## Part 1

The analytic solution to the problem, for and , is

## Part 2

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual 10x10 nodes.png |  |

Figure : Numeric solution for steady state, error and Residuals during ADI iterations for 10 by 10 cells.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual 10x10 nodes.png |  |

Figure : Numeric solution for steady state, error and Residuals during ADI iterations for 20 by 20 cells.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual 10x10 nodes.png |  |

Figure : Numeric solution for steady state, error and Residuals during ADI iterations for 40 by 40 cells.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual 10x10 nodes.png |  |

Figure : Numeric solution for steady state, error and Residuals during ADI iterations for 80 by 80 cells.

## Part 3

The numeric solution was considered converged when the L2 residual of the steady state equation with the current numeric solution was below a given tolerance.

### A)

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Explicit time marching numeric solution, error, and residuals at each time step for 10 by 10 cells.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Explicit time marching numeric solution, error, and residuals at each time step for 20 by 20 cells.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Explicit time marching numeric solution, error, and residuals at each time step for 40 by 40 cells.

### B)

Only partial convergence of the solution by the ADI solver at each timestep was implemented. The ADI routine stopped when the residual of the current solution was an order of magnitude less than the initial residual.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Implicit time marching numeric solution, error, and residuals at each time step for time step equal to 2 times the maximum allowable time step for explicit time marching solution.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Implicit time marching numeric solution, error, and residuals at each time step for time step equal to 5 times the maximum allowable time step for explicit time marching solution.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Implicit time marching numeric solution, error, and residuals at each time step for time step equal to 10 times the maximum allowable time step for explicit time marching solution.

|  |  |
| --- | --- |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phin exp 10x10 nodes.png | E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Phie exp 10x10 nodes.png |
| E:\Users\Michael\Documents\MATLAB\Coursework\ME 811\HW 2\Residual exp 10x10 nodes.png |  |

Figure : Implicit time marching numeric solution, error, and residuals at each time step for time step equal to 50 times the maximum allowable time step for explicit time marching solution.

## Conclusions

For the direction calculation method for the steady state solution (Part 2), the errors between the numeric and analytic solution are largely concentrated at the corner nodes along x = 1, due to the discontinuity of the boundary conditions at these locations. Increasing the grid resolution does not significantly impact the magnitude of the errors are the corner node, though it does limit the extent of the errors into the interior nodes. The tolerance for the residual was held constant for all grid resolutions and solution methods, so the lack of change in the error magnitudes is not terribly surprising. From the residual plots, it is seen that doubling the number of cells in each direction (and hence quadrupling the total number of cells), results in a fourfold increase in the number of iterations required by the ADI solver to reach convergence.

As with the direct solution method, for the time marching methods, increasing the resolution of the grid reduces the extent of the error in the interior nodes, though it does little to change the magnitude of the error at the corner nodes. For the explicit time marching method, quadrupling the number of cells results in quadrupling of the number of required time iterations to reach convergence. Note that for this scheme, doubling the grid resolution (and thus halving the dx and dy terms) results in a maximum allowable timestep that is one fourth of the original timestep. For the implicit method, increasing the temporal step size had no effect on the numeric solution or error. Increasing the temporal step size resulted in a proportional decrease in the number of time iterations required to reach convergence.

In terms of computational time, the explicit time marching scheme was the fastest to converge as no matrix inversions were required; all computations were made using matrix multiplication and addition. Even though it required far more iterations than the direction solution, each iteration was accomplished much more quickly. However, this result may be specific only to Matlab implementations, as Matlab performs matrix operations efficiently whereas for or while loops (which are used much more heavily by the direct and implicit time marching methods) are implemented quite slowly. The implicit time marching scheme was by far the slowest to converge, as it required two loops (time marching iterations and ADI iterations). All methods were able to produce identical accuracy in the numeric solution. Once the matrix representing the linear set of equations and the source term are generated, implementation of the three solution methods is similar, with the direction solution being the simplest and the implicit method being the most difficult. It is likely that one would only implement the implicit time marching method if the other solution methods were failing to converge.

