

- 3.1 a. Test this script by performing a grid refinement study to verify that it is second order accurate.
- b. Modify the script so that it works on a rectangular domain  $[a_x, b_x] \times [a_y, b_y]$ , but still with  $\Delta x = \Delta y = h$ . Test your modified script on a non-square domain.
- c. Further modify the code to allow  $\Delta x \neq \Delta y$  and test the modified script.

*Solution.*

Here, the code was edited to be in a for loop so that the  $m$  values would increase in size from 200 to 2000. We see that the order of accuracy converges to  $\mathcal{O}(h^2)$ .

The code here was edited to be a rectangular domain of  $[0, 1] \times [0, 2]$ . The code was also edited such that  $h$  was the same with  $h = 0.2$ .

The code here was only slightly edited such that  $h_x = 0.2$  and  $h_y = 0.2222$ .

```

1  % poisson2.m  -- solve the Poisson problem  $u_{xx} + u_{yy} = f(x,y)$ 
2  % on  $[a,b] \times [a,b]$ .
3  %
4  % The 5-point Laplacian is used at interior grid points.
5  % This system of equations is then solved using backslash.
6  %
7  % From http://www.amath.washington.edu/~rjl/fdmbook/chapter3 (2007)
8
9  clear;
10 close all;
11 clc;
12
13 for i=1:10
14
15     a = 0;
16     b = 1;
17     m = 20*10*i;
18     h = (b-a)/(m+1);
19     x = linspace(a,b,m+2);    % grid points x including boundaries
20     y = linspace(a,b,m+2);    % grid points y including boundaries
21
22
23     [X,Y] = meshgrid(x,y);    % 2d arrays of x,y values
24     X = X';                  % transpose so that X(i,j),Y(i,j) are
25     Y = Y';                  % coordinates of (i,j) point
26
27     Iint = 2:m+1;            % indices of interior points in x
28     Jint = 2:m+1;            % indices of interior points in y
29     Xint = X(Iint,Jint);     % interior points
30     Yint = Y(Iint,Jint);
31
32     f = @(x,y) 1.25*exp(x+y/2);    % f(x,y) function
33
34     rhs = f(Xint,Yint);          % evaluate f at interior points for ...
35     % right hand side
36     % rhs is modified below for boundary ...
37     % conditions.
38
39     utrue = exp(X+Y/2);          % true solution for test problem
40
41     % set boundary conditions around edges of usoln array:
42
43     usoln = utrue;              % use true solution for this test problem
44     % This sets full array, but only ...
45     % boundary values
46     % are used below. For a problem where ...
47     % utrue
48     % is not known, would have to set each ...
49     % edge of
50     % usoln to the desired Dirichlet ...
51     % boundary values.

