Homework #2

ES_APPM 346-0

Due Feb. 8, 2021

Introduction

One method by which polymers are formed is via the intermediate formation of radicals. By radicals, we mean organic species which are chemically active, i.e. compounds which have electrons free to form covalent bonds. In this project you will consider a simplified model of this process.

Consider a solution which initially contains two compounds given by I and M. The compound M is an organic chemical which is a monomer. The objective of polymerization is to attach many copies of M together via some covalent bond to form a polymer. Let P denote the polymer. Polymers can be made in various lengths, but in this model we will not distinguish between polymers of different lengths.

The monomer M will not form a polymer by itself, but instead requires the assistance of the initiator I. The initiator does not react directly with M, but first decomposes into two radicals R. Think of I as two organic structures connected together by a weak covalent bond. When I decomposes, the two radicals become chemically active.

We now describe the chemical process. The first reaction is the decomposition of the intiator into two radicals, which is irreversible:

$$I \to 2R.$$

Once the radical R is formed, it reacts with M to form a radical P_1. This is called the initiation step

$$R + M \to P_1.$$

The species P_1 is a polymeric radical, it is a polymer chain of length one and is chemically active because it has a free electron available for forming bonds which it inherited from R. This is the beginning of the polymerization process which will connect a chain of molecules M together. Polymerization now continues by attaching more monomers,

$$P_i + M \to P_{i+1}.$$

This is called the propagation step.

Finally, in order to get a polymer, you have to stop this chain and get a chemically inactive molecule. This is the termination step and in this sequence there are three ways for the reaction to terminate. In one termination mechanism, two radicals can combine into an inert species,

$$R + R \to \text{inert}.$$

A second termination step is when a polymeric radical combines with R. In this case, the result is an actual polymer, which is called a dead polymer because it can no longer grow,

$$P_i + R \to \text{dead polymer}.$$

Finally, two polymeric radicals can combine to also give a dead polymer. In this case, the length of the dead polymer is the sum of the lengths of the two reactants,

$$P_i + P_j \to \text{dead polymer of length } i + j.$$

We now combine all these reactions into a single system. This poses a problem since we do not know the length of any particular polymeric radical and we don't want to distinguish them. Let $\dot{P}$ represent a polymeric radical which is independent of length, and let P represent a dead polymer of arbitrary length. With this representation, then the propagation step is

$$\dot{P} + M \to \dot{P}.$$

Putting this all together, we have five species, I, R, M, $\dot{P}$, P, and six reactions

$$\begin{array}{lrll} \text{Decomposition:} & I & \to 2R & (1) \\ \text{Initiation:} & R + M & \to \dot{P} & (2) \\ \text{Propagation:} & \dot{P} + M & \to \dot{P} & (3) \\ \text{Termination:} & R + R & \to \text{inert} & (4) \\ & \dot{P} + R & \to P & (5) \\ & \dot{P} + \dot{P} & \to P & (6) \end{array}$$

We can now translate this into differential equations. The species I appears only in the decomposition step in which it decays into radicals. Therefore, the equation for the rate of change of I is given by

$$\frac{dI}{dt} = -k_d I.$$

The coefficient k_d is the rate of decomposition and has units of inverse time. We will assume in this project that all reactions are isothermal meaning that the reaction rates are independent of temperature.

The species R increases as a result of decomposition, and is consumed in the initiation and two of the termination reactions. The equation is given by

$$\frac{dR}{dt} = 2k_d I - k_i M R - 2k_t R^2 - k_t R\dot{P}.$$

For this project, we will assume that all the termination processes have the same rate k_t. Note that the last three terms involve the product of two concentrations. This is because the left side of the reaction involves the combination of two species.

The equation for M is similarly derived:

$$\frac{dM}{dt} = -k_i M R - k_p M \dot{P}.$$

Since $\dot{P}$ inherits its free electron from R, it is not unreasonable to assume that $k_p = k_i$, so we will.

Finally, we get equations for $\dot{P}$ and P given by

$$\begin{aligned} \frac{d\dot{P}}{dt} &= k_i M R - k_t R\dot{P} - 2k_t \dot{P}^2 \\ \frac{dP}{dt} &= k_t \dot{P} R + k_t \dot{P}^2. \end{aligned}$$

In this project you will develop an adaptive stiff solver using a combination of BDF 1 and 2 methods and use it to solve this system of equations.

