Mikhail Gaerlan
16 March 2018
MAT 226B Freund

Part 1

```
function V = solvePoisson(F,a,b,c,d)
%Solve two-dimensional Poisson's equation T_m1V + VT_m2 = F
%   input  - F            right-hand side
%            [a,b]        x interval
%            [c,d]        y interval
%   output - V            solution matrix
[m2,m1] = size(F); h1 = (b-a)/(m1+1); h2 = (d-c)/(m2+1);
G = multZ(h2,c,d,multZ(h1,a,b,F.').'); V = zeros(m2,m1);
for k=1:m1; for j=1:m2; V(j,k)=G(j,k)/(l(h2,j,c,d)+l(h1,k,a,b)); end; end
V = multZ(h2,c,d,multZ(h1,a,b,V.').');
end
function lam = l(h,j,a,b); lam = 2*(1-cos(pi*h*j/(b-a))); end
function W = multZ(h,a,b,V)
%Multiply ZV where Z is the eigenvector matrix of T_m1
%   input  - V is an m1 x m2 matrix
%   output - W = ZV
[m1,m2] = size(V);
Vt = [zeros(1,m2);V;zeros(m1+1,m2)];
Wt = fft(Vt); W = -sqrt(2*h/(b-a))*imag(Wt(2:m1+1,:));
end
```

Part 2

The bilinear function can be represented by the following linear system.

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}$$

Thus

$$\begin{aligned} \begin{vmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{vmatrix} &= \begin{vmatrix} x_1 & y_2 & x_1y_2 \\ x_2 & y_1 & x_2y_1 \\ x_2 & y_2 & x_2y_2 \end{vmatrix} - \begin{vmatrix} x_1 & y_1 & x_1y_1 \\ x_2 & y_1 & x_2y_1 \\ x_2 & y_2 & x_2y_2 \end{vmatrix} + \begin{vmatrix} x_1 & y_1 & x_1y_1 \\ x_1 & y_2 & x_1y_2 \\ x_2 & y_2 & x_2y_2 \end{vmatrix} - \begin{vmatrix} x_1 & y_1 & x_1y_1 \\ x_1 & y_2 & x_1y_2 \\ x_2 & y_1 & x_2y_1 \end{vmatrix} \\ &= -x_1^2 - y_1^2 + 2x_1^2y_1y_2 - x_1^2y_2^2 + 2x_1x_2y_1^2 - 4x_1x_2y_1y_2 + \\ &\quad 2x_1x_2y_2^2 - x_2^2y_1^2 + 2x_2^2y_1y_2 - x_2^2y_2^2 \\ &= -(x_1 - x_2)^2 (y_1 - y_2)^2 \\ &\neq 0 \qquad \text{since } x_1 \neq x_2, y_1 \neq y_2 \end{aligned}$$

Since the matrix is nonsingular, the system has a unique solution.

