
ES_APPM 346 Project 2

NORTHWESTERN UNIVERSITY

NICOLAS GUERRA
FEBRUARY 8, 2021

Contents

1	Written Assignment	1
1.1	Problem One	1
1.2	Problem Two	1
1.3	Problem Three	1
2	Polymerization Assignment	3
2.1	Problem One	3
2.2	Problem Two	4
2.3	Problem Three	5
2.4	Problem Four	7

1 Written Assignment

1.1 Problem One

Below are the 12 terms of the differential equations labeled with the corresponding reaction.

$$\begin{aligned} & -k_d I \quad (1) \\ & 2k_d I \quad (1) \\ & -k_i M R \quad (2) \\ & -2k_t R^2 \quad (4) \\ & -k_t R \dot{P} \quad (5) \\ & -k_i M R \quad (2) \\ & -k_i M \dot{P} \quad (3) \\ & k_i M R \quad (2) \\ & -k_t R \dot{P} \quad (5) \\ & -s k_t \dot{P}^2 \quad (6) \\ & k_t \dot{P} R \quad (5) \\ & k_t \dot{P}^2 \quad (6) \end{aligned}$$

1.2 Problem Two

Below is the Jacobian $\frac{\delta \mathbf{F}}{\delta \mathbf{y}}$ for the system.

$$\frac{\delta \mathbf{F}}{\delta \mathbf{y}} = \begin{bmatrix} -k_d & 2k_d & 0 & 0 & 0 \\ 0 & -k_i M - 4k_t R - k_t \dot{P} & -k_i M & k_i M - k_t \dot{P} & k_t \dot{P} \\ 0 & -k_i R & -k_i R - k_i \dot{P} & k_i R & 0 \\ 0 & -k_t R & -k_i M & -k_t R - 4k_t \dot{P} & k_t R + 2k_t \dot{P} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

Notice that the T stands for transpose.

1.3 Problem Three

In order to show the system is in equilibrium, one can illustrate that the derivatives are zero when $I = \dot{P} = R = 0$.

$$\begin{aligned}
\frac{dI}{dt} &= -k_d I = -k_d(0) = 0 \\
\frac{dR}{dt} &= 2k_d I - k_i M R - 2k_t R^2 - k_t R \dot{P} = 2k_d(0) - k_i M(0) - 2k_t(0)^2 - k_t(0)(0) = 0 \\
\frac{dM}{dt} &= -k_i M R - k_i M \dot{P} = -k_i M(0) - k_i M(0) = 0 \\
\frac{d\dot{P}}{dt} &= k_i M R - k_t R \dot{P} - 2k_t \dot{P}^2 = k_i M(0) - k_t(0)(0) - 2k_t(0)^2 = 0 \\
\frac{dP}{dt} &= k_t \dot{P} R + k_t \dot{P}^2 = k_t(0)(0) + k_t(0)^2 = 0
\end{aligned}$$

One must also show that, while in equilibrium, the eigenvalues of the Jacobian are less than or equal to zero. Here is the Jacobian when $I = \dot{P} = R = 0$.

$$\frac{\delta \mathbf{F}}{\delta \mathbf{y}} = \begin{bmatrix} -k_d & 2k_d & 0 & 0 & 0 \\ 0 & -k_i M & -k_i M & k_i M & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -k_i M & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

To find the eigenvalues, λ , one must solve for λ in the following equation where $A = \frac{\delta \mathbf{F}}{\delta \mathbf{y}}$ and I is a 5×5 identity matrix.

$$\det(A - \lambda I) = 0$$

$$\det(A - \lambda I) = \begin{bmatrix} -k_d - \lambda & 2k_d & 0 & 0 & 0 \\ 0 & -k_i M - \lambda & -k_i M & k_i M & 0 \\ 0 & 0 & 0 - \lambda & 0 & 0 \\ 0 & 0 & -k_i M & 0 - \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 - \lambda \end{bmatrix}^T = 0$$

Using cofactor expansion, one can simplify the determinant of the matrix.