```

```
48 % adjust the rhs to include boundary terms:
49 rhs(:,1) = rhs(:,1) - usoln(Iint,1)/h^2;
50 rhs(:,m) = rhs(:,m) - usoln(Iint,m+2)/h^2;
51 rhs(1,:) = rhs(1,:) - usoln(1,Jint)/h^2;
52 rhs(m,:) = rhs(m,:) - usoln(m+2,Jint)/h^2;
53
54
55 % convert the 2d grid function rhs into a column vector for rhs of ...
    system:
56 F = reshape(rhs,m*m,1);
57
58 % form matrix A:
59 I = speye(m);
60 e = ones(m,1);
61 T = spdiags([e -4*e e],[-1 0 1],m,m);
62 S = spdiags([e e],[-1 1],m,m);
63 A = (kron(I,T) + kron(S,I)) / h^2;
64
65
66 % Solve the linear system:
67 uvec = A\F;
68
69 % reshape vector solution uvec as a grid function and
70 % insert this interior solution into usoln for plotting purposes:
71 % (recall boundary conditions in usoln are already set)
72
73 usoln(Iint,Jint) = reshape(uvec,m,m);
74
75 % assuming true solution is known and stored in utrue:
76 E=usoln-uttrue;
77 err = max(max(abs(usoln-uttrue)));
78 fprintf('Error relative to true solution of PDE = %10.3e \n',err)
79
80 % plot results:
81
82 clf
83 hold on
84
85 % plot grid:
86 % plot(X,Y,'g'); plot(X',Y', 'g');
87
88 % plot solution:
89 contour(X,Y,usoln,30,'k')
90
91 axis([a b a b])
92 daspect([1 1 1])
93 title('Contour plot of computed solution')
94 hold on
95 end
96 error_loglog(h, err);
97 hold on
```

```

1
2 % poisson2.m -- solve the Poisson problem  $u_{xx} + u_{yy} = f(x,y)$ 
3 % on  $[a,b] \times [a,b]$ .
4 %
5 % The 5-point Laplacian is used at interior grid points.
6 % This system of equations is then solved using backslash.
7 %
8 % From http://www.amath.washington.edu/~rjl/fdmbook/chapter3 (2007)
9
10 clear;
11 close all;
12 clc;
13
14 a = 0;
15 b = 1;
16 c = 0;
17 d = 2;
18 mx = 4;
19 my=8;
20 hx = (b-a)/(mx+1);
21 hy=(d-c)/(my+2);
22 x = linspace(a,b,mx+2); % grid points x including boundaries
23 y = linspace(c,d,my+2); % grid points y including boundaries
24
25
26 [X,Y] = meshgrid(x,y); % 2d arrays of x,y values
27 X = X'; % transpose so that X(i,j),Y(i,j) are
28 Y = Y'; % coordinates of (i,j) point
29
30 Iint = 2:mx+1; % indices of interior points in x
31 Jint = 2:my+1; % indices of interior points in y
32 Xint = X(Iint,Jint); % interior points
33 Yint = Y(Iint,Jint);
34
35 f = @(x,y) 1.25*exp(x+y/2); % f(x,y) function
36
37 rhs = f(Xint,Yint); % evaluate f at interior points for ...
    right hand side
38 % rhs is modified below for boundary ...
    conditions.
39
40 utrue = exp(X+Y/2); % true solution for test problem
41
42 % set boundary conditions around edges of usoln array:
43
44 usoln = utrue; % use true solution for this test problem
45 % This sets full array, but only ...
    boundary values
46 % are used below. For a problem where ...
    utrue
47 % is not known, would have to set each ...
    edge of
48 % usoln to the desired Dirichlet ...

```

```

                                boundary values.
49
50
51 % adjust the rhs to include boundary terms:
52 rhs(:,1) = rhs(:,1) - usoln(Iint,1)/hy^2;
53 rhs(:,my) = rhs(:,my) - usoln(Iint,my+2)/hy^2;
54 rhs(1,:) = rhs(1,:) - usoln(1,Jint)/hx^2;
55 rhs(mx,:) = rhs(mx,:) - usoln(mx+2,Jint)/hx^2;
56
57
58 % convert the 2d grid function rhs into a column vector for rhs of ...
    system:
59 F = reshape(rhs,mx*my,1);
60
61 % form matrix A:
62 Ix = speye(mx);
63 Iy=speye(my);
64 ex = ones(mx,1);
65 ey=ones(my,1);
66 T = spdiags([ex -4*ex ex],[-1 0 1],mx,mx);
67 S = spdiags([ey ey],[-1 1],my,my);
68 A = (kron(Iy,T) + kron(S,Ix)) / hx^2;
69
70
71 % Solve the linear system:
72 uvec = A\F;
73
74 % reshape vector solution uvec as a grid function and
75 % insert this interior solution into usoln for plotting purposes:
76 % (recall boundary conditions in usoln are already set)
77
78 usoln(Iint,Jint) = reshape(uvec,mx,my);
79
80 % assuming true solution is known and stored in utrue:
81 err = max(max(abs(usoln-utrue)));
82 fprintf('Error relative to true solution of PDE = %10.3e \n',err)
83
84 % plot results:
85
86 clf
87 hold on
88
89 %plot grid:
90 plot(X,Y,'g'); plot(X',Y','g')
91
92 % plot solution:
93 contour(X,Y,usoln,30,'k')
94
95 axis([a b c d])
96 daspect([1 1 1])
97 title('Contour plot of computed solution')
98 hold off