Part 3

```
function [A,A1,A2,I1,I2,Xs,Ys,b] = laplace_matching(m,as,bs,f,v)
%Create Laplacian for two rectangles with matching grids Av=b
%   input  - as       second rectangle x-axis starting point
%            bs       second rectangle y-axis starting point
%            m        grid length 1/2^m
%   output - A        laplacian
%            A1       region 1 laplacian
%            A2       region 2 laplacian
%            I1       matrix that returns region 1 elements
%            I2       matrix that returns region 2 elements
%            Xs       x-values of regions 1 and 2
%            Ys       y-values of regions 1 and 2
%            b        b such that Av = b

%Parameter initialization
h = 1/2^m;
rectangle1_x1=0;rectangle1_x2=1; rectangle1_y1=0; rectangle1_y2=3;
rectangle2_x1=as;rectangle2_x2=as+4;rectangle2_y1=bs; rectangle2_y2=bs+1;
rectangle_x1 = min([rectangle1_x1,rectangle2_x1]);
rectangle_x2 = max([rectangle1_x2,rectangle2_x2]);
rectangle_y1 = min([rectangle1_y1,rectangle2_y1]);
rectangle_y2 = max([rectangle1_y2,rectangle2_y2]);
m1 = (rectangle_x2-rectangle_x1)/h+1;
m2 = (rectangle_y2-rectangle_y1)/h+1;

%Create a large rectangular grid that includes both regions
[X,Y]=meshgrid(...
    rectangle_x1:h:rectangle_x2,...
    rectangle_y1:h:rectangle_y2);

%Create the Laplacian matrix for the large rectangular area
T_m1 = 2*speye(m2)+...
    sparse(2:m2,1:m2-1,-ones(m2-1,1),m2,m2)+...
    sparse(1:m2-1,2:m2,-ones(m2-1,1),m2,m2); A_m = m1*m2;
I_m1 = speye(m2); A_rectangle=spalloc(A_m,A_m,5*A_m-2*m1-2*m2);
A_rectangle(1:m2,1:m2) = T_m1+2*I_m1; A_rectangle(1:m2,m2+1:2*m2) = -I_m1;
for i=2:m1-1
    A_rectangle((i-1)*m2+1:i*m2,(i-1)*m2+1:i*m2) = T_m1+2*I_m1;
    A_rectangle((i-1)*m2+1:i*m2,i*m2+1:(i+1)*m2) = -I_m1;
    A_rectangle((i-1)*m2+1:i*m2,(i-2)*m2+1:(i-1)*m2) = -I_m1;
end
A_rectangle((m1-1)*m2+1:m1*m2,(m1-1)*m2+1:m1*m2) = T_m1+2*I_m1;
A_rectangle((m1-1)*m2+1:m1*m2,(m1-2)*m2+1:(m1-1)*m2) = -I_m1;

%Logicals that specify regions of the rectangular area
is_in_rectangle1 = rectangle1_x1 < X & X < rectangle1_x2 &...
    rectangle1_y1 < Y & Y < rectangle1_y2;
is_in_rectangle2 = rectangle2_x1 < X & X < rectangle2_x2 &...
    rectangle2_y1 < Y & Y < rectangle2_y2;
```

```
is_in_rectangle1_or_2 = is_in_rectangle1 | is_in_rectangle2;
is_in_rectangle1_and_2 = is_in_rectangle1 & is_in_rectangle2;
is_in_rectangle1_reshape = reshape(is_in_rectangle1,[A_m,1]);
is_in_rectangle2_reshape = reshape(is_in_rectangle2,[A_m,1]);
is_in_rectangle12_reshape = reshape(is_in_rectangle1_or_2,[A_m,1]);

%Filter out the regions of interest
A1 = A_rectangle(is_in_rectangle1_reshape,is_in_rectangle1_reshape);
A2 = A_rectangle(is_in_rectangle2_reshape,is_in_rectangle2_reshape);
A = A_rectangle(is_in_rectangle12_reshape,is_in_rectangle12_reshape);

%Create the matrices that get the elements of regions
total_elems = sum(is_in_rectangle12_reshape);
I1_pre = is_in_rectangle1_reshape(is_in_rectangle12_reshape);
I1_elems = sum(I1_pre); otherI1 = I1_pre.*(1:total_elems)';
ind1 = otherI1(otherI1 ~= 0);
I1 = sparse(1:I1_elems,ind1,ones(I1_elems,1),I1_elems,total_elems);
I2_pre = is_in_rectangle2_reshape(is_in_rectangle12_reshape);
I2_elems = sum(I2_pre); otherI2 = I2_pre.*(1:total_elems)';
ind2 = otherI2(otherI2 ~= 0);
I2 = sparse(1:I2_elems,ind2,ones(I2_elems,1),I2_elems,total_elems);

%Create the border logicals
is_next_to_x1 = X==rectangle1_x1+h & rectangle1_y1<Y & Y<rectangle1_y2;
is_next_to_x2 = X==rectangle1_x2-h & rectangle1_y1<Y &...
    Y<rectangle1_y2 & ~is_in_rectangle1_and_2;
is_next_to_y1 = Y==rectangle1_y1+h & rectangle1_x1<X & X<rectangle1_x2;
is_next_to_y2 = Y==rectangle1_y2-h & rectangle1_x1<X & X<rectangle1_x2;
is_next_to_b = Y==rectangle2_y1+h & rectangle2_x1<X &...
    X<rectangle2_x2 & ~is_in_rectangle1_and_2 & is_in_rectangle2;
is_next_to_b_1 = Y==rectangle2_y2-h & rectangle2_x1<X &...
    X<rectangle2_x2 & ~is_in_rectangle1_and_2 & is_in_rectangle2;
is_next_to_a_4 = X==rectangle2_x2-h & rectangle2_y1<Y & Y<rectangle2_y2;

%Create the right hand-side
F = f(X,Y); G = v(X,Y); H = zeros(m2,m1);
for i = 1:m1
    for j = 1:m2
        if is_in_rectangle1_or_2(j,i); H(j,i) = h^2*F(j,i); end
        if is_next_to_x1(j,i); H(j,i) = H(j,i)  + G(j,i-1); end
        if is_next_to_x2(j,i); H(j,i) = H(j,i)  + G(j,i+1); end
        if is_next_to_y1(j,i); H(j,i) = H(j,i)  + G(j-1,i); end
        if is_next_to_y2(j,i); H(j,i) = H(j,i)  + G(j+1,i); end
        if is_next_to_b(j,i); H(j,i) = H(j,i)   + G(j-1,i); end
        if is_next_to_b_1(j,i); H(j,i) = H(j,i) + G(j+1,i); end
        if is_next_to_a_4(j,i); H(j,i) = H(j,i) + G(j,i+1); end
    end
end
b = H(is_in_rectangle1_or_2);
Xs = X(is_in_rectangle1_or_2);Ys = Y(is_in_rectangle1_or_2);
end
```