$$\begin{aligned}
& \begin{bmatrix} -k_d - \lambda & 2k_d & 0 & 0 & 0 \\ 0 & -k_i M - \lambda & -k_i M & k_i M & 0 \\ 0 & 0 & 0 - \lambda & 0 & 0 \\ 0 & 0 & -k_i M & 0 - \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 - \lambda \end{bmatrix}^T = \\
& -k_d - \lambda \begin{bmatrix} -k_i M - \lambda & -k_i M & k_i M & 0 \\ 0 & -\lambda & 0 & 0 \\ 0 & -k_i M & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix}^T
\end{aligned}$$

$$+2k_d \begin{bmatrix} 0 & -k_i M & k_i M & 0 \\ 0 & -\lambda & 0 & 0 \\ 0 & -k_i M & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix}^T = (-k_d - \lambda)(-k_i M - \lambda)(-\lambda)\lambda^2 = 0$$

Solving for λ , one finds that the eigenvalues are $-k_d$, $-k_i M$, and 0 with a multiplicity of three. Assuming k_d and k_i are positive and seeing that M can physically only be greater than zero, one sees that all the eigenvalues are less than or equal to zero showing that the equilibrium solution is stable.

2 Polymerization Assignment

2.1 Problem One

After building a quasi-Newton method solver $solver(k, gam, dt, C, yn, tol)$ where


- $k = [k_d, k_i, k_t]$ 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

the following inputs were given and returned the following converged result.
Inputs

- $k = [1000, 1, 1]$
- $gam = 1$
- $dt = 10^{-6}$
- $C = [100; 0; 100; 0; 0]$
- $yn = [100; 0; 100; 0; 0]$
- $tol = 10^{-7}$

Output

- $z = [99.9001; 0.1998; 100; 0; 0]$

After experimenting with the tolerance for the error, it seems that the maximum tolerance can be and still return a result that converges like above is $tol = 10^{-1}$. 

2.2 Problem Two

After creating a Backward Difference Formula 1 Program , *bdf1()*, by using *solver()*, one can then take the inputs from above to find a solution to the system. While doing so, one can also see how the number of steps in the program can have an effect on the solution. Below are the solutions using *bdf1()* with $N = 1000$, $N = 2000$, and $N = 4000$.

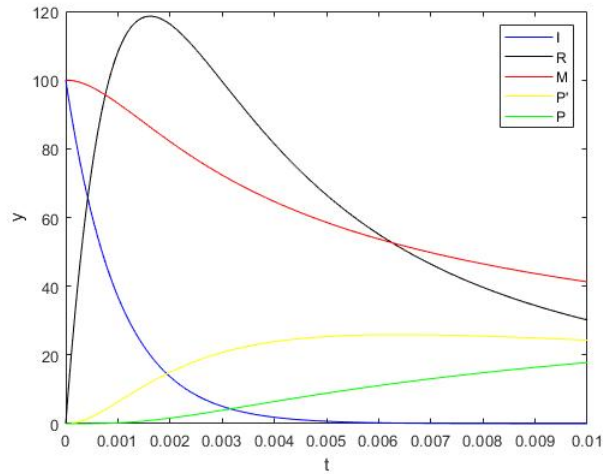


Figure 1: Numerical solution after using Backwards Difference Formula 1 method with $N=1000$

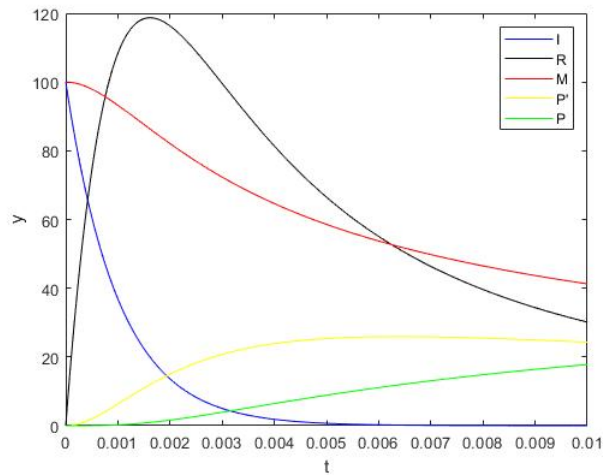


Figure 2: Numerical solution after using Backwards Difference Formula 1 method with $N=2000$