## Appendix

function [phin Residual] = H2P2(N,M)

%Solves given steady state equation using FVM coupled with ADI.

%Completed by Michael Crawley for ME 811 HW#2 Problem#2

%Generate mesh

x = 1/(2\*N):1/N:1-1/(2\*N);

dx = mean(diff(x));

y = 1/(2\*M):1/M:1-1/(2\*M);

dy = mean(diff(y));

n = 1:100; %number of terms for analytic solution

[xx yy nn] = meshgrid(x,y,n);

AR = dx/dy; iAR = dy/dx;

%calculate analytic solution

phia = (2/pi)\*sum((((-1).^(nn+1)+1)./nn).\*sin(nn\*pi.\*yy).\*sinh(nn\*pi.\*xx)./sinh(nn\*pi),3);

%Calculate Source terms (including sources from boundary conditions)

gamma = ones(M,N,4); %last dimension is for the four faces; east, west, north, and south (in that order)

S = zeros(M,N)\*dx\*dy;

S(:,end) = S(:,end)-2\*gamma(:,end,1)\*iAR; %apply boundary conditions at x = 1

gamma = reshape(gamma,M\*N,4);

%Generate matrix for linear system of equations

X0 = -((gamma(:,1)+gamma(:,2))\*iAR+(gamma(:,3)+gamma(:,4))\*AR);

X0(1:M) = X0(1:M)-gamma(1:M,2)\*iAR; %apply boundary condition at x = 0

X0(end-M+1:end) = X0(end-M+1:end)-gamma(end-M+1:end,1)\*iAR; %apply boundary condition at x = 1

X0(1:M:end) = X0(1:M:end)-gamma(1:M:end,4)\*AR; %apply boundary condition at y = 0

X0(M:M:end) = X0(M:M:end)-gamma(M:M:end,3)\*AR; %apply boundary condition at y = 1

X1 = gamma(:,3)\*AR;

X1(M:M:end) = 0; %apply boundary condition at y = 1

Xn1 = gamma(:,4)\*AR;

Xn1(1:M:end) = 0; %apply boundary condition at y = 0

XM = gamma(:,1)\*iAR;

XM(end-M+1:end) = 0; %apply boundary condition at x = 1

XnM = gamma(:,2)\*iAR;

XnM(1:M) = 0; %apply boundary condition at x = 0;

X1 = circshift(X1,1);

XM = circshift(XM,M);

Xn1 = circshift(Xn1,-1);

XnM = circshift(XnM,-M);

X = spdiags([XnM Xn1 X0 X1 XM],[-M -1 0 1 M],N\*M,N\*M);

%Calculate numeric solution

iResidual = norm(reshape(S,M\*N,1)); %calculate initial residual

[phin Residual] = ADI2d(X,S,'-TDMA',iResidual/(10E6));

%Plot results

figure; pcolor(x,y,phin); shading interp; colorbar; colormap(flipud(gray)); title(['Numeric Solution for ', num2str(N),'x',num2str(M),' nodes']);saveas(gcf,['Phin ', num2str(N),'x',num2str(M),' nodes'],'fig');

figure; pcolor(x,y,phia-phin); shading interp; colorbar; colormap gray; title(['Numeric Error for ', num2str(N),'x',num2str(M),' nodes']);saveas(gcf,['Phie ', num2str(N),'x',num2str(M),' nodes'],'fig');

figure; semilogy(Residual);title(['Residual for ', num2str(N),'x',num2str(M),' nodes']);xlabel('Iteration');ylabel('Residual');saveas(gcf,['Residual ', num2str(N),'x',num2str(M),' nodes'],'fig');

end

function [phin Residual] = H2P3a(N,M)

%Solves given equation using FVM by explicit time marching.