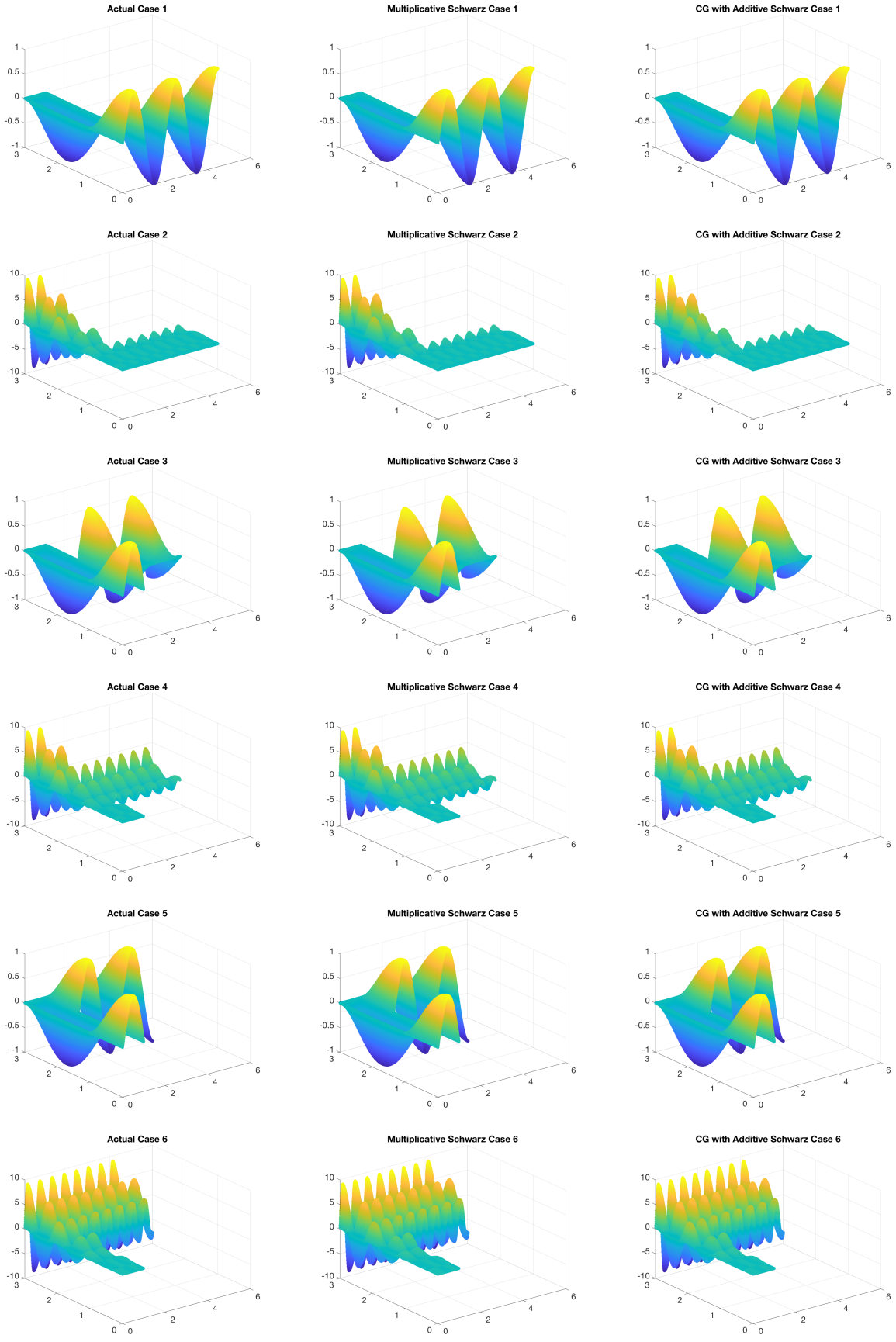
```
function [vapp,k] = mult_Schwarz(m,nmax,as,bs,tol,A,b,I1,I2)
%Multiplicative Schwarz Method
%   input  - m        grid length 1/2^m
%            nmax     max iterations
%            as       second rectangle x-axis starting point
%            bs       second rectangle y-axis starting point
%            tol      convergence tolerance
%            A        Laplacian
%            b        right-hand side vector
%            I1       matrix that returns region 1 elements
%            I2       matrix that returns region 2 elements
%   output - vapp     approximate solution
%            k        number of iterations reached
rectangle1_x1=0;rectangle1_x2=1; rectangle1_y1=0; rectangle1_y2=3;
rectangle2_x1=as;rectangle2_x2=as+4;rectangle2_y1=bs; rectangle2_y2=bs+1;
rectangle1_m1 = 1*2^m+1; rectangle1_m2 = 3*2^m+1;
rectangle2_m1 = 4*2^m+1; rectangle2_m2 = 1*2^m+1;

vapp = zeros(length(b),1);
error = b - A*vapp;
for n = 1:nmax
    vapp = vapp + multB1(b-A*vapp);
    vapp = vapp + multB2(b-A*vapp);
    error = error-Minv(A*error); k = n; if norm(error) < tol; break; end
end

    function b = multB1(v)
        m1 = rectangle1_m1-2; m2 = rectangle1_m2-2;
        z = reshape(I1*v,[m2,m1]);
        g = solvePoisson(z,rectangle1_x1,rectangle1_x2,...
            rectangle1_y1,rectangle1_y2);
        b = I1'*reshape(g,[m1*m2,1]);
    end
    function b = multB2(v)
        m1 = rectangle2_m1-2; m2 = rectangle2_m2-2;
        z = reshape(I2*v,[m2,m1]);
        g = solvePoisson(z,rectangle2_x1,rectangle2_x2,...
            rectangle2_y1,rectangle2_y2);
        b = I2'*reshape(g,[m1*m2,1]);
    end
    function b = Minv(v); b = multB1(v)+multB2(v)-multB2(A*multB1(v)); end
end
```

```
function [vapp,k] = CG_add_Schwarz(m,nmax,as,bs,tol,A,b,I1,I2)
%Multiplicative Schwarz Method
%   input  - m         grid length 1/2^m
%            nmax      max iterations
%            as        second rectangle x-axis starting point
%            bs        second rectangle y-axis starting point
%            tol       convergence tolerance
%            A         Laplacian
%            b         right-hand side vector
%            I1        matrix that returns region 1 elements
%            I2        matrix that returns region 2 elements
%   output - vapp      approximate solution
%            k         number of iterations reached
rectangle1_x1=0;rectangle1_x2=1; rectangle1_y1=0; rectangle1_y2=3;
rectangle2_x1=as;rectangle2_x2=as+4;rectangle2_y1=bs; rectangle2_y2=bs+1;
rectangle1_m1 = 1*2^m+1; rectangle1_m2 = 3*2^m+1;
rectangle2_m1 = 4*2^m+1; rectangle2_m2 = 1*2^m+1;

[vapp,~,~,k] = pcg(A,b,tol,nmax,@Minv);

