MTH 452-552 Assignment 1 Solution notes. **M. Peszynska**
Prof. M. Peszynska Due: 01/12/2018

Problem 1. Given a fixed parameter $\tau > 0$, consider the (rough) ODE

$$u' = f(t,u), \quad t \in [0,T], \quad u(0) = 0, \tag{1}$$

with $f(t,u) = 1$, for $j\tau \le t < (j+\frac{1}{2})\tau$, and $f(t,u) = -1$ for $(j+\frac{1}{2})\tau \le t < (j+1)\tau$. Here $j = 0, 1, \ldots J$, and J is given.
(a) Verify the conditions for the well-posedness for $T = \tau/4$. Does the theory apply?
(b) Repeat (a) for $T = J\tau$ and $J > 1$.
(c) Find the solution $u(t), t \in [0,T], T = \tau J$, and $J > 1$.
Since f is not smooth, you should think carefully how you would relax the notion of the solution to (1)*. Describe your notion. Did you contradict (b)?*
(d, MTH 552) Propose a method to regularize f by a smoother function f_ε. Discuss the properties of the corresponding u_ε.

Solution notes. (*) The function $f(\cdot,\cdot)$ does not depend on u, thus it is Lipschitz constant with $L = 0$.
(a) Here $f(\cdot,\cdot)$ is continuous on $\mathcal{D} = [0,T] \times \mathbb{R}$, thus from (*) there exists a unique solution to the problem on $\mathcal{D}$.
(b) Now $f(\cdot,\cdot)$ has (multiple) discontinuities, thus the well-posedness holds only on each of the half time subintervals.
(c) We can try to piece together the solution, which comes to be a sawtooth piecewise-linear with slopes 1 and -1. (See code below). We recognize however that the solution is not differentiable at any of the points $j\tau/2$.
(d) We can regularize f by some f_ε which is continuous (or more). For example, consider the following modification to the function $sgn(x)$ close to its jump from -1 to 1 at $x = 0$:

$$sgn_\varepsilon(x) = \begin{cases} -1, & x < -\varepsilon \\ \frac{x}{\varepsilon}, & |x| < \varepsilon \\ 1, & x > \varepsilon \end{cases} \tag{2}$$

Problem 2. Implement FE method for the problem above and test its convergence when $\tau = 1$. (Use the HW template provided at `http://math.oregonstate.edu/~mpesz/latex.html` to see how to assess the error). Provide appropriate plots and tables.
(a) Consider the case $T = \tau/4$ first.
(b) Consider $J = 5, T = J\tau$ next.
(c) If possible, determine h so that the error $e(h)|_{t=T}$ is less than 10^{-4}.
(d, MTH 552 students) Consider the regularized problem as in 1.(d), and apply FE to this problem. (You can use ode45 with a very small time step as a proxy for the exact solution to the regularized problem). Vary both ε and h. What choice yields the best results?

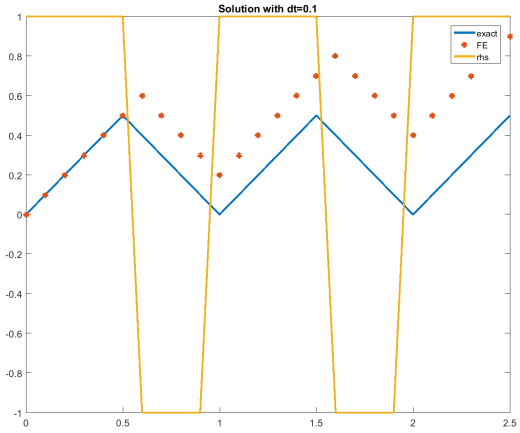

FIGURE 1. Plot of the rhs function, true solution, and FE solution

Solution notes and code. We can adapt the code `FEerr.m` provided in the class notes. The only difficulty is coding the piecewise functions defining the function f and the exact solution $u(t)$; see CODE.