```

```

1
2 % poisson2.m -- solve the Poisson problem  $u_{xx} + u_{yy} = f(x,y)$ 
3 % on  $[a,b] \times [a,b]$ .
4 %
5 % The 5-point Laplacian is used at interior grid points.
6 % This system of equations is then solved using backslash.
7 %
8 % From http://www.amath.washington.edu/~rjl/fdmbook/chapter3 (2007)
9
10 clear;
11 close all;
12 clc;
13
14 a = 0;
15 b = 1;
16 c = 0;
17 d = 2;
18 mx = 4;
19 my=8;
20 hx = (b-a)/(mx+1);
21 hy=(d-c)/(my+1);
22 x = linspace(a,b,mx+2); % grid points x including boundaries
23 y = linspace(c,d,my+2); % grid points y including boundaries
24
25
26 [X,Y] = meshgrid(x,y); % 2d arrays of x,y values
27 X = X'; % transpose so that X(i,j),Y(i,j) are
28 Y = Y'; % coordinates of (i,j) point
29
30 Iint = 2:mx+1; % indices of interior points in x
31 Jint = 2:my+1; % indices of interior points in y
32 Xint = X(Iint,Jint); % interior points
33 Yint = Y(Iint,Jint);
34
35 f = @(x,y) 1.25*exp(x+y/2); % f(x,y) function
36
37 rhs = f(Xint,Yint); % evaluate f at interior points for ...
    right hand side
38 % rhs is modified below for boundary ...
    conditions.
39
40 utrue = exp(X+Y/2); % true solution for test problem
41
42 % set boundary conditions around edges of usoln array:
43
44 usoln = utrue; % use true solution for this test problem
45 % This sets full array, but only ...
    boundary values
46 % are used below. For a problem where ...
    utrue
47 % is not known, would have to set each ...
    edge of
48 % usoln to the desired Dirichlet ...

```

```

                                boundary values.
49
50
51 % adjust the rhs to include boundary terms:
52 rhs(:,1) = rhs(:,1) - usoln(Iint,1)/hy^2;
53 rhs(:,my) = rhs(:,my) - usoln(Iint,my+2)/hy^2;
54 rhs(1,:) = rhs(1,:) - usoln(1,Jint)/hx^2;
55 rhs(mx,:) = rhs(mx,:) - usoln(mx+2,Jint)/hx^2;
56
57
58 % convert the 2d grid function rhs into a column vector for rhs of ...
    system:
59 F = reshape(rhs,mx*my,1);
60
61 % form matrix A:
62 Ix = speye(mx);
63 Iy=speye(my);
64 ex = ones(mx,1);
65 ey=ones(my,1);
66 T = spdiags([ex -4*ex ex],[-1 0 1],mx,mx);
67 S = spdiags([ey ey],[-1 1],my,my);
68 A = (kron(Iy,T) + kron(S,Ix)) / (hx*hy);
69
70
71 % Solve the linear system:
72 uvec = A\F;
73
74 % reshape vector solution uvec as a grid function and
75 % insert this interior solution into usoln for plotting purposes:
76 % (recall boundary conditions in usoln are already set)
77
78 usoln(Iint,Jint) = reshape(uvec,mx,my);
79
80 % assuming true solution is known and stored in utrue:
81 err = max(max(abs(usoln-utrue)));
82 fprintf('Error relative to true solution of PDE = %10.3e \n',err)
83
84 % plot results:
85
86 clf
87 hold on
88
89 % plot grid:
90 % plot(X,Y,'g'); plot(X',Y','g')
91
92 % plot solution:
93 contour(X,Y,usoln,30,'k')
94
95 axis([a b c d])
96 daspect([1 1 1])
97 title('Contour plot of computed solution')
98 hold off

```

- 3.2 a. Show that the 9-point Laplacian (3.17) has the truncation error derived in Section 3.5.
- b. Modify the matlab script poisson.m to use the 9-point Laplacian (3.17) instead of the 5-point Laplacian, and to solve the linear system (3.18) where  $f_{ij}$  is given by (3.19). Perform a grid refinement study to verify that fourth order accuracy is achieved.

