```
%*************************************************************************
%*     Example 14.1                                                      *
%*     filename: ch14pr01.m                                              *
%*     program listing number: 14.1                                      *
%*                                                                       *
%*     This program solves 2-dimensional Laplace equation using Jacobi   *
%*     method.                                                           *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  04/16/2017.                                   *
%*************************************************************************
close all;
clear all;

% parameters
a=1.0;
b=1.0;
V=1.0;

% spacial domain
Nx=201; % number of grids
Ny=101;
dx=0.1; % spacial step
dy=0.1;
x=linspace(-b,b,Nx);
y=linspace(0,a,Ny);

% time step
h=1./4.;

%tolerence
tol=1.e-9;

% sampling interval
M=10;

% initial profile
phi0=zeros(Ny,Nx);
phi0(:,1)=V;
phi0(:,Nx)=V;
phi0(1,:)=0;
phi0(Ny,:)=0;

% allocate arrays
phi1=phi0;

figure(1)
k=0;
diff=realmax();
while diff>tol
    k=k+1;
    for i=2:Ny-1
        for j=2:Nx-1
            phi1(i,j)=h*(phi0(i-1,j)+phi0(i+1,j)+phi0(i,j-1)+phi0(i,j+1));
        end
    end
    if mod(k,M)==0  % record the results
        s=(phi1-phi0).^2;
        diff=sum(s(:));
        fprintf('%d : diff=%14.6e\n',k,diff);
        pcolor(phi1); axis equal tight; shading interp;
        xlabel('x');
        ylabel('y');
        drawnow;
```

```
    end
    phi0=phi1;
end
colorbar

% Plot time evolution as cuntour
figure(2);
contour(x,y,phi1);
hold on;
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1));
[GX,GY]=gradient(phi1);
G=sqrt(GY.^2+GY.^2);
GX=GX./G;GY=GY./G;
quiver(X,Y,GX(6:10:Ny-1,6:10:Nx-1),GY(6:10:Ny-1,6:10:Nx-1),2)
hold off
axis equal tight;
xlabel('x');
ylabel('y');
```

```
%*************************************************************************
%*     Example 14.2                                                      *
%*     filename: ch14pr02.m                                              *
%*     program listing number: 14.2                                      *
%*                                                                       *
%*     This program solves 2-dimensional Laplace equation using the      *
%*     Gauss-Seidel method.                                              *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  04/16/2017.                                   *
%*************************************************************************
close all;
clear all;

% parameters
a=1.0;
b=1.0;
V=1.0;

% spacial domain
Nx=201; % number of grids
Ny=101;
dx=0.1; % spacial step
dy=0.1;
x=linspace(-b,b,Nx);
y=linspace(0,a,Ny);

% time step
h=1./4.;

%tolerence
tol=1.e-9;

% sampling interval
M=10;

% initial profile
phi0=zeros(Ny,Nx);
phi0(:,1)=V;
phi0(:,Nx)=V;
phi0(1,:)=0;
phi0(Ny,:)=0;

% allocate arrays
phi1=phi0;

k=0;
diff=realmax();
figure(1)
while diff>tol
    k=k+1;
    for i=2:Ny-1
        for j=2:Nx-1
            phi1(i,j)=h*(phi1(i-1,j)+phi0(i+1,j)+phi1(i,j-1)+phi0(i,j+1));
        end
    end
    if mod(k,M)==0  % record the results
        s=(phi1-phi0).^2;
        diff=sum(s(:));
        fprintf('%d : diff=%14.6e\n',k,diff);
        pcolor(phi1); axis equal tight; shading interp;
        xlabel('x');
        ylabel('y');
        drawnow;
```

```
    end
    phi0=phi1;
end
colorbar

% Plot time evolution as 3D plot
figure(2);
contour(x,y,phi1);
hold on;
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1));
[GX,GY]=gradient(phi1);
G=sqrt(GY.^2+GY.^2);
GX=GX./G;GY=GY./G;
quiver(X,Y,GX(6:10:Ny-1,6:10:Nx-1),GY(6:10:Ny-1,6:10:Nx-1),2)
hold off
axis equal tight;
xlabel('x');
ylabel('y');
```

```
%***********************************************************************
%*     Example 14.3                                                    *
%*     filename: ch14pr03.m                                            *
%*     program listing number: 14.3                                    *
%*                                                                     *
%*     This program solves 2-dimensional Laplace equation using the SOR *
%*     method.                                                         *
%*                                                                     *
%*     Programed by Ryoichi Kawai for Computational Physics Course.    *
%*     Last modification:  04/16/2017.                                 *
%***********************************************************************
close all;
clear all;