Convergence: In case (a) we find that the solution is linear and FE finds the solution perfectly well, thus the error is close to machine 0. For (b), we see generally first order convergence, but care is advised due to the roughness of $f(\cdot)$. The error may even increase when h increases, which is always scary. The reason for this is how the time steps are distributed close to the points of the discontinuity of $f(\cdot)$.

For example, running the code below with $h = 1/11$ and $h = 1/12$, or $h = 1/9$ and $h = 1/10$. But consider $h = 1/11$ and $h = 1/110$ and $h = 1/1110$. Compare $h = 1/10$ and $h = 1/11$, etc.

For (c) and my code, anything less than $1/40000$ should work.

Part (d) alleviates the difficulty of roughness, but introduces the modeling error between f and f_ε. That is, comparing u to $u_{\varepsilon,h}$ makes only sense when h and ε are tied together.

```
function FEerr = HW1_piecewiseFE(h)
    %% set up function
    f=@(t,y)(piecewise_constant(t));
    uexact = @(t)(piecewise_linear(t));
    %% set up discretization
    T=2.5;tsteps=(0:h:T)';n=length(tsteps);
    %% the numerical solution will be in vector uh
    uh = 0*tsteps;
    %% initial condition
    uh(1)=uexact(0);
    %% time stepping loop
    for j=2:n, uh(j)=uh(j-1)+h*f(tsteps(j-1),uh(j-1));end;
    %% plot the numerical and exact solutions
```

```
    plot(tsteps,uexact(tsteps),'-',tsteps,uh,'*',tsteps,f(tsteps,uh));legend('exact','FE
    title(sprintf('Solution with dt=%g',h));
    %% calculate error
    FEerr = norm(uh-uexact(tsteps),inf);
end
function v = piecewise_constant(xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% define v(xx) piecewise
%% v(x) = A, if j \myeps \leq x \leq (j+1/2) \myeps
%% v(x) = B, if (j+1/2) \myeps \leq x \leq (j+1) \myeps
%% here j=0,1,...maxJ
%%% A, B, myeps, max J are hard coded
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% examples of hardcoded parameters: uncomment these to change
    %% myeps = 1/5; A = 1; B = -1; maxJ = 1;
    %%
    myeps = 1; A = 1; B = -1; maxJ = 5;
    %%
    maxX = maxJ*myeps;
    %% scale xx and myeps to the interval (0,1);
    x = xx/maxX; myeps = myeps/maxX;
    %% calculate relative position of a point in the cell (0,myeps)
    %% by shifting to the interval (0,myeps)ans scaling
    y = (x- myeps*floor(x/myeps))/myeps;
    %% initialize v
    v = y;
    %% piecewise define v
    leftind = (y<=0.5);
    rightind = (y>0.5);
    v(leftind) = A;
    v(rightind) = B;
end
function v = piecewise_linear(xx)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% define v(xx) piecewise
%% v(x) = linear1, if j \myeps \leq x \leq (j+1/2) \myeps
%% v(x) = linear2, if (j+1/2) \myeps \leq x \leq (j+1) \myeps
%% here j=0,1,...maxJ
%%% myeps, max J are hard coded,
%%% linear1 and linear 2 have slopes 1,-1 and are globally continuous
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% examples of hardcoded parameters: uncomment these
```

```
    myeps = 1; maxJ = 5;
    %%
    maxX = maxJ*myeps;
    %% scale xx and myeps to the interval (0,1);
    x = xx/maxX; myeps = myeps/maxX;
    %% calculate relative position of x in the cell (0,1);
    y = (x- myeps*floor(x/myeps))/myeps;
    %% initialize v
    v = y;
    %% piecewise define v
    leftind = (y<=0.5);
    rightind = (y>0.5);
    v(leftind) = y(leftind);
    v(rightind) = 1-y(rightind);
end
```

Remark 1 (Morale: caution is needed when dealing with rough ODEs)**.** *The well-posedness may not apply for rough problems, thus it is not clear if we even have differentiable solutions. The numerical methods may not necessarily behave as the theory predicts.*
Question: which part of the theory of errors does not apply? Why?