function b = multB1(v)
m1 = rectangle1_m1-2; m2 = rectangle1_m2-2;
z = reshape(I1*v,[m2,m1]);
g = solvePoisson(z,rectangle1_x1,rectangle1_x2,...
    rectangle1_y1,rectangle1_y2);
b = I1'*reshape(g,[m1*m2,1]);
end

function b = multB2(v)
m1 = rectangle2_m1-2; m2 = rectangle2_m2-2;
z = reshape(I2*v,[m2,m1]);
g = solvePoisson(z,rectangle2_x1,rectangle2_x2,...
    rectangle2_y1,rectangle2_y2);
b = I2'*reshape(g,[m1*m2,1]);
end

function b = Minv(v); b = multB1(v)+multB2(v); end
end
```

```
clearvars; clc; m = 8; nmax = 100; tol = 1e-6;
tic; params = [1/2,0,0,1,1/2;1/2,0,2,7/2,2;1/4,1,0,1,1/2;1/4,1,2,7/2,2;...
    1/2,2,0,1,1/2;1/2,2,2,7/2,2]; n = size(params,1);
relerrs1 = []; relerrs2 = []; nsolves1 = []; nsolves2 = [];
for i = 1:n
    as=params(i,1);bs=params(i,2);alpha=params(i,3);beta=params(i,4);
    gamma=params(i,5);
    [A,~,~,I1,I2,X,Y,b] = laplace_matching(m,as,bs,...
        @(x,y) f(x,y,alpha,beta,gamma),@(x,y) v(x,y,alpha,beta,gamma));
    V = v(X,Y,alpha,beta,gamma);
    [Va1,iters1] = mult_Schwarz(m,nmax,as,bs,tol,A,b,I1,I2);
    [Va2,iters2] = CG_add_Schwarz(m,nmax,as,bs,tol,A,b,I1,I2);
    err1=max(abs(Va1-V))/max(abs(V));err2=max(abs(Va2-V))/max(abs(V));
    relerrs1 = [relerrs1;err1]; relerrs2 = [relerrs2;err2];
    nsolves1 =[nsolves1;iters1]; nsolves2 =[nsolves2;iters2];
    scatter3(X,Y,V,'o','fill','CData',V);set(gca,'FontSize',16);
    title(strcat("Actual Case ",int2str(i)));
    saveas(gcf,strcat("../Figures/final_3_actual_",int2str(i),".png"));
    scatter3(X,Y,Va1,'o','fill','CData',Va1);set(gca,'FontSize',16);
    title(strcat("Multiplicative Schwarz Case ",int2str(i)));
    saveas(gcf,strcat("../Figures/final_3_mult_",int2str(i),".png"));
    scatter3(X,Y,Va2,'o','fill','CData',Va2);set(gca,'FontSize',16);
    title(strcat("CG with Additive Schwarz Case ",int2str(i)));
    saveas(gcf,strcat("../Figures/final_3_pcg_",int2str(i),".png"));
end
matrix2latex(relerrs1,'../Tables/final_3_relerr_mult.tex','alignment','r',
    'format','%-.15e');
matrix2latex(relerrs2,'../Tables/final_3_relerr_pcg.tex','alignment','r','
    format','%-.15e');
matrix2latex(nsolves1,'../Tables/final_3_iters_mult.tex','alignment','r','
    format','%-4d');
matrix2latex(nsolves2,'../Tables/final_3_iters_pcg.tex','alignment','r','
    format','%-4d');toc;
function true = v(x,y,a,b,g); true=y.^a.*sin(b.*pi.*x).*cos(g.*pi.*y); end
function fun = f(x,y,a,b,g)
fun = b.^2.*pi.^2.*y.^a.*sin(b.*pi.*x).*cos(g.*pi.*y)...
    -a.*(a-1).*y.^(a-2).*sin(b.*pi.*x).*cos(g.*pi.*y)...
    +2.*a.*g.*pi.*y.^(a-1).*sin(b.*pi.*x).*sin(g.*pi.*y)...
    +g.^2.*pi.^2.*y.^a.*sin(b.*pi.*x).*cos(g.*pi.*y);
end
```

