Adv. Topics in ME - FDM

EGME 4950

 Homework 3

 Submitted by

Matt Earl

Submitted to

Dr. Zhaohui Qin

Submitted on

April 22, 2021

**Problem Statement:**

Solve the 1-D filter flow problem.

Water flows through a filter, as shown in the figure below.

Shape

Description automatically generated with medium confidence

Because of the filtration material inside the filter, water experiences a friction force which has to be compensated by a pressure gradient:

|  |  |  |
| --- | --- | --- |
|  |  |  |

This is the momentum equation we have to solve for velocity.

Another equation needed is the continuity equation, which is simply

|  |  |  |
| --- | --- | --- |
|  |  |  |

Suppose the water pressure at the flow inlet is 20 Pa and at the outlet is 0 Pa; the of the term in the momentum equation is a function of : .

I want you to write a code to find the water velocity in the filter and the pressure distribution across the filter.

One thing you should avoid is using any calculus in the code. Of course you can use calculus to find the exact solution and compare it with your numerical solution.

Attach your code and results as figures.

**Exact Solution:**

The continuity equation is integrated quite easily:

Where is just a constant. The momentum equation is then:

Integrating over the region [0,1] we solve for the constant velocity distribution:

The pressure distribution is thus:

The distributions are shown in Figure 1.

Chart, line chart

Description automatically generated

***Figure 1:*** *Exact Solution*

**Numerical Solution:**

Step 1. Set up a staggered mesh:

Set up a pressure mesh with N+2 points.

Set up a velocity mesh with N+1 points.

We will associate with every velocity point a value of calculated from the velocity mesh x position.

Step 2: Guess and

We call the preliminary guess of velocity is constantly 1.

We call the preliminary guess of pressure is linear.

Step 3: Use momentum equation to determine the next value of use a forward difference and Picard linearization to deal with the nonlinearity.

NOTE: In my code and in this analysis, I will call this freshly calculated by the symbol

Step 4: Use the continuity to solve for the pressure corrections, is we use backward

difference:

Step 5: Determine the pressure corrections by assuming that is proportional to . This is the very heart of the SIMPLE method. We assume that there is a associated with every

Let us define the variable . There is a associated with each . By substituting in , all internal nodes are governed by the equation:

At the boundaries we have and .

We can solve for these values of by using TDMA.

Step 6: Use the pressure corrections to calculate the velocity corrections we need the

values of and to calculate a

Step 7: Use the pressure corrections and velocity corrections to find the next and

Step 8: Implement under relaxation. We call the relaxation factor for pressure and the

relaxation factor for velocity . Take the last guess and and the new values calculated in step 7 and .

Step 9: Iterate until convergence. I defined my residual by looking at successive values of

velocity.

Here is a diagram showing the way I deal with the staggered mesh in Figure 2.

A piece of paper with writing

Description automatically generated with low confidence

***Figure 2:*** *Solving procedure.*

Using a mesh size of , relaxation factor , and a threshold of results converge in 26 iterations. With the results plotted below in Figure 3.

Chart, line chart

Description automatically generated

***Figure 3:*** *Numerical Solutions Plotted against Ideal Solution.*

Code attached in the appendix.

**Appendix**

%%% FDM HW3 %%%

clear; clc;

%%% Set up Mixed Meshes

N = 8;

dx = 1/(N+1);

x = [dx/2:dx:1-dx/2];

xP = [0:dx:1];

%%% Boundary Conditions and Initial Guesses

Pin = 20;

Pout = 0;

c = 50 - 30\*x;

% Guesses

P = linspace(Pin,Pout,N+2);

Pstar = P;

Pprime = zeros(1,N+2);

u = ones(1,N+1);

ustar = u;

uprime = zeros(1,N+1);

alphaP = 0.8;

alphau = 1-alphaP;

res = 1;

iterations = 0;

ulast = u;

Plast = P;

while res > 0.5e-8

%%% Picard Linearization and forward difference in momentum equation.

Plast = P;

ulast = u;

for i = 1:N+1

u(i) = (Pstar(i)-Pstar(i+1))/(c(i)\*dx\*ustar(i));

end

%%% This new value of u solves momentum but not continuity

ustar = u;

%%% Solve for pressure Corrections using continuity

A = ones(1,N+2);

B = A;

C = B;

D = C;

for i = 1:N+1

d(i) = 1/(c(i)\*dx\*ustar(i));

end

for i = 2:N+1

A(i) = -d(i-1);

B(i) = (d(i)+d(i-1));

C(i) = -d(i);

D(i) = (ustar(i-1)-ustar(i));

end

A(1) = 0;

A(end) = 0;

C(1) = 0;

C(end) = 0;

D(1) = 0;

D(end) = 0;

Pprime = TDMA(A,B,C,D);

%%% Correcting Pressure

P = Pstar + Pprime;

%%% Solve for corresponding velocity corrections

for i = 1:N+1

uprime(i)= d(i)\*(Pprime(i)-Pprime(i+1));

end

u = ustar + uprime;

%%% Start the cycle over again.

res = norm(u - ulast);

Pstar = P + alphaP\*(Plast-P);%(P+Plast)/2 ;%+ (1-omega)\*(P-Pstar);

ustar = u + alphau\*(ulast-u);%(u+ulast)/2 ;%+ (omega)\*(u - ustar);

iterations = iterations+1;

end

Q = sqrt(4/7);

plot(xP,-Q^2\*(50\*xP - (30/2)\*xP.^2) + Pin,'-\*',xP,P,x,Q\*ones(1,N+1),'-\*',x,u)

xlabel('Distance x')

ylabel('Pressure, Velocity')