%Completed by Michael Crawley for ME 811 HW#2 Problem#3a

%Set convergence tolerance

Rtol = 1E-6;

itrmax = 1E5;

%Generate mesh

x = 1/(2\*N):1/N:1-1/(2\*N);

dx = mean(diff(x));

y = 1/(2\*M):1/M:1-1/(2\*M);

dy = mean(diff(y));

n = 1:100; %number of terms for analytic solution

[xx yy nn] = meshgrid(x,y,n);

AR = dx/dy; iAR = dy/dx;

%calculate analytic solution

phia = (2/pi)\*sum((((-1).^(nn+1)+1)./nn).\*sin(nn\*pi.\*yy).\*sinh(nn\*pi.\*xx)./sinh(nn\*pi),3);

%Calculate Source terms (including sources from boundary conditions)

rhoc = ones(N\*M,1);

gamma = ones(M,N,4); %last dimension is for the four faces; east, west, north, and south (in that order)

S = zeros(M,N)\*dx\*dy;

S(:,end) = S(:,end)-2\*gamma(:,end,1)\*iAR; %apply boundary conditions at x = 1

S = reshape(S,N\*M,1);

gamma = reshape(gamma,M\*N,4);

%Calculate timestep

dt = mean(rhoc)/4/(1/dx^2+1/dy^2)/mean(mean(gamma));

%Generate matrix for linear system of equations

X0 = -((gamma(:,1)+gamma(:,2))\*iAR+(gamma(:,3)+gamma(:,4))\*AR);

X0(1:M) = X0(1:M)-gamma(1:M,2)\*iAR; %apply boundary condition at x = 0

X0(end-M+1:end) = X0(end-M+1:end)-gamma(end-M+1:end,1)\*iAR; %apply boundary condition at x = 1

X0(1:M:end) = X0(1:M:end)-gamma(1:M:end,4)\*AR; %apply boundary condition at y = 0

X0(M:M:end) = X0(M:M:end)-gamma(M:M:end,3)\*AR; %apply boundary condition at y = 1

X1 = gamma(:,3)\*AR;

X1(M:M:end) = 0; %apply boundary condition at y = 1

Xn1 = gamma(:,4)\*AR;

Xn1(1:M:end) = 0; %apply boundary condition at y = 0

XM = gamma(:,1)\*iAR;

XM(end-M+1:end) = 0; %apply boundary condition at x = 1

XnM = gamma(:,2)\*iAR;

XnM(1:M) = 0; %apply boundary condition at x = 0;

X1 = circshift(X1,1);

XM = circshift(XM,M);

Xn1 = circshift(Xn1,-1);

XnM = circshift(XnM,-M);

X = spdiags([XnM Xn1 X0 X1 XM],[-M -1 0 1 M],N\*M,N\*M);

%Time march (explicit) until convergence is reached

phin = zeros(N\*M,1);

Residual = zeros(1,itrmax);

Residual(1) = norm(X\*phin-S);

counter = 0;

while Residual(counter+1)>Rtol && counter <= itrmax

phin = phin+dt./(rhoc\*dx\*dy).\*(X\*phin-S);

counter = counter +1;

Residual(counter+1) = norm(X\*phin-S);

end

Residual = Residual(1:counter+1);

phin = reshape(phin,M,N);

%plot results

figure; pcolor(x,y,phin); shading interp; colorbar; colormap(flipud(gray)); title(['Numeric Solution for ', num2str(N),'x',num2str(M),' nodes']);saveas(gcf,['Phin exp ', num2str(N),'x',num2str(M),' nodes'],'fig');

figure; pcolor(x,y,phia-phin); shading interp; colorbar; colormap gray; title(['Numeric Error for ', num2str(N),'x',num2str(M),' nodes']);saveas(gcf,['Phie exp ', num2str(N),'x',num2str(M),' nodes'],'fig');

figure; semilogy(Residual);title(['Residual for ', num2str(N),'x',num2str(M),' nodes']);xlabel('Iteration');ylabel('Residual');saveas(gcf,['Residual exp ', num2str(N),'x',num2str(M),' nodes'],'fig');

end

function [phin Residual] = H2P3b(N,M,dtr)

%Solves given equation using FVM by implicit time marching.