Case	Multiplicative Schwarz		CG with Additive Schwarz	
	Relative Maximal Error	# R_1 Solves	Relative Maximal Error	# R_1 Solves
(i)	1.068736599729725e-05	1	1.068932784664689e-05	8
(ii)	1.107670592472620e-04	10	1.107699548938841e-04	7
(iii)	1.071413154762890e-05	1	1.101750926146838e-05	8
(iv)	1.106840400874727e-04	11	1.105455987666748e-04	8
(v)	1.068736599729725e-05	1	1.110856789987569e-05	8
(vi)	1.024290146721919e-04	11	1.024271635730477e-04	8

If the function was smoother, then the multiplicative Schwarz method was more efficient; otherwise, `pcg` was more efficient. The run time was approximately 230 seconds for all 12 runs.


Actual Case 1
Multiplicative Schwarz Case 1
CG with Additive Schwarz Case 1
Actual Case 2
Multiplicative Schwarz Case 2
CG with Additive Schwarz Case 2
Actual Case 3
Multiplicative Schwarz Case 3
CG with Additive Schwarz Case 3
Actual Case 4
Multiplicative Schwarz Case 4
CG with Additive Schwarz Case 4
Actual Case 5
Multiplicative Schwarz Case 5
CG with Additive Schwarz Case 5
Actual Case 6
Multiplicative Schwarz Case 6
CG with Additive Schwarz Case 6

Part 4

```
function [vapp,X,Y,k] = alt_Schwarz(m,nmax,as,bs,tol,f,v)
%Multiplicative Schwarz Method
%   input  - m        grid length 1/2^m
%            nmax     max iterations
%            as       second rectangle x-axis starting point
%            bs       second rectangle y-axis starting point
%            tol      convergence tolerance
%            A        Laplacian
%            b        right-hand side vector
%            I1       matrix that returns region 1 elements
%            I2       matrix that returns region 2 elements
%   output - vapp     approximate solution
%            k        number of iterations reached
rectangle1_x1=0;rectangle1_x2=1; rectangle1_y1=0; rectangle1_y2=3;
rectangle2_x1=as;rectangle2_x2=as+4;rectangle2_y1=bs; rectangle2_y2=bs+1;
rectangle1_m1 = 1*2^m+1; rectangle1_m2 = 3*2^m+1;
rectangle2_m1 = 4*2^m+1; rectangle2_m2 = 1*2^m+1;
A1_m = rectangle1_m1*rectangle1_m2; A2_m = rectangle2_m1*rectangle2_m2;

h = 1/2^m;
[X1,Y1]=meshgrid(...
    rectangle1_x1:h:rectangle1_x2,...
    rectangle1_y1:h:rectangle1_y2);
[X2,Y2]=meshgrid(...
    rectangle2_x1:h:rectangle2_x2,...
    rectangle2_y1:h:rectangle2_y2);
X1_reshape = reshape(X1,[A1_m,1]); Y1_reshape = reshape(Y1,[A1_m,1]);
X2_reshape = reshape(X2,[A2_m,1]); Y2_reshape = reshape(Y2,[A2_m,1]);

is_in_rectangle1 = rectangle1_x1 < X1 & X1 < rectangle1_x2 &...
    rectangle1_y1 < Y1 & Y1 < rectangle1_y2;
is_in_rectangle2 = rectangle2_x1 < X2 & X2 < rectangle2_x2 &...
    rectangle2_y1 < Y2 & Y2 < rectangle2_y2;
is_gamma1 = (X1 == rectangle1_x2 & Y1 > rectangle2_y1 & Y1 < rectangle2_y2
   );
is_gamma2 = (X2 == rectangle2_x1 & Y2 < rectangle2_y2 & Y2 > rectangle2_y1
   );
is_gamma3 = (Y2 == rectangle2_y1 & X2 < rectangle1_x2 & X2 > rectangle2_x1
   );
is_gamma4 = (Y2 == rectangle2_y2 & X2 < rectangle1_x2 & X2 > rectangle2_x1
   );
is_next_to_gamma1 = (X1 == rectangle1_x2-h & Y1 > rectangle2_y1 & Y1 <
   rectangle2_y2);
is_next_to_gamma2 = (X2 == rectangle2_x1+h & Y2 < rectangle2_y2 & Y2 >
   rectangle2_y1);
is_next_to_gamma3 = (Y2 == rectangle2_y1+h & X2 < rectangle1_x2 & X2 >
   rectangle2_x1);
is_next_to_gamma4 = (Y2 == rectangle2_y2-h & X2 < rectangle1_x2 & X2 >
   rectangle2_x1);
```