% parameters
a=1.0;
b=1.0;
V=1.0;

% spacial domain
Nx=201; % number of grids
Ny=101;
dx=0.1; % spacial step
dy=0.1;
x=linspace(-b,b,Nx);
y=linspace(0,a,Ny);

% time step
h=1./4.;

% SOR parameter
r=0.5*(cos(pi/Nx)+cos(pi/Ny));
w=2./(1.+sqrt(1-r^2));

%tolerence
tol=1.e-9;

% sampling interval
M=10;

% initial profile
phi0=zeros(Ny,Nx);
phi0(:,1)=V;
phi0(:,Nx)=V;
phi0(1,:)=0;
phi0(Ny,:)=0;

% allocate arrays
phi1=phi0;

k=0;
diff=realmax();
figure(1)
while diff>tol
    k=k+1;
    for i=2:Ny-1
        for j=2:Nx-1
            phi1(i,j)=(1.0-w)*phi0(i,j)+w*h*(phi1(i-1,j)+phi0(i+1,j)+phi1(i,j-1)+phi0(i,j+1));
        end
    end
    if mod(k,M)==0  % record the results
        s=(phi1-phi0).^2;
        diff=sum(s(:));
        fprintf('%d : diff=%14.6e\n',k,diff);
```

```
        pcolor(phi1); axis equal tight; shading interp;
        xlabel('x');
        ylabel('y');
        drawnow;
    end
    phi0=phi1;
end
colorbar

% Plot time evolution as 3D plot
figure(2)
contour(x,y,phi1);
hold on;
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1));
[GX,GY]=gradient(phi1);
G=sqrt(GX(6:10:Ny-1,6:10:Nx-1).^2+GY(6:10:Ny-1,6:10:Nx-1).^2);
VX=GX(6:10:Ny-1,6:10:Nx-1)./G;VY=GY(6:10:Ny-1,6:10:Nx-1)./G;
quiver(X,Y,VX,VY,0.5)
hold off
axis equal tight;
xlabel('x');
ylabel('y');
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
#***********************************************************************
#*     Example 14.1                                                    *
#*     filename: ch14pr01.m                                            *
#*     program listing number: 14.1                                    *
#*                                                                     *
#*     This program solves 2-dimensional Laplace equation using Jacobi *
#*     method.  (Too slow for Python)                                  *
#*                                                                     *
#*     Programed by Ryoichi Kawai for Computational Physics Course.    *
#*     Last modification:  04/16/2017.                                 *
#***********************************************************************
"""
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

# parameters
a=1.0
b=1.0
V=1.0

# spacial domain
Nx=201 # number of grids
Ny=101
dx=0.1 # spacial step
dy=0.1
x=np.linspace(-b,b,Nx)
y=np.linspace(0,a,Ny)

# time step
h=1./4.

#tolerence
tol=1.e-9

# sampling interval
M=10

phi0=None
phi1=None

# initial profile
phi0=np.zeros((Ny,Nx))
phi1=np.zeros((Ny,Nx))
phi1[:,0]=phi0[:,0]=V
phi1[:,-1]=phi0[:,-1]=V
phi1[0,:]=phi0[0,:]=0.0
phi1[0,:]=phi0[-1,:]=0.0

plt.close('all')
fig, ax =plt.subplots()
k=0
diff=tol+1.
while diff>tol:
    k=k+1
    for i in range(1,Ny-1):
        for j in range(1,Nx-1):
            phi1[i,j]=h*(phi0[i-1,j]+phi0[i+1,j]+phi0[i,j-1]+phi0[i,j+1])

    if np.mod(k,M)==0:  # record the results
        diff=np.sum((phi1[:,:]-phi0[:,:])**2)
```

```
        print('{0:d} : diff={1:14.6e}'.format(k,diff))
        cax = ax.imshow(phi1,extent=(-b,b,0.0,a))
        plt.pause(0.0001)

    phi0[:,:]=phi1[:,:]

c_min=phi1.min()
c_max=phi1.max()
print(c_max,c_min)

cbar=fig.colorbar(cax, ticks=[0.0,0.2,0.4,0.6,0.8,1.0])

"""
# Plot time evolution as cuntour
figure(2)
contour(x,y,phi1)
hold on
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1))
[GX,GY]=gradient(phi1)
G=sqrt(GY.^2+GY.^2)
GX=GX./GGY=GY./G
quiver(X,Y,GX(6:10:Ny-1,6:10:Nx-1),GY(6:10:Ny-1,6:10:Nx-1),2)
hold off
axis equal tight
xlabel('x')
ylabel('y')
"""
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
%***********************************************************************
%*     Example 14.2                                                    *
%*     filename: ch14pr02.m                                            *
%*     program listing number: 14.2                                    *
%*                                                                     *
%*     This program solves 2-dimensional Laplace equation using the    *
%*     Gauss-Seidel method.                                            *
%*                                                                     *
%*     Programed by Ryoichi Kawai for Computational Physics Course.    *
%*     Last modification:  04/16/2017.                                 *
%***********************************************************************
"""
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

# parameters
a=1.0
b=1.0
V=1.0

# spacial domain
Nx=201 # number of grids
Ny=101
dx=0.1 # spacial step
dy=0.1
x=np.linspace(-b,b,Nx)
y=np.linspace(0,a,Ny)

# time step
h=1./4.