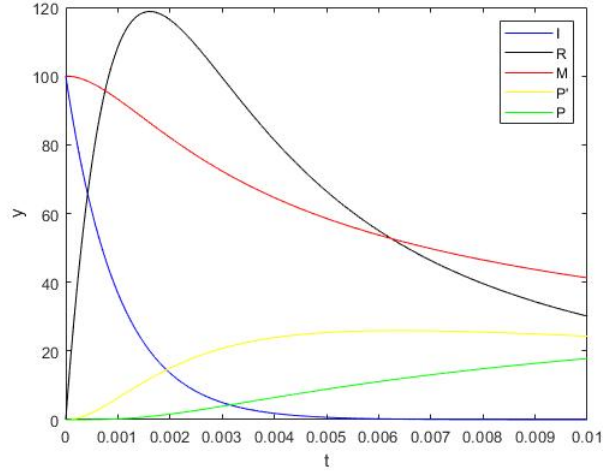


Figure 3: Numerical solution after using Backwards Difference Formula 1 method with $N=4000$

First impressions of each of the three solutions is that they are identical but that is actually not the case. Let \mathbf{y}^1 , \mathbf{y}^2 , and \mathbf{y}^3 be the last values of Figure 1, 2, and 3, respectively. Assuming that \mathbf{y}^3 is the exact solution, here is the error of \mathbf{y}^1 and \mathbf{y}^2 by find the 2-norm.

Table 1: Error of \mathbf{y}^1 and \mathbf{y}^2

\mathbf{y}^1	\mathbf{y}^2
0.0602	0.0201

One can see that the error in \mathbf{y}^2 decreased by a factor of three when compared with \mathbf{y}^1 . Seeing both error values, one can note that $bdf1()$ is first order accurate because the error is less than or equal to a constant times the step size. To find the constant in this occasion, one can plug in the value of .0602 into the error and $1e-5$ for the step size to find that the constant is greater than or equal to 6020. Plugging the constant back in with a new step size of $5e-6$, one can see that the result is greater than 0.0201. This holds true that the method is first order accurate.

2.3 Problem Three ♥

After creating a Backward Difference Formula 2 Program , $bdf2()$, by using $solver()$, one can then take the same inputs from problem one to find a solution to the system. While doing so, one can also see how the number of steps in the program can have an effect of the solution. Below are the solutions using $bdf1()$ with $N = 1000$, $N = 2000$, and $N = 4000$.

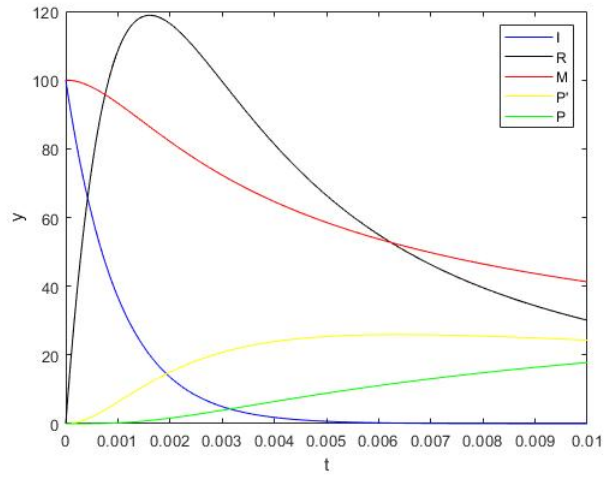


Figure 4: Numerical solution after using Backwards Difference Formula 2 method with $N = 1000$

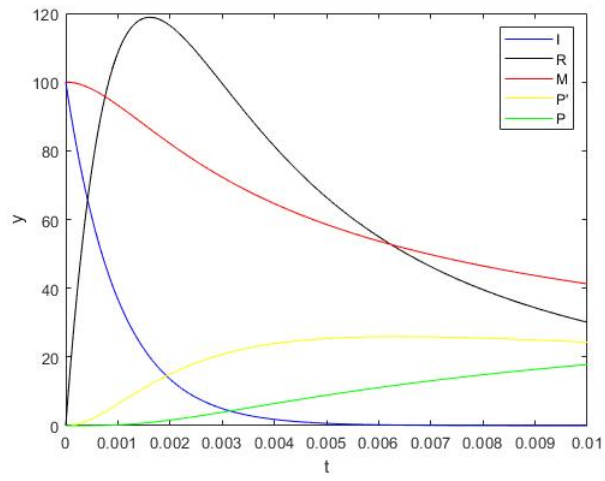


Figure 5: Numerical solution after using Backwards Difference Formula 2 method with $N = 2000$