%Completed by Michael Crawley for ME 811 HW#2 Problem#3b

%Set convergence tolerance

Rtol = 1E-6;

itrmax = 1E5;

%Generate mesh

x = 1/(2\*N):1/N:1-1/(2\*N);

dx = mean(diff(x));

y = 1/(2\*M):1/M:1-1/(2\*M);

dy = mean(diff(y));

n = 1:100; %number of terms for analytic solution

[xx yy nn] = meshgrid(x,y,n);

AR = dx/dy; iAR = dy/dx;

%calculate analytic solution

phia = (2/pi)\*sum((((-1).^(nn+1)+1)./nn).\*sin(nn\*pi.\*yy).\*sinh(nn\*pi.\*xx)./sinh(nn\*pi),3);

%Calculate Source terms (including sources from boundary conditions)

rhoc = 1;

gamma = ones(M,N,4); %last dimension is for the four faces; east, west, north, and south (in that order)

S = zeros(M,N)\*dx\*dy;

S(:,end) = S(:,end)-2\*gamma(:,end,1)\*iAR; %apply boundary conditions at x = 1

S = reshape(S,N\*M,1);

gamma = reshape(gamma,M\*N,4);

%Calculate timestep

dt = dtr\*mean(rhoc)/2/(1/dx^2+1/dy^2)/mean(mean(gamma));

%Generate matrix for linear system of equations

X0 = -((gamma(:,1)+gamma(:,2))\*iAR+(gamma(:,3)+gamma(:,4))\*AR);

X0(1:M) = X0(1:M)-gamma(1:M,2)\*iAR; %apply boundary condition at x = 0

X0(end-M+1:end) = X0(end-M+1:end)-gamma(end-M+1:end,1)\*iAR; %apply boundary condition at x = 1

X0(1:M:end) = X0(1:M:end)-gamma(1:M:end,4)\*AR; %apply boundary condition at y = 0

X0(M:M:end) = X0(M:M:end)-gamma(M:M:end,3)\*AR; %apply boundary condition at y = 1

X1 = gamma(:,3)\*AR;

X1(M:M:end) = 0; %apply boundary condition at y = 1

Xn1 = gamma(:,4)\*AR;

Xn1(1:M:end) = 0; %apply boundary condition at y = 0

XM = gamma(:,1)\*iAR;

XM(end-M+1:end) = 0; %apply boundary condition at x = 1

XnM = gamma(:,2)\*iAR;

XnM(1:M) = 0; %apply boundary condition at x = 0;

X1 = circshift(X1,1);

XM = circshift(XM,M);

Xn1 = circshift(Xn1,-1);

XnM = circshift(XnM,-M);

X = spdiags([XnM Xn1 X0 X1 XM],[-M -1 0 1 M],N\*M,N\*M);

%Time march (implicit) until convergence is reached

phin = zeros(N\*M,1);

Residual = zeros(1,itrmax);

Residual(1) = norm(X\*phin-S);

counter = 0;

I = spdiags(ones(N\*M,1),0,N\*M,N\*M);

while Residual(counter+1)>Rtol && counter <= itrmax

phin = ADI2d(I-dt/(rhoc\*dx\*dy)\*X,reshape(phin-dt/(rhoc\*dx\*dy)\*S,M,N),'-TDMA',Residual(counter+1)/10);

phin = reshape(phin,M\*N,1);

counter = counter +1;

Residual(counter+1) = norm(X\*phin-S);

end

Residual = Residual(1:counter+1);

phin = reshape(phin,M,N);