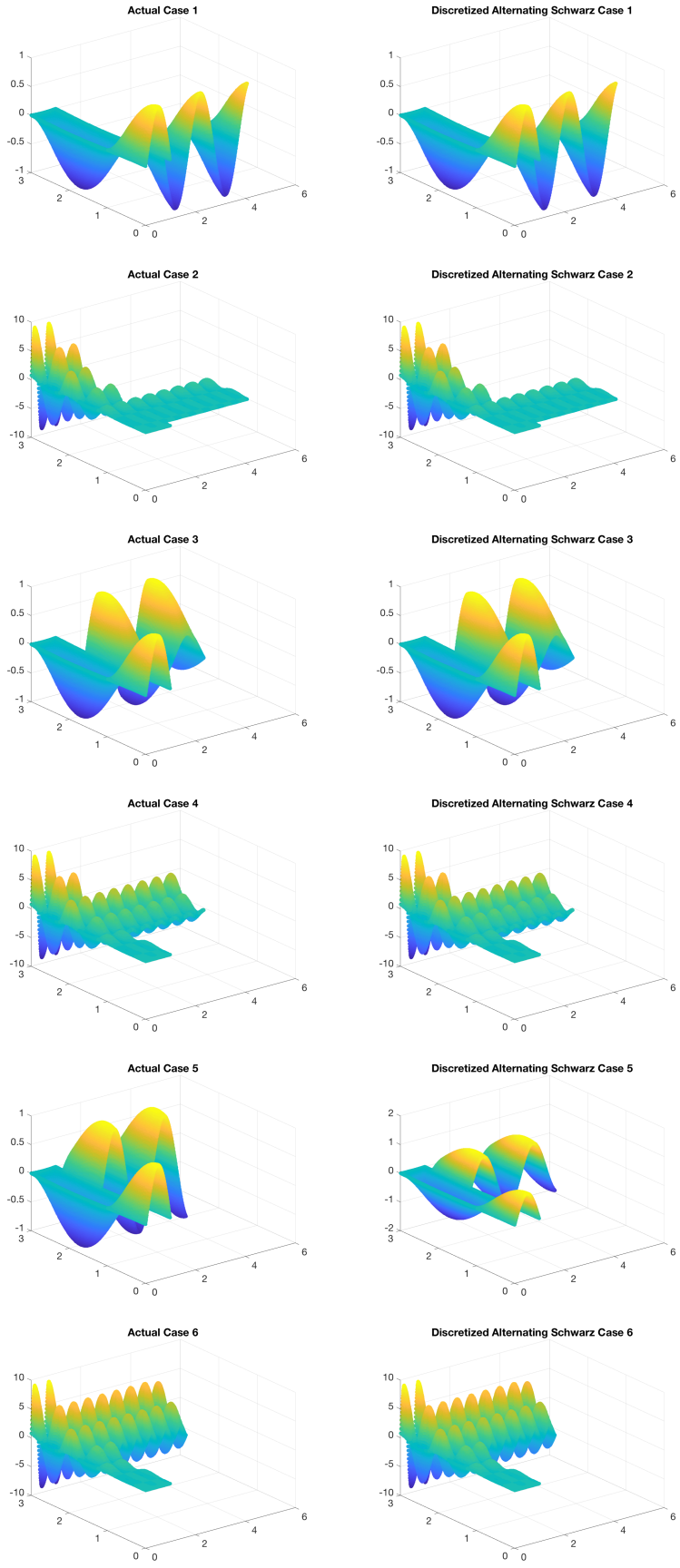
```
42
43 is_rect1_x1 = (X1 == rectangle1_x1 & Y1 > rectangle1_y1 & Y1 <
       rectangle1_y2);
44 is_rect1_x2 = (X1 == rectangle1_x2 & Y1 > rectangle1_y1 & Y1 <
       rectangle1_y2 &...
45     (Y1 > rectangle2_y2 | Y1 < rectangle2_y1));
46 is_rect1_y1 = (Y1 == rectangle1_y1 & X1 > rectangle1_x1 & X1 <
       rectangle1_x2);
47 is_rect1_y2 = (Y1 == rectangle1_y2 & X1 > rectangle1_x1 & X1 <
       rectangle1_x2);
48 is_rect2_x2 = (X2 == rectangle2_x2 & Y2 > rectangle2_y1 & Y2 <
       rectangle2_y2);
49 is_rect2_y1 = (Y2 == rectangle2_y1 & X2 > rectangle2_x1 &...
50     X2 > rectangle1_x2 & Y2 < rectangle2_x2);
51 is_rect2_y2 = (Y2 == rectangle2_y2 & X2 > rectangle2_x1 &...
52     X2 > rectangle1_x2 & Y2 < rectangle2_x2);
53
54 is_next_to_rect1_x1 = (X1 == rectangle1_x1+h & Y1 > rectangle1_y1 & Y1 <
       rectangle1_y2);
55 is_next_to_rect1_x2 = (X1 == rectangle1_x2-h & Y1 > rectangle1_y1 & Y1 <
       rectangle1_y2 &...
56     (Y1 > rectangle2_y2 | Y1 < rectangle2_y1));
57 is_next_to_rect1_y1 = (Y1 == rectangle1_y1+h & X1 > rectangle1_x1 & X1 <
       rectangle1_x2);
58 is_next_to_rect1_y2 = (Y1 == rectangle1_y2-h & X1 > rectangle1_x1 & X1 <
       rectangle1_x2);
59 is_next_to_rect2_x2 = (X2 == rectangle2_x2-h & Y2 > rectangle2_y1 & Y2 <
       rectangle2_y2);
60 is_next_to_rect2_y1 = (Y2 == rectangle2_y1+h & X2 > rectangle2_x1 &...
61     X2 > rectangle1_x2 & Y2 < rectangle2_x2);
62 is_next_to_rect2_y2 = (Y2 == rectangle2_y2-h & X2 > rectangle2_x1 &...
63     X2 > rectangle1_x2 & Y2 < rectangle2_x2);
64 to_ignore2 = (X2 < rectangle2_x1+1/5);
65 to_accept1 = is_in_rectangle1;
66 to_accept2 = is_in_rectangle2 & ~to_ignore2;
67
68 ind_rect1_x1 = get_indices(is_rect1_x1);
69 ind_rect1_x2 = get_indices(is_rect1_x2);
70 ind_rect1_y1 = get_indices(is_rect1_y1);
71 ind_rect1_y2 = get_indices(is_rect1_y2);
72 ind_rect2_x2 = get_indices(is_rect2_x2);
73 ind_rect2_y1 = get_indices(is_rect2_y1);
74 ind_rect2_y2 = get_indices(is_rect2_y2);
75
76 ind_next_to_rect1_x1 = get_indices(is_next_to_rect1_x1);
77 ind_next_to_rect1_x2 = get_indices(is_next_to_rect1_x2);
78 ind_next_to_rect1_y1 = get_indices(is_next_to_rect1_y1);
79 ind_next_to_rect1_y2 = get_indices(is_next_to_rect1_y2);
80 ind_next_to_rect2_x2 = get_indices(is_next_to_rect2_x2);
81 ind_next_to_rect2_y1 = get_indices(is_next_to_rect2_y1);
82 ind_next_to_rect2_y2 = get_indices(is_next_to_rect2_y2);
83
84 ind_in_rectangle1 = get_indices(is_in_rectangle1);
85 ind_gamma1 = get_indices(is_gamma1);
```

```
ind_next_to_gamma1 = get_indices(is_next_to_gamma1);
ind_in_rectangle2 = get_indices(is_in_rectangle2);
ind_gamma2 = get_indices(is_gamma2);
ind_gamma3 = get_indices(is_gamma3);
ind_gamma4 = get_indices(is_gamma4);
ind_next_to_gamma2 = get_indices(is_next_to_gamma2);
ind_next_to_gamma3 = get_indices(is_next_to_gamma3);
ind_next_to_gamma4 = get_indices(is_next_to_gamma4);
ind_to_accept1 = get_indices(to_accept1);
ind_to_accept2 = get_indices(to_accept2);

v1 = reshape(v(X1,Y1),[A1_m,1]);
v1(ind_in_rectangle1) = zeros(length(ind_in_rectangle1),1);
v1(ind_gamma1) = zeros(length(ind_gamma1),1);
v2 = reshape(v(X2,Y2),[A2_m,1]);
