MATLAB-2011

Simulace 1D proudění krve v elastické cévě

1D proudění krve v elastické cévě lze popsat následujícím systémem parciálních diferenciálních rovnic

$$\begin{aligned} \frac{\partial A}{\partial t} + \frac{\partial Au}{\partial x} &= 0, \\ \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\varrho}\frac{\partial p}{\partial x} &= 0, \end{aligned}$$

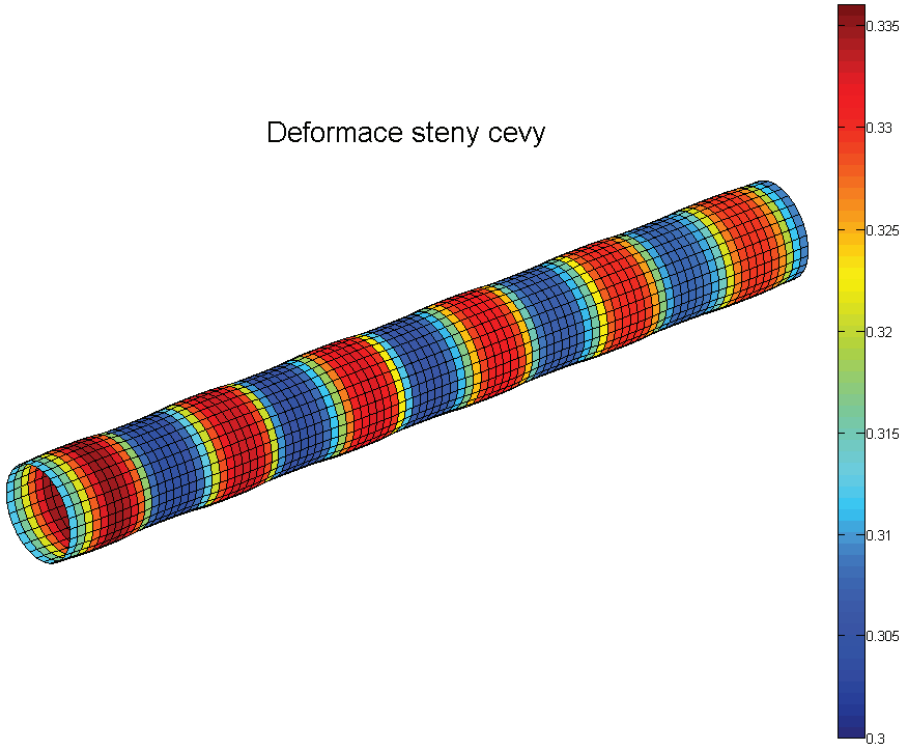
kde $A = A(x,t)$ je plocha průřezu cévy v bodě x, $u = u(x,t)$ je rychlost, $p = p(x,t)$ je tlak a ϱ je hustota.

Systém rovnic je doplněn konstitutivním vztahem mezi tlakem a plochou

$$p = p_{ext} + \beta(\sqrt{A} - \sqrt{A_0}),$$

kde p_{ext} je externí tlak působící na cévu, β je materiálový parametr cévy, A_0 je nezdeformovaná plocha průřezu cévy.

Uvedený systém rovnic byl numericky vyřešen pomocí metody konečných objemů v systému MATLAB, viz zdrojový kód.



