**Malgorzata Peszynska, Oregon State University Mathematics**

***Notes on implementation of a solver for Stefan problem in semismooth Newton framework using complementarity condition framework***

Date: July 7, 2018

---

## 1. Model

We provide below the notes on implementation of a numerical scheme for Stefan problem

$$\frac{\partial}{\partial t}(u) - D\nabla^2 v = f \tag{1a}$$

$$u \in \beta(v) \tag{1b}$$

In this system $u$ is the enthalpy, $v$ is the temperature. The multivalued graph $\beta$ describes the relationship $u = v + LH(v)$ equivalently

$$u \in \beta(v) = \begin{cases} v, & v \leq 0 \\ [0, L], & v = 0 \\ v + L, & v + L. \end{cases} \tag{2}$$

The relationship (2) can be also written as, with $m(r) := \max(0, \min(r, L))$ (this is one of semismooth functions used for Mixed Complementarity constraint). (See [GMPS paper] or [Ulbrich]. )

$$u - m(u) = v, \tag{3}$$

with

$$m(r) := \begin{cases} 0, & r \leq 0 \\ r, & 0 \leq r \leq L \\ L, & r > L. \end{cases} \tag{4}$$

The system (1) requires initial condition on $u$ and boundary conditions on $v$.

## 2. Discretization

We discretize $u, v$ independently using conservative FD. (Integrated in space and time, with uniform spatial grid parameter $h$ and time step $n$.) In residual form we have

$$R_j := h^2(u_j^n - u_j^{n-1}) + \tau D(2v_j^n - v_{j-1}^n - v_{j+1}^n) - h^2\tau f_j^n \;=\; 0, \tag{5a}$$

$$R_j^\phi := u_j^n - m(u_j^n) - v_j^n \;=\; 0. \tag{5b}$$

It remains to specify boundary conditions (in $v$) and initial condition (in $u$), or previous time step value.

## 3. Solver

At each time step $n$ we solve simultaneously for $u^n$ and $v^n$ using Newton's method. (within the framework of Semismooth Newton methods Ulbrich]).

The residuals $R_j$ in (5a), (5b) must be evaluated at each $j$, and we must compute the jacobian i.e. the derivatives $\frac{dR_j}{du_j}$, $\frac{dR_j}{du_j \pm 1}$, which go to $JAC$. Next we calculate the block matrix $JACV$ which collects $\frac{dR_j}{dv_j}$, $\frac{dR_j}{dv_j \pm 1}$ etc.

For the second part of residual $R^\phi$ the derivatives depend on the cases in (4), and have to be coded as such. Either way these are block diagonal matrices. $\frac{dR_j^\phi}{du_j}$ which go to $PHIJAC$ and $\frac{dR_j^\phi}{dv_j}$ which go to $PHIJACV$.

Finally we collect these. We use $RES = [R; R_\phi]^T$ and

$$A = \begin{bmatrix} JAC & JACV \\ PHIJAC & PHIJACV \end{bmatrix} \tag{6}$$

In each Newton step we solve $A\Delta R = -RES$.