Written Assignment

1. The system of equations we must solve are given by

$$\begin{aligned}
\frac{dI}{dt} &= -k_d I \\
\frac{dR}{dt} &= 2k_d I - k_i MR - 2k_t R^2 - k_t R\dot{P} \\
\frac{dM}{dt} &= -k_i MR - k_i M\dot{P} \\
\frac{d\dot{P}}{dt} &= k_i MR - k_t R\dot{P} - 2k_t \dot{P}^2 \\
\frac{dP}{dt} &= k_t \dot{P} R + k_t \dot{P}^2.
\end{aligned}$$

   The six reactions included in this model are listed (1)–(6) above on pg 2. For each term on the right hand side of this system of ordinary differential equations, label which reaction led to the term. For example, the first (and only) term for the equation for I' is $-k_d I$, and obviously this is a result of the decomposition reaction (1), so that term would be labelled (1). Label the remaining 11 terms with the corresponding reaction.

2. Define $\mathbf{y} = (I, R, M, \dot{P}, P)^T$, then the system of equations can be written in the form $\mathbf{y}' = \mathbf{F}(\mathbf{y})$ since the right hand side does not depend directly on t. Compute the Jacobian $\dfrac{\partial \mathbf{F}}{\partial \mathbf{y}}$ for this system.

3. Show that the system is in equilibrium, i.e. all derivatives are zero, whenever $I = \dot{P} = R = 0$, and show that the equilibrium solution is stable. To do the latter task, show that when the system is in equilibrium the eigenvalues of the Jacobian are less than or equal to zero. You may assume that all the rate constants k_d, k_i, and k_t are positive. Note: in truth, this is not a sufficient test for stability because there is at least one eigenvalue equal to zero, but this is good enough for our purposes here.

Polymerization Assignment

I will provide the same outline of the code for each of these problems in Matlab Grader and let you know when it is set up. Your code will be checked in Matlab grader.

1. The first step in building a stiff solver is building the implicit equation solver. We will be using the BDF first and second order methods in this project so that we can also do adaptive time stepping. This means that each each time step t_n to t_{n+1}, you will have to solve an equation of the form

$$\mathbf{z} = \gamma \Delta t \mathbf{F}(\mathbf{z}) + \mathbf{C} \tag{1}$$

   for $\mathbf{z}$, where $\mathbf{C}$ is a constant vector, and γ is a scalar constant. Remember, we will be using the quasi-Newton method to solve the system. Fill in the details for the following outline in Matlab that will serve as your Newton solver.

```
function z = solver(k,gam,dt,C,yn,tol)
% z = solver(k,gam,dt,C,yn,tol)
% solves z = gamma*dt*F(z) + C
% where
% k = [kd, ki, kt] parameter values
% gam = gamma
% dt = time step size
% C = constant vector
```

```
% yn = initial value for z, and should be the previous time step
% tol = tolerance for the error in the residual.

J = JacF(k,yn);
% initialize z and compute initial residual r
% Build the matrix A = I - dt*gamma*J
while norm(r,2) > tol
% update the value of z. You may use the Matlab backslash operator to compute A^(-1)r
% update the value of the residual
end

end

function yp = F(k,y)
% k = [kd, ki, kt] parameter values

% return the value of F as a column vector of length 5
end

function J = JacF(k,y)
% k = [kd, ki, kt] parameter values

% return the Jacobian dF/dy as a 5x5 matrix
% evaluated at y
end
```

Check your solver by running the case where $\mathbf{C} = \mathbf{y} = (100, 0, 100, 0, 0)^T$, $k_d = 1000$, $k_t = k_i = 1$, $\gamma = 1$, $\Delta t = 10^{-6}$ and verifying that it solves the equation for BDF 1: $\mathbf{z} = \Delta t \mathbf{F}(\mathbf{z}) + \mathbf{C}$ within the tolerance 10^{-7}. Determine by experimentation the maximum step size Δt which will permit convergence for these conditions.