*Solution.*

We start with  $\nabla_5^2 u(x_i, y_j) = \frac{1}{h^2}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j})$ . Hence,  $\nabla_9^2 u(x_i, y_j) = \frac{1}{6h^2}(4h^2 \nabla_5^2 u(x_i, y_j) + u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} - 4u_{i,j})$ . We also know the local truncation error of the 5-point Laplacian is given by  $\tau_{ij} = \frac{1}{12}h^2(u_{xxxx} + u_{yyyy}) + \mathcal{O}(h^4)$ . Now applying the Taylor series expansion of  $u(x+h, y+h)$  centered at  $u(x_i, y_j)$ , we get

$$\begin{aligned} u(x+h, y+h) &= u(x_i, y_j) + u(x_i, y_j)_x h + u(x_i, y_j)_y h + \frac{1}{2}u(x_i, y_j)_{xx} h^2 + u(x_i, y_j)_{xy} h^2 \\ &\quad + \frac{1}{2}u(x_i, y_j)_{yy} h^2 + \frac{1}{6}u(x_i, y_j)_{xxx} h^3 + \frac{1}{2}u(x_i, y_j)_{xxy} h^3 \\ &\quad + \frac{1}{2}u(x_i, y_j)_{xyy} h^3 + \frac{1}{6}u(x_i, y_j)_{yyy} h^3 + \frac{1}{24}u(x_i, y_j)_{xxxx} h^4 \\ &\quad + \frac{1}{6}u(x_i, y_j)_{xxxy} h^4 + \frac{1}{4}u(x_i, y_j)_{xxyy} h^4 + \frac{1}{6}u(x_i, y_j)_{xyyy} h^4 \\ &\quad + \frac{1}{24}u(x_i, y_j)_{yyyy} h^4 + \mathcal{O}(h^5) \end{aligned}$$

Therefore

$$\begin{aligned} &u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} = \\ &4u(x_i, y_j) + 2h^2 \nabla^2 u(x_i, y_j) + h^4 \left( \frac{1}{6}u(x_i, y_j)_{xxxx} + u(x_i, y_j)_{xxyy} \right. \\ &\quad \left. + \frac{1}{6}u(x_i, y_j)_{yyyy} \right) + \mathcal{O}(h^6) \end{aligned}$$

Thus, we have that

$$\begin{aligned} \nabla_9^2 u(x_i, y_j) &= \\ \frac{1}{6h^2} \left( \frac{1}{3}h^4(u(x_i, y_j)_{xxxx} + u(x_i, y_j)_{yyyy}) + 6h^2 \nabla^2 u(x_i, y_j) + \right. \\ &\quad \left. h^4 \left[ \frac{1}{6}u(x_i, y_j)_{xxxx} + u(x_i, y_j)_{xxyy} + \frac{1}{6}u(x_i, y_j)_{yyyy} \right] + \mathcal{O}(h^6) \right) = \\ &\quad \nabla^2 u(x_i, y_j) + \frac{1}{12}h^2 \nabla^2 (\nabla^2 u(x_i, y_j)) + \mathcal{O}(h^4) = \end{aligned}$$



$$\nabla^2 u(x_i, y_j) + \frac{1}{12} h^2 \nabla^2 f(x_i, y_j) + \mathcal{O}(h^4)$$

Whence, the local truncation error for the 9-point Laplacian is given by  $\tau_{ij} = \nabla_9^2 u(x_i, y_j) - \nabla^2 u(x_i, y_j) = \frac{1}{12} h^2 \nabla^2 f(x_i, y_j) + \mathcal{O}(h^4)$  as derived in section 3.5.