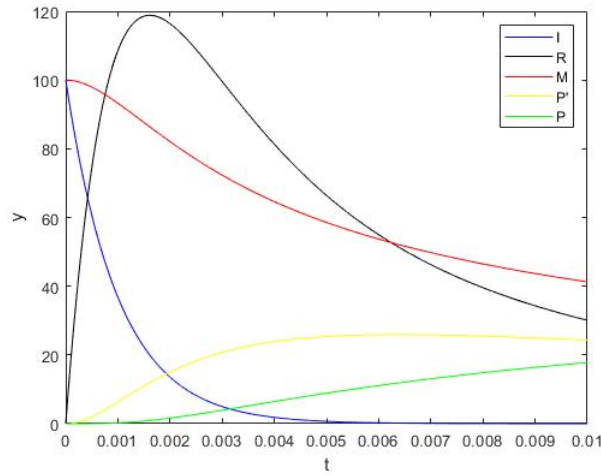


Figure 6: Numerical solution after using Backwards Difference Formula 2 method with $N = 4000$

Once again, first impressions of each of the three solutions is that they are identical but that is actually not the case. Let y^4 , y^5 , and y^6 be the last values of Figure 4, 5, and 6, respectively. Assuming that y^6 is the exact solution, here is the error of y^4 and y^5 by finding the 2-norm.

Table 2: Error of y^4 and y^5

y^1	y^2
2.0741e-4	4.3142e-5

y^2 having more steps was able to out perform y^1 in having a smaller error. $bdf2()$ also outperforms $bdf1()$ in having less error because $bdf2()$ uses two values to interpolate. It is important to note that $bdf2()$ is, in fact, second order accurate because the error is less than or equal to a constant times the step size squared. More specifically, to find the constant by letting the step size equal $1e-5$ and the error be $2.0741e-4$, the constant is greater than or equal to 2074100. Plugging this constant back in with the step size being $5e-4$, one finds that the result is greater than $4.3142e-5$. This supports the claim that $bdf2()$ is second order accurate. ▀

2.4 Problem Four ▀

During the program `bdf_nicolasguerra()`, the step sized will now be optimized 90%. This means that the step size will no longer be uniform, but rather change. Using an initial step size of 10^{-7} , Newton solver tolerance of 10^{-7} , global error tolerance of 1, and a terminal time of .1, one produce the following graphs.

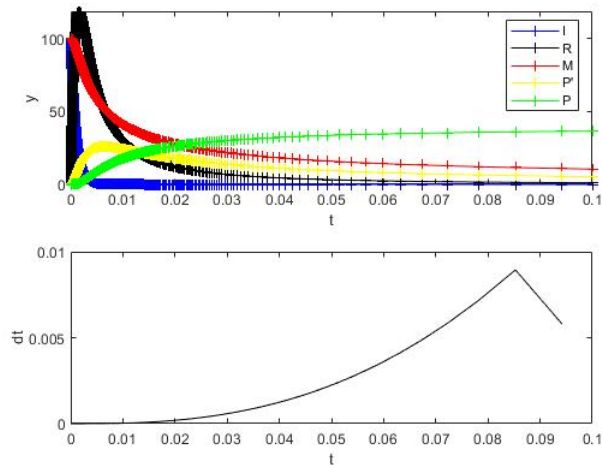


Figure 7: Numerical solution from an adaptive solver after using Backwards Difference Formula methods 1 and 2 along with the step size over time

Looking at the top graph, it looks no different than the previous ones except now we have a wider time range to view. Looking at the bottom graph, one can see that the time step starts off extremely small and then increases exponentially until a certain point where it dips down. Using an adaptive time stepper is extremely useful when it comes to stiff systems. Using larger uniform time steps would yield inaccurate results, so the only way to get better results with uniform time steps would be to cut them in size. However, Although using a very small time step would give better results, each section of the solution must use the same time step even when unnecessary. This could be inefficient and costly. Instead, with this adaptive time stepper, the regions are set up with an optimal time step suited for them. This makes the process run smoother and yields accurate results as well.