## 4. Code

```
function [x,v]=semi_nonlinear_Stefan2phase_forJulia (M,Tend,dt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% solves a 1D nonlinear diffusion problem (Stefan two phase problem)
%% run as
%% semi_nonlinear_Stefan2phase_forJulia (50,1,0.001)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% M. Peszynska for Julia Kowalski, 7/2018
%%  Copyright Department of Mathematics, Oregon State University
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = 0; b = 1;
%Tend = 0.13;

%%% the control parameters below control the individual terms
Storage = 1;
Diffusion = 1e1;
Latent = 10;
%%%
function w = acc_2fun(x,u,v)
    w = u ;% ones(size(u));
end
function w = acc_2dufun(x,u,v)
    w = 1+0*u;
end
function w = acc_2dvfun(x,u,v)
    w = 0*size(v);
end
```

```
%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function u = initial (x)
    u =-1+0*x;
end
function v = boundary(x0,t)
    v = -1; if x0>0,v=1;end;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w = rhs_fun(x,u,t)
  w = 0*u;
  w(find(x>0.7 & x<0.8))=10;
end

function v = rhs_dfun(x,u,t)
    v = 0*u;
end

%%%%%%%%%%%%%%%%%%%%%%%%
dx = (b-a)/(M-1);
x = a:dx:b; x = x';

%%%% initial condition; also: initial guess for stationary problems
uold = initial (x);
vold = 0*uold;

clf;
t = 0;
totiter = 0;
maxit = 21;
tol = 1e-12;
ntime = 0;

nsteps = ceil(Tend/dt);
erru = zeros(nsteps,1); errv = erru; errvstat=erru;

%%%% time loop
while t < Tend

    t = t + dt;
    ntime = ntime + 1;

    it = 0;
    %% uold is previous time step
```

```
%% we compute u (new time step) using Newton's iteration
oldacc = dx*dx*acc_2fun(x,uold,vold);

%% choose as initial Newton guess the value from previous iteration
u = uold; v = vold;
resnorm = inf;
while it < maxit && resnorm >= tol

    %% compute properties using current values of u
    acc = dx*dx  * acc_2fun(x,u,v);
    accdu = dx*dx* acc_2dufun(x,u,v);

    rhs = dx*dx*dt * rhs_fun(x,u,t);
    rhsd = dx*dx*dt * rhs_dfun(x,u,t);

    %% accumulation terms
    res = acc - oldacc - rhs;

    %% diffusion terms
    for j = 2:size(x,1) -1
        res(j) = res(j) + Diffusion*dt*(2*v(j)-v(j-1)-v(j+1));
    end
    %% bcond in residual
    j = 1;         res (j,1) = dx*dx*(v(j)-boundary(a,t));
    j = size(x,1); res (j,1) = dx*dx*(v(j)-boundary(b,t));

    %%%% compute jacobians: jac=dres/du, jacv=dres/dv
    jac = sparse(size(x,1),size(x,1));
    jacv = sparse(size(x,1),size(x,1));
    for j = 2:size(x,1) -1
        %%% accumulation terms
        jac(j,j) = accdu(j);

        %%%% diffusion terms:
        jacv(j,j) = jacv(j,j) + 2*Diffusion*dt;
        jacv(j,j-1) = jacv(j,j-1) - Diffusion*dt;
        jacv(j,j+1) = jacv(j,j+1) - Diffusion*dt;

        %%%% contribution to diagonal terms from source terms
        jac(j,j) = jac(j,j) - rhsd(j);
    end
    %% bcond in jacobian
    j = 1;jacv(j,j) = dx*dx;
    j = size(x,1);jacv(j,j) = dx*dx;
```

```
    %%%% constraint equation
    resphi = 0*res;
    for j=1:length(resphi)
        resphi(j) = u(j)-v(j) - max(0,min(u(j),Latent));
    end

    phijacu = sparse(size(x,1),size(x,1));
    phijacv = sparse(size(x,1),size(x,1));
    for j=1:length(u)
        phijacv(j,j)=-1;
    end

    for j=1:length(u)
        if u(j) <=Latent && u(j) >=0
           phijacu(j,j)=0;
        else
           phijacu(j,j)=1;
        end
    end
    %% jacobian for the bcond
    phijacu(1,1) = 1;phijacu(M,M) = 1;
    phijacv(1,1) = -1;phijacv(M,M) = -1;

    %%
    jacall = [jac,jacv;phijacu,phijacv];
    resall = [res;resphi];


    %%%% solve linear system
    %%%%%%%% test size of residual: if small, quit
    resnorm = norm(resall,inf);
    %fprintf('iter=%d res norm=%g\n',it,resnorm);%pause
    if it >1  %% force code to make at least one linear solve
        if resnorm < tol, break; end
    end
    it = it + 1;

    corr = jacall \ resall;
    ucorr = corr(1:length(u));
    vcorr = corr(length(u)+1:end);

    unew = u - ucorr; vnew = v - vcorr;
    u = unew; v = vnew;

end
```

```
    %% Newton converged or broke ...
    if it == maxit && resnorm >= tol
        fprintf('Time step =%d time=%g. STOP: Newton did not converge in %d iters\n',...
            ntime,t,maxit);
        break;
    end
    totiter = totiter + it;


    if 1
      fprintf(...
        'Time step=%d time=%g Newton finished. Iters=%d (total=%d, aver=%g). Final res n
            ntime,t,it,totiter,totiter/ntime,resnorm);
        plot(x,unew,x,vnew);
%        axis([a,b,-0.5,2]);
        title(sprintf('Time t=%g',t));
        legend('enthalpy','temperature');
        pause(0.05);
    end


    uold = u;
    vold = v;
end
```

## 5. Example

Th example hard-coded in the template starts from $u(x,0) = -1$, and uses $v(0,t) = -1$, and $v(1,t) = 1$. It also has a piecewise constant source whcih can be turned off if you wish.

The solution shows both $u$ and $v$.

If you focus on $v$, you will see the characteristic discontinuity of derivative at the interface when $v = 0$.

## 6. References

There are many proper references of course for Stefan problem. I include here only those directly needed.


GMPS N. Gibson, P. Medina, M. Peszynska, R. Showalter, Evolution of phase transitions in methane hydrate, J. Math. Anal. Appl. Volume 409, Issue 2 (2014), pp 816-833, doi=10.1016/j.jmaa.2013.07.023. http://math.oregonstate.edu/ mpesz/documents/publications/GMPS13.pdf

Ulbrich Michael Ulbrich. Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces, volume 11 of MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011.