2. Solve the system with the initial conditions $\mathbf{y} = (100, 0, 100, 0, 0)^T$ for the interval $0 \leq t \leq 0.01$ using BDF 1, aka Backward Euler. Fill in the details for this function to do the task:

```
function [t,y] = bdf1test(T,N,k,y0,tol)
% y = bdf1test(T,dt,k,y0,tol)
% T = terminal time
% N = number of time steps
% k = [kd,ki,kt] parameters
% y0 = initial conditions for [I,M,R,Pdot,P] as a column vector
% tol = error tolerance for solver function

y = y0;
dt = T/N;
for n=1:N
        % update y using BDF 1 method and your solver function to solve the implicit s
end
t=(0:N)*dt;
figure(1)
plot(t,y(1,:),'b-',t,y(2,:),'k-',t,y(3,:),'r-',t,y(4,:),'y-',t,y(5,:),'g-');
legend('I','R','M','P''','P');
end
```

Run your code using $N = 1000$, $N = 2000$, and $N = 4000$ and use solver tolerance 10^{-7}. Let $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3$ be the terminal values of the solution. If you assume that $\mathbf{y}^3$ is the exact solution, then you can compare the error in $\mathbf{y}^1$ to the error in $\mathbf{y}^2$ at the terminal time. Do this comparison and then explain why your results indicate your method is first order accurate.

3. Repeat problem 2 substituting BDF 2 for BDF 1. Since BDF 2 is a two-step method, the first time step will be taken using BDF 1, and then all subsequent steps should be taken using BDF 2. Recall that the variable time step BDF 2 method is given by:

$$\mathbf{y}_{n+1} = \frac{(\Delta t_{n-1} + \Delta t_n)^2}{\Delta t_{n-1}(2\Delta t_n + \Delta t_{n-1})}\mathbf{y}_n - \frac{\Delta t_n^2}{\Delta t_{n-1}(2\Delta t_n + \Delta t_{n-1})}\mathbf{y}_{n-1} + \frac{\Delta t_n(\Delta t_n + \Delta t_{n-1})}{2\Delta t_n + \Delta t_{n-1}}\mathbf{F}(\mathbf{y}_{n+1}),$$

where $\Delta t_n = t_{n+1} - t_n$. Also repeat the comparison at the end, only this time you should explain why your results indicate your method is second order accurate.

4. Write an adaptive solver for this system. For the error estimate, suppose that you have computed $\mathbf{y}_{n+1}^{\text{BDF1}}$ and $\mathbf{y}_{n+1}^{\text{BDF2}}$ for a given time step Δt_n, then an estimate for the error is the difference between the two:

$$\text{error} \approx \|\mathbf{y}_{n+1}^{\text{BDF1}} - \mathbf{y}_{n+1}^{\text{BDF2}}\|.$$

Use an initial time step size of $\Delta t = 10^{-7}$, a Newton solver tolerance of 10^{-7}, a global error tolerance of 1, and terminal time $T = 0.1$. Your code should follow this template:

```
function [y,t] = bdf(T,idt,k,y0,tol,itol)
% bdf_yourname(T,k,y0,tol,itol)
% T = terminal time
% idt = initial time step
% k = [kd, ki, kt] parameters
% y0 = initial condition
% tol = error tolerance for adaptive time stepping
% itol = tolerance for Newton solver

dt = idt;
t = 0;
% take one BDF 1 step with time step idt
t = [0,dt];
while t(end) < T
    % set new time step to previous time step so that dt_n == dt_(n-1)

    % compute y1_(n+1) using BDF 1 and step dt_n
    % compute y2_(n+1) using BDF 2 and steps dt_n = dt_(n-1)
    % estimate optimal time step dt_n* using norm(y2-y1,2) as the error estimate

    % compute y1_(n+1) using BDF 1 using dt_n*
    % compute y2_(n+1) using BDF 2 using dt_n* and dt_(n-1)

    while norm(y2-y1,2) > dtn/T*tol  % where dtn = dt_n*

        % estimate optimal time step dt_n* using norm(y2-y1,2) as the error estimate
        % compute y1_(n+1) using BDF 1 using dt_n*
        % compute y2_(n+1) using BDF 2 using dt_n* and dt_(n-1)

    end
```

```
    t = [t, t(end)+dtn];
    y = [y, y1];
end

figure(1)
subplot(2,1,1)
plot(t,y(1,:),'b-+',t,y(2,:),'k-+',t,y(3,:),'r-+',t,y(4,:),'y-+',t,y(5,:),'g-+');
legend('I','R','M','P''','P')
subplot(2,1,2)
plot(t(1:end-1),t(2:end)-t(1:end-1),'k-')
end
```

Note that the optimal time step computed in the algorithm is not the actual optimal, but is 0.9 times optimal and is limited by $\Delta t_n \leq 2\Delta t_{n-1}$ and $\Delta t_n \leq T - t_n$ as discussed in the notes.

You should also note that this is a challenging problem. Any deviation may result in a method that doesn't converge, so you must get the steps right to be successful.