#tolerence
tol=1.e-9

# sampling interval
M=10

phi0=None
phi1=None

# initial profile
phi0=np.zeros((Ny,Nx))
phi1=np.zeros((Ny,Nx))
phi1[:,0]=phi0[:,0]=V
phi1[:,-1]=phi0[:,-1]=V
phi1[0,:]=phi0[0,:]=0.0
phi1[0,:]=phi0[-1,:]=0.0

plt.close('all')
fig, ax =plt.subplots()
k=0
diff=tol+1.
while diff>tol:
    k=k+1
    for i in range(1,Ny-1):
        for j in range(1,Nx-1):
            phi1[i,j]=h*(phi1[i-1,j]+phi0[i+1,j]+phi1[i,j-1]+phi0[i,j+1])

    if np.mod(k,M)==0:  # record the results
        diff=np.sum((phi1[:,:]-phi0[:,:])**2)
```

```
        print('{0:d} : diff={1:14.6e}'.format(k,diff))
        cax = ax.imshow(phi1,extent=(-b,b,0.0,a))
        plt.pause(0.0001)

    phi0[:,:]=phi1[:,:]

c_min=phi1.min()
c_max=phi1.max()
print(c_max,c_min)

cbar=fig.colorbar(cax, ticks=[0.0,0.2,0.4,0.6,0.8,1.0])

"""
# Plot time evolution as cuntour
figure(2)
contour(x,y,phi1)
hold on
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1))
[GX,GY]=gradient(phi1)
G=sqrt(GY.^2+GY.^2)
GX=GX./GGY=GY./G
quiver(X,Y,GX(6:10:Ny-1,6:10:Nx-1),GY(6:10:Ny-1,6:10:Nx-1),2)
hold off
axis equal tight
xlabel('x')
ylabel('y')
"""
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
%*************************************************************************
%*     Example 14.3                                                      *
%*     filename: ch14pr03.m                                              *
%*     program listing number: 14.3                                      *
%*                                                                       *
%*     This program solves 2-dimensional Laplace equation using the SOR  *
%*     method.                                                           *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  04/16/2017.                                   *
%*************************************************************************
"""
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

# parameters
a=1.0
b=1.0
V=1.0

# spacial domain
Nx=201 # number of grids
Ny=101
dx=0.1 # spacial step
dy=0.1
x=np.linspace(-b,b,Nx)
y=np.linspace(0,a,Ny)

# time step
h=1./4.

# SOR parameter
r=0.5*(np.cos(np.pi/Nx)+np.cos(np.pi/Ny));
w=2./(1.+np.sqrt(1-r**2));

#tolerence
tol=1.e-9

# sampling interval
M=10

phi0=None
phi1=None

# initial profile
phi0=np.zeros((Ny,Nx))
phi1=np.zeros((Ny,Nx))
phi1[:,0]=phi0[:,0]=V
phi1[:,-1]=phi0[:,-1]=V
phi1[0,:]=phi0[0,:]=0.0
phi1[0,:]=phi0[-1,:]=0.0

plt.close('all')
fig, ax =plt.subplots()
k=0
diff=tol+1.
while diff>tol:
    k=k+1
    for i in range(1,Ny-1):
        for j in range(1,Nx-1):
```

```
            phi1[i,j]=(1.0-w)*phi0[i,j]\
                        +w*h*(phi1[i-1,j]+phi0[i+1,j]+phi1[i,j-1]+phi0[i,j+1])

    if np.mod(k,M)==0:  # record the results
        diff=np.sum((phi1[:,:]-phi0[:,:])**2)
        print('{0:d} : diff={1:14.6e}'.format(k,diff))
        cax = ax.imshow(phi1,extent=(-b,b,0.0,a))
        plt.pause(0.0001)

    phi0[:,:]=phi1[:,:]

c_min=phi1.min()
c_max=phi1.max()
print(c_max,c_min)

cbar=fig.colorbar(cax, ticks=[0.0,0.2,0.4,0.6,0.8,1.0])

"""
# Plot time evolution as cuntour
figure(2)
contour(x,y,phi1)
hold on
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1))
[GX,GY]=gradient(phi1)
G=sqrt(GY.^2+GY.^2)
GX=GX./GGY=GY./G
quiver(X,Y,GX(6:10:Ny-1,6:10:Nx-1),GY(6:10:Ny-1,6:10:Nx-1),2)
hold off
axis equal tight
xlabel('x')
ylabel('y')
"""
```