v2(ind_in_rectangle2) = zeros(length(ind_in_rectangle2),1);
v2(ind_gamma2) = zeros(length(ind_gamma2),1);
v2(ind_gamma3) = zeros(length(ind_gamma3),1);
v2(ind_gamma4) = zeros(length(ind_gamma4),1);

b1 = reshape(h^2*f(X1,Y1),[A1_m,1]);
b1(ind_next_to_rect1_x1) = b1(ind_next_to_rect1_x1) + v1(ind_rect1_x1);
b1(ind_next_to_rect1_x2) = b1(ind_next_to_rect1_x2) + v1(ind_rect1_x2);
b1(ind_next_to_rect1_y1) = b1(ind_next_to_rect1_y1) + v1(ind_rect1_y1);
b1(ind_next_to_rect1_y2) = b1(ind_next_to_rect1_y2) + v1(ind_rect1_y2);
b2 = reshape(h^2*f(X2,Y2),[A2_m,1]);
b2(ind_next_to_rect2_x2) = b2(ind_next_to_rect2_x2) + v2(ind_rect2_x2);
b2(ind_next_to_rect2_y1) = b2(ind_next_to_rect2_y1) + v2(ind_rect2_y1);
b2(ind_next_to_rect2_y2) = b2(ind_next_to_rect2_y2) + v2(ind_rect2_y2);

for n = 1:nmax
    v1(ind_in_rectangle1) = A1inv(b1 + bgamma1Igamma1region2(v2));
    v2(ind_in_rectangle2) = A2inv(b2 + bgamma2Igamma2region1(v1));
    k = n;
end
vapp = [v1(ind_to_accept1);v2(ind_to_accept2)];
X = [X1(ind_to_accept1);X2(ind_to_accept2)];
Y = [Y1(ind_to_accept1);Y2(ind_to_accept2)];
    function b1 = bgamma1Igamma1region2(v2)
        b1 = zeros(A1_m,1); b1p = zeros(A1_m,1);
        for i = ind_gamma1'
            [x,y,z] = get_four_points(v2,i,X1_reshape,Y1_reshape,X2,Y2,h);
            b1p(i) = interpolate(X1_reshape(i),Y1_reshape(i),x,y,z);
        end; b1(ind_next_to_gamma1) = b1p(ind_gamma1);
    end
    function b2 = bgamma2Igamma2region1(v1)
        b2 = zeros(A2_m,1); b2p = zeros(A2_m,1);
        for i = ind_gamma2'
            [x,y,z] = get_four_points(v1,i,X2_reshape,Y2_reshape,X1,Y1,h);
            b2p(i) = interpolate(X2_reshape(i),Y2_reshape(i),x,y,z);
        end; b2(ind_next_to_gamma2) = b2p(ind_gamma2);
        for i = ind_gamma3'
            [x,y,z] = get_four_points(v1,i,X2_reshape,Y2_reshape,X1,Y1,h);
            b2p(i) = interpolate(X2_reshape(i),Y2_reshape(i),x,y,z);
```

```
        end; b2(ind_next_to_gamma3) = b2p(ind_gamma3);
        for i = ind_gamma4'
            [x,y,z] = get_four_points(v1,i,X2_reshape,Y2_reshape,X1,Y1,h);
            b2p(i) = interpolate(X2_reshape(i),Y2_reshape(i),x,y,z);
        end; b2(ind_next_to_gamma4) = b2p(ind_gamma4);
    end
    function b = A1inv(v)
        m1 = rectangle1_m1-2; m2 = rectangle1_m2-2;
        z = reshape(v(ind_in_rectangle1),[m2,m1]);
        g = solvePoisson(z,rectangle1_x1,rectangle1_x2,...
            rectangle1_y1,rectangle1_y2);
        b = reshape(g,[m1*m2,1]);
    end
    function b = A2inv(v)
        m1 = rectangle2_m1-2; m2 = rectangle2_m2-2;
        z = reshape(v(ind_in_rectangle2),[m2,m1]);
        g = solvePoisson(z,rectangle2_x1,rectangle2_x2,...
            rectangle2_y1,rectangle2_y2);
        b = reshape(g,[m1*m2,1]);
    end
    function [x,y,z] = get_four_points(v2,ind,X1,Y1,X2,Y2,h)
        is_near_point = X2 > X1(ind)-h & X2 < X1(ind)+h &...
            Y2 > Y1(ind)-h & Y2 < Y1(ind)+h;
        ind_near_point = get_indices(is_near_point);
        x = X2(ind_near_point); y = Y2(ind_near_point); z = v2(
           ind_near_point);
    end
    function u = interpolate(x0,y0,x,y,z)
        interpmatrix = [1,x(1),y(1),x(1)*y(1);...
            1,x(2),y(2),x(2)*y(2);...
