Ny,1:Nx,1)=utemp(1:Ny,1:Nx,3);
end
```

```
uout(1:Ny,1:Nx)=utemp(1:Ny,1:Nx,3);...

%utemp copied with boundary values
end
end
```

Fine to coarse

2.
```
function [uout]=fine_to_coarse(uin,RHS,maxiter,invA,invB,A,B,a,b,c,nx,ny,...
itype,omg,Nlvlmax,Nlvlmin,uout,eps,epsnew)

%Fine to coarse
for n=Nlvlmin:Nlvlmax
fprintf('na=%d \n',n)
%Solve for Lu=R iteratively "maxiter" times
uout{n}=iterative_solve(uin{n},RHS{n},maxiter,invA{n},...
invB{n},A{n},B{n},a(n),b(n),c(n),nx(n),ny(n),itype,omg);
%Compute the residual from the smoothened solution
eps{n}=residual(uout{n},RHS{n},a(n),b(n),c(n),nx(n),ny(n));
%Define the new RHS for the next level
if(n~=Nlvlmax)
epsnew{n+1}=restriction(eps{n},nx(n),ny(n));
RHS{n+1}=-epsnew{n+1};
uin{n+1}(1:ny(n+1),1:nx(n+1))=0;
end
end


end
```

3. ### Coarse to fine

```
function [uout]=coarse_to_fine(Nlvlmax,uout,uoutnew,Nlvlmin)
%Coarse to fine
for n=Nlvlmax:-1:Nlvlmin
fprintf('nb=%d \n',n-1)
uoutnew{n-1}=prolongation(uout{n});
uout{n-1}=uout{n-1}+uoutnew{n-1};
end


end
```

Restriction

4.
```
function [epsnew]=restriction(eps,Nx,Ny)
Nxnew=(Nx-1)/2+1;
Nynew=(Ny-1)/2+1;

epsnew(1:Nynew,1:Nxnew)=0;
%Perform restriction by copying every alternate points in the grid
epsnew(1:Nynew,1:Nxnew)=eps(1:2:Ny,1:2:Nx);
end
```

5. ## Prolongation

```
function [uoutnew]=prolongation(uout)
[Ny,Nx]=size(uout);
Nxnew=(Nx-1)*2+1;
Nynew=(Ny-1)*2+1;

%Prolongation step
uoutnew(1:Nynew,1:Nxnew)=0;
uoutnew(1:Nynew,1:Nxnew)=interp2(uout);


end
```

Residual

6.
```
function [eps]=residual(uout,RHS,a,b,c,Nx,Ny)
eps(1:Ny,1:Nx)=0;
%Compute residual \epsilon=Lu-R
for i=2:Nx-1
for j=2:Ny-1
eps(j,i)=a*(uout(j,i-1)+uout(j,i+1))+...
c*(uout(j-1,i)+uout(j+1,i))+b*uout(j,i)-RHS(j,i);
end
end

end
```

7. ## Non-Multigrid iterative solvers

Line-SOR ADI (non-MG)

```
if (itype==4)
```

```
utemp(1:Ny,1:Nx,1:3)=0;
utemp(1:Ny,1:Nx,1  )=uin(1:Ny,1:Nx);
utemp(1   ,:   ,2  )=utemp(1 ,: ,1);
utemp(Ny  ,:   ,2  )=utemp(Ny,: ,1);
utemp(:   ,1   ,2  )=utemp(: ,1 ,1);
utemp(:   ,Nx  ,2  )=utemp(: ,Nx,1);

utemp(1   ,:   ,3  )=utemp(1 ,: ,1);
utemp(Ny  ,:   ,3  )=utemp(Ny,: ,1);
utemp(:   ,1   ,3  )=utemp(: ,1 ,1);
utemp(:   ,Nx  ,3  )=utemp(: ,Nx,1);
error=1;
counter=0;
ts=cputime;
while(error>tol)
counter=counter+1
for j=2:Ny-1
BCmatx(1,1:Nx-2)=0;
BCmatx(1,   1)=-a*utemp(j ,1 ,1);
BCmatx(1,Nx-2)=-a*utemp(j ,Nx,1);
Atemp(1:Nx-2,1)=RHS(j,2:Nx-1)-c*(utemp(j+1,2:Nx-1,1)...
+utemp(j-1,2:Nx-1,2))+BCmatx(1,1:Nx-2);
utemp(j,2:Nx-1,2)=mtimes(invB,Atemp);
end

for i=2:Nx-1
BCmaty(1:Ny-2,1)=0;
BCmaty(1,   1)=-c*utemp(1 ,i,1);
BCmaty(Ny-2,1)=-c*utemp(Ny,i,1);
Btemp(1:Ny-2,1)=RHS(2:Ny-1,i)-a*(utemp(2:Ny-1,i-1,3)...
+utemp(2:Ny-1,i+1,2))+BCmaty(1:Ny-2,1);
utemp(2:Ny-1,i,3)=mtimes(invA,Btemp);
end
%SOR
utemp(2:Ny-1,2:Nx-1,3)=  utemp(2:Ny-1,2:Nx-1,1)*(1-omg)...
+utemp(2:Ny-1,2:Nx-1,3)*omg;
%Stopping criteria
error=norm(residual(utemp(1:Ny,1:Nx,3),RHS,a,b,c,Nx,Ny))
errora(counter)=error;
if (error>tol)
utemp(2:Ny-1,2:Nx-1,1)=utemp(2:Ny-1,2:Nx-1,3);
else
uout(1:Ny,1:Nx)=utemp(1:Ny,1:Nx,3);
end
end
```

```
te=cputime-ts;

end
```

Point GS+SOR

```
if (itype==5)

utemp(1:Ny,1:Nx,1:2)=0;
utemp(1:Ny,1:Nx,1  )=uin(1:Ny,1:Nx);
utemp(1   ,:   ,2  )=utemp(1 ,: ,1);
utemp(Ny  ,:   ,2  )=utemp(Ny,: ,1);
utemp(:   ,1   ,2  )=utemp(: ,1 ,1);
utemp(:   ,Nx  ,2  )=utemp(: ,Nx,1);

error=1;
counter=0;
ts=cputime;
while(error>tol)
counter=counter+1
for i=2:Nx-1
for j=2:Ny-1
utemp(j,i,2)=RHS(j,i)*(1/b)-(a/b)*(utemp(j,i-1,2)+utemp(j,i+1,1))...
-(c/b)*(utemp(j-1,i,2)+utemp(j+1,i,1));
end
end
%SOR
utemp(2:Ny-1,2:Nx-1,2)=   utemp(2:Ny-1,2:Nx-1,2)*omg...
+utemp(2:Ny-1,2:Nx-1,2)*(1-omg);
%Stopping criteria
error=norm(residual(utemp(1:Ny,1:Nx,2),RHS,a,b,c,Nx,Ny))
errora(counter)=error;
if (error>tol)
utemp(2:Ny-1,2:Nx-1,1)=utemp(2:Ny-1,2:Nx-1,2);
else
uout(1:Ny,1:Nx)=utemp(1:Ny,1:Nx,2);
end
end
te=cputime-ts;
end

end
```