Obrázek 1: Simulace proudeni krve pro parametry, $L = 10$ (délka trubice), $A_0 = 1$, $\beta = 0.2$, $\varrho = 0.5$, $p_{ext} = 1$, a vstupní rychlost $u_{in} = 0.1 + 0.05\sin(2t)$

```
%=========================================================================
% Tento program simuluje 1D proudeni krve v elasticke ceve.
% Na vstupu je predepsana pulsujici okrajova podminka pro rychlost.
% Plzen, 14.9.2011
%=========================================================================

function proudeni_1D
global A0 b ro pext
L = 10;     % delka trubice
A0 = 1;     % pocatecni plocha
ui = 0.1;   % pocatecni rychlost
b = 0.2;    % tuhost cevy
ro = 0.5;   % hustota krve
pext = 1;   % externi tlak

iter = 1000;    % pocet iteraci
CFL = 0.3;      % CFL cislo
n = 100;        % pocet kontrolnich objemu
h = L/n;        % velikost prostoroveho kroku
x = linspace(0,L,n+1);
I = 1:n;
xs = (x(I) + x(I+1))/2;

% pocatecni podminky
U = ones(n,2);  % alokace potrebnych matic
U(I,1) = A0*U(I,1);
U(I,2) = ui*U(I,2);
Un = zeros(n,2);

% tisk
figure('color','w');
m = 30;
fi = linspace(0,2*pi,m);
X = zeros(n,m);
Y = zeros(n,m);
Z = zeros(n,m);
C = zeros(n,m);
for i = 1:n
    for j = 1:m
        X(i,j) = xs(i);
    end
end
for i = 1:n
    Y(i,:) = sqrt(U(i,1))*cos(fi);
    Z(i,:) = sqrt(U(i,1))*sin(fi);
end
trub = surface(X,Y,Z,C);
title('Deformace steny cevy v zavislosti na case.')
axis equal;
R = 1.1*sqrt(A0/pi);
axis([0 L -R R -R R]);
view(3);
axis off;
% shading interp;
colorbar;
% konec tisku
```

```
% vlastni vypocet
t = 0;
for op = 1:iter
    % vypocet casoveho kroku z CFL podminky stability
    I = 1:n;
    lam = U(I,2) + (U(I,1)).^(1/4)*sqrt(b/(2*ro));
    dt = min(CFL*h./lam);

    % linearni rekonstrukce
    [UL,UR] = rekonstruuj(U);

    % vypocet toku na hranici kontrolniho objemu
    Psi = zeros(n+1,2);
    for i = 1:n-1
        Psi(i+1,:) = tok_LFF(UL(i,:),UR(i+1,:));
    end

    % okrajove podminky
    % vstup
    Ain = A0;
    uin = ui + 0.05*sin(2*t);
    Psi(1,:) = tok_LFF([Ain,uin],UR(1,:));

    % vystup
    Aout = A0;
    uout = ui;
    Psi(n+1,:) = tok_LFF(UL(n,:),[Aout,uout]);

    % vypocet nasledujici casove hladiny pomoci explicitni Eulerovy metody
    Un(I,:) = U(I,:) - dt/h*(Psi(I+1,:)-Psi(I,:));

    % prechod do nasledujici casove hladiny
    U = Un;
    t = t + dt;

    % tisk
    if(mod(op,5) == 0)
        for i = 1:n
            r = sqrt(U(i,1)/pi);
            Y(i,:) = r*cos(fi);
            Z(i,:) = r*sin(fi);
        end
        for j = 1:m
            C(:,j) = U(:,1)/pi;
        end
        C(n,1) = 0.3;
        C(n,2) = 0.336;
        set(trub,'Ydata',Y,'Zdata',Z,'Cdata',C);
        drawnow;
    end
end


%_____________________________________________________________________________
```

```
function F = tok_LFF(UL,UR) % numericky tok na stene kontrolniho objemu
global A0 b ro pext
W1 = UL(2) + 4*(UL(1))^(1/4)*sqrt(b/(2*ro));
W2 = UR(2) - 4*(UR(1))^(1/4)*sqrt(b/(2*ro));

U = [((W1-W2)/4)^4*(ro/(2*b))^2, (W1+W2)/2];
p = pext + b*(sqrt(U(1))-sqrt(A0));

F = [U(1)*U(2), U(2)^2/2 + p/ro];


%_____________________________________________________________________________
function [UL,UR] = rekonstruuj(U) % linearni rekonstrukce
n = length(U(:,1));
UL = zeros(n,2);
UR = zeros(n,2);
Su = zeros(n,2);
Sd = zeros(n,2);

I = 2:n;
Su(I,:) = U(I,:) - U(I-1,:); % zpetna diference

I = 1:n-1;
Sd(I,:) = U(I+1,:) - U(I,:); % dopredna diference

S = minmod(Su,Sd);

I = 1:n;
UL(I,:) = U(I,:) + S/2;
UR(I,:) = U(I,:) - S/2;


%_____________________________________________________________________________
function C = minmod(A,B) % minmod limiter
n = length(A(:,1));
C = zeros(n,2);
log1 = zeros(n,2);
log2 = zeros(n,2);
I = 1:n;
K = 1:2;
log1(abs(A(I,K)) < abs(B(I,K)) & A(I,K).*B(I,K) > 0) = 1;
log2(abs(A(I,K)) > abs(B(I,K)) & A(I,K).*B(I,K) > 0) = 1;
C(log1 == 1) = A(log1 == 1);
C(log2 == 1) = B(log2 == 1);
```