            1,x(3),y(3),x(3)*y(3);...
            1,x(4),y(4),x(4)*y(4)];
        c = interpmatrix\z;
        u = c(1)+c(2)*x0+c(3)*y0+c(4)*x0*y0;
    end
    function indices = get_indices(logical_region)
        I = I_region(logical_region);
        nn = size(I,2);
        pre_ind = I'*I*(1:nn)';
        indices = pre_ind(pre_ind~=0);
    end
    function I = I_region(logical_region)
        [m1,m2] = size(logical_region);
        A_m = m1*m2;
        logical_reshape = reshape(logical_region,[A_m,1]);
        num_elems = sum(logical_reshape);
        ind1 = logical_reshape.*(1:A_m)';
        ind = ind1(ind1~=0);
        I = sparse(1:num_elems,ind,ones(num_elems,1),num_elems,A_m,
           num_elems);
    end
end
```

```
clearvars; clc; m = 7; nmax = 30; tol = 1e-6;
tic; params = [2/5,1/5,0,1,1/2;2/5,1/5,2,7/2,2;...
    1/5,6/5,0,1,1/2;1/5,6/5,2,7/2,2;...
    2/5,9/5,0,1,1/2;2/5,9/5,2,7/2,2]; n = size(params,1);
relerrs1 = []; relerrs2 = [0]; nsolves1 = []; nsolves2 = [0];
for i = 1:n
    as=params(i,1);bs=params(i,2);alpha=params(i,3);beta=params(i,4);
    gamma=params(i,5);
    [vapp,X,Y,iters1] = alt_Schwarz(m,nmax,as,bs,tol,@(x,y)f(x,y,alpha,
       beta,gamma),@(x,y)v(x,y,alpha,beta,gamma));
    vact = v(X,Y,alpha,beta,gamma);
    err1=max(abs(vact-vapp))/max(abs(vact));
    relerrs1 = [relerrs1;err1]; nsolves1 =[nsolves1;iters1];
    scatter3(X,Y,vact,'o','fill','CData',vact);set(gca,'FontSize',16);
    title(strcat("Actual Case ",int2str(i)));
    saveas(gcf,strcat("../Figures/final_4_actual_",int2str(i),".png"));
    scatter3(X,Y,vapp,'o','fill','CData',vapp);set(gca,'FontSize',16);
    title(strcat("Discretized Alternating Schwarz Case ",int2str(i)));
    saveas(gcf,strcat("../Figures/final_4_add_",int2str(i),".png"));
end
matrix2latex(relerrs1,'../Tables/final_4_relerr_add.tex','alignment','r','
   format','%-.15e');
matrix2latex(relerrs2,'../Tables/final_4_relerr_gmres.tex','alignment','r'
   ,'format','%-.15e');
matrix2latex(nsolves1,'../Tables/final_4_iters_add.tex','alignment','r','
   format','%-4d');
matrix2latex(nsolves2,'../Tables/final_4_iters_gmres.tex','alignment','r',
   'format','%-4d'); toc;
function true = v(x,y,a,b,g); true=y.^a.*sin(b.*pi.*x).*cos(g.*pi.*y); end
function fun = f(x,y,a,b,g)
fun = b.^2.*pi.^2.*y.^a.*sin(b.*pi.*x).*cos(g.*pi.*y)...
    -a.*(a-1).*y.^(a-2).*sin(b.*pi.*x).*cos(g.*pi.*y)...
    +2.*a.*g.*pi.*y.^(a-1).*sin(b.*pi.*x).*sin(g.*pi.*y)...
    +g.^2.*pi.^2.*y.^a.*sin(b.*pi.*x).*cos(g.*pi.*y);
end
```

Case	Discretized Alternating Schwarz		GMRES with Preconditioner	
	Relative Maximal Error	# R_1 Solves	Relative Maximal Error	# R_1 Solves
(i)	8.826018913059459e-04	30	0.000000000000000e+00	0
(ii)	2.204336957196286e-03	30		
(iii)	8.859326481153902e-04	30		
(iv)	2.101333790306490e-02	30		
(v)	6.033999160146575e-04	30		
(vi)	1.636715246570604e-02	30		


Actual Case 1
Discretized Alternating Schwarz Case 1
Actual Case 2
Discretized Alternating Schwarz Case 2
Actual Case 3
Discretized Alternating Schwarz Case 3
Actual Case 4
Discretized Alternating Schwarz Case 4
Actual Case 5
Discretized Alternating Schwarz Case 5
Actual Case 6
Discretized Alternating Schwarz Case 6