The modified code here is of the 9 point Laplacian. Also, upon a grid refinement, we find that the order of accuracy indeed converges to fourth order.

```

1 clear;
2 close all;
3 clc;
4
5 for i=1:10
6
7     a = 0;
8     b = 1;
9     m = 40*2*i;
10    h = (b-a)/(m+1);
11    x = linspace(a,b,m+2);    % grid points x including boundaries
12    y = linspace(a,b,m+2);    % grid points y including boundaries
13
14
15    [X,Y] = meshgrid(x,y);    % 2d arrays of x,y values
16    X = X';                    % transpose so that X(i,j),Y(i,j) are
17    Y = Y';                    % coordinates of (i,j) point
18
19    Iint = 2:m+1;              % indices of interior points in x
20    Jint = 2:m+1;              % indices of interior points in y
21    Xint = X(Iint,Jint);       % interior points
22    Yint = Y(Iint,Jint);
23
24    f = @(x,y) 1.25*exp(x+y/2);    % f(x,y) function
25
26    rhs = f(Xint,Yint)+(h^2/12)*1.25*f(Xint,Yint);    % evaluate f ...
27    % at interior points for right hand side
28    % rhs is modified below for boundary ...
29    % conditions.
30
31    utrue = exp(X+Y/2);          % true solution for test problem
32
33    % set boundary conditions around edges of usoln array:
34
35    usoln = utrue;               % use true solution for this test problem
36    % This sets full array, but only ...
37    % boundary values
38    % are used below. For a problem where ...
39    % utrue
40    % is not known, would have to set each ...
41    % edge of
42    % usoln to the desired Dirichlet ...
43    % boundary values.
44
45
46    % form matrix A:
47    I = speye(m);
48    e = ones(m,1);
49    T= spdiags([(4/h^2)*e (-10/h^2 - 10/h^2)*e (4/h^2)*e], [-1 0 1], ...
50    m, m);
51    T2=spdiags([(1/h^2)*e (2/h^2 + 2/h^2)*e (1/h^2)*e], [-1 0 1], m, m);
52    K = spdiags([e 4*e e],[-1 0 1],m,m);
53    S = spdiags([e e],[-1 1],m,m);

```

```
47 A = (1/6) * (kron(I,T) + kron(S,T2));
48
49 % adjust the rhs to include boundary terms:
50 rhs(:,1) = rhs(:,1) - K*usoln(Iint,1)/(6*h^2);
51 rhs(:,m) = rhs(:,m) - K*usoln(Iint,m+2)/(6*h^2);
52 rhs(1,:) = rhs(1,:) - (usoln(1,Jint)*K)/(6*h^2);
53 rhs(m,:) = rhs(m,:) - (usoln(m+2,Jint)*K)/(6*h^2);
54 rhs(1,1) = rhs(1,1) - usoln(1, 1)/(6*h^2);
55 rhs(1,m) = rhs(1,m) - usoln(1, m+2)/(6*h^2);
56 rhs(m,1) = rhs(m,1) - usoln(m+2, 1)/(6*h^2);
57 rhs(m,m) = rhs(m,m) - usoln(m+2, m+2)/(6*h^2);
58
59
60 % convert the 2d grid function rhs into a column vector for rhs of ...
    system:
61 F = reshape(rhs,m*m,1);
62
63 % Solve the linear system:
64 uvec = A\F;
65
66 % reshape vector solution uvec as a grid function and
67 % insert this interior solution into usoln for plotting purposes:
68 % (recall boundary conditions in usoln are already set)
69
70 usoln(Iint,Jint) = reshape(uvec,m,m);
71
72 % assuming true solution is known and stored in utrue:
73 err = max(max(abs(usoln-utrue)));
74 fprintf('Error relative to true solution of PDE = %10.3e \n',err)
75
76 % plot results:
77
78 clf
79 hold on
80
81 % plot grid:
82 % plot(X,Y,'g'); plot(X',Y','g')
83
84 % plot solution:
85 contour(X,Y,usoln,30,'k')
86
87 axis([a b a b])
88 daspect([1 1 1])
89 title('Contour plot of computed solution')
90 hold off
91
92 end
93
94 error_loglog(h,err);
```