%plot results

figure; pcolor(x,y,phin); shading interp; colorbar; colormap(flipud(gray)); title(['Numeric Solution for ', num2str(N),'x',num2str(M),'x',num2str(dtr),' nodes']);saveas(gcf,['Phin imp ', num2str(N),'x',num2str(M),'x',num2str(dtr),' nodes'],'fig');

figure; pcolor(x,y,phia-phin); shading interp; colorbar; colormap gray; title(['Numeric Error for ', num2str(N),'x',num2str(M),'x',num2str(dtr),' nodes']);saveas(gcf,['Phie imp ', num2str(N),'x',num2str(M),'x',num2str(dtr),' nodes'],'fig');

figure; semilogy(Residual);title(['Residual for ', num2str(N),'x',num2str(M),'x',num2str(dtr),' nodes']);xlabel('Iteration');ylabel('Residual');saveas(gcf,['Residual imp ', num2str(N),'x',num2str(M),'x',num2str(dtr),' nodes'],'fig');

end

function [phi,R] = ADI2d(X,S,method,Rtol,itrmax)

%Performs the Alternating Direction Implicit solver for a 2 dimensional

%system.

%Note: conversion from 2-D grid to 1-D array must be done as i,j -> k = M(i-1)+j,

%where there are M nodes in the y direction and N nodes in the x

%direction

%Code Version: 1.0 @ 2011-04-10

%Inputs:

% X: matrix for the x-derivative terms (M\*N,M\*N)

% Y: matrix for the y-derivative terms (M\*N,M\*N)

% S: source term, either matrix or scalar (M,N)

% Rtol: Residual tolerance (optional; default 1E-5)

% itrmax: maximum number of iterations (optional default 1E4)

%Outpus:

% phi: Solution to the problem

% R: L2norm of the Residuals at each iteration

if ~exist('Rtol','var')

Rtol = 1E-5;

end

if ~exist('itrmax','var')

itrmax = 1E4;

end

[M N] = size(S);

phi = zeros(M\*N,1);

S = reshape(S,M\*N,1);

R = zeros(1,itrmax);

R(1) = norm(X\*phi-S);

counter = 0;

if strcmpi(method,'-TDMA')

while (R(counter+1) >= Rtol) && (counter < itrmax)

for j = 1:M %Row sweep

k = j:M:j+M\*(N-1); %determine nodal array points

Yt = X; Yt(k,k) = 0;

t = Yt\*phi;

b = S(k)-t(k);

A = X(k,k);

phi(k) = TDMsolver(A,b);

end

for i = 1:N %Column sweep

k = M\*(i-1)+1:M\*i;

Yt = X; Yt(k,k) = 0;

t = Yt\*phi;

b = S(k)-t(k);

A = X(k,k);

phi(k) = TDMsolver(A,b);

end

counter = counter + 1;

R(counter+1) = norm(X\*phi-S);

end

else

while (R(counter+1) >= Rtol) && (counter < itrmax)

for j = 1:M %Row sweep

k = j:M:j+M\*(N-1); %determine nodal array points

Yt = X; Yt(k,k) = 0;

t = Yt\*phi;

b = S(k)-t(k);

A = X(k,k);

phi(k) = GaussSeidel(A,b,Rtol);

end

for i = 1:N %Column sweep

k = M\*(i-1)+1:M\*i;

Yt = X; Yt(k,k) = 0;

t = Yt\*phi;

b = S(k)-t(k);

A = X(k,k);

phi(k) = GaussSeidel(A,b,Rtol);

end

counter = counter + 1;

R(counter+1) = norm(X\*phi-S);

end

end

R = R(1:counter+1);

phi = reshape(phi,M,N);

end

function [x] = TDMsolver(A,b)

% Solves tridiagonal matrix using Thomas' algorithm.

%Inputs:

% A: Full matrix, where Ax=b

% b: Column vector

%Outputs:

% x: Solution to the linear equation

n = length(b);

x = zeros(n,1);

aL = [0; diag(A,-1)];

a = diag(A);

aR = [diag(A,1); 0];

aR(1) = aR(1)/a(1);

b(1) = b(1)/a(1);

for i=2:n

t = 1/(a(i)-aR(i-1)\*aL(i));

aR(i) = aR(i)\*t;

b(i) = (b(i)-b(i-1)\*aL(i))\*t;

end

x(n) = b(n);

for i = n-1:-1:1

x(i) = b(i)-aR(i)\*x(i+1);

end

end