MatLab - Systems of Differential Equations

This section examines systems of differential equations. It goes through the key steps of solving systems of differential equations through the numerical methods of MatLab along with its graphical solutions. The system of differential equations is introduced. Analysis begins with finding equilibria. Near an equilibrium the linear behavior is most important, which requires studying eigenvalue problems. Numerical routines can simulate the system of differential equations, while the special routine *pplane* allows easy study of the system with excellent graphics.

The general two dimensional autonomous system of differential equations in the state variables $x_1(t)$ and $x_2(t)$ can be written:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2), \\ \dot{x}_2 &= f_2(x_1, x_2), \end{aligned}$$

where the functions f_1 and f_2 may be nonlinear. This section will concentrate on the case when f_1 and f_2 are linear to parallel the lecture notes, **Systems of Two First Order Equations**. The **Greenhouse/Rockbed Model** from the lecture notes is given by the linear system of differential equations:

$$\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \end{pmatrix} = \begin{pmatrix} -\frac{13}{8} & \frac{3}{4} \\ \frac{1}{4} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \begin{pmatrix} 14 \\ 0 \end{pmatrix}.$$

This example will provide the primary case for our **MatLab** commands listed below.

Equilibria

Equilibria occur when the derivative is zero. In the general case where the right hand side of the system of differential equations is nonlinear, this problem can be very complex. However, when the functions f_1 and f_2 are linear, then finding equilibria reduces to solving a linear system of equations. This is very basic for MatLab and can be accomplished in a number of ways.

For the greenhouse/rockbed model above, the equilibrium model satisfies:

$$\begin{pmatrix} -\frac{13}{8} & \frac{3}{4} \\ \frac{1}{4} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} u_{1e} \\ u_{2e} \end{pmatrix} = \begin{pmatrix} -14 \\ 0 \end{pmatrix}. \tag{1}$$

This is more simply written:

$$\mathbf{A}\mathbf{u}_e = \mathbf{b},$$

where $\mathbf{A}$ is the matrix of coefficients, $\mathbf{u}_e$ is the equilibrium solution, and $\mathbf{b}$ is the nonhomogeneous vector from the external environment.

Below we provide **three** ways in MatLab to find $\mathbf{u}_e$, the equilibrium solution. The two most common means to solving this linear system are to use the program *linsolve* or to take advantage of MatLab's ability to invert a matrix. Define the variables, then the following commands readily provide the solution.

```
A = [-13/8 3/4;1/4 -1/4]; b = [-14;0];
u = linsolve(A,b)
u = inv(A)*b
```

Both results produce the variable $\mathbf{u} = [16, 16]^T$ for the equilibrium solution.

The third method is to use the row-reduced echelon form for transforming an augmented matrix into the identity with the solution in the last column. The MatLab program for this is *rref*. (There used to be a common teaching tool called *rrefmovie*, which showed all the steps of the process including the names of the operation, but this function appears to have been removed after Version 10 of MatLab.) Below we show the commands necessary for this solution method.

```
B = [A,b]
rref(B)
```

The results are the following:

$$B = \begin{bmatrix} -1.6250 & 0.7500 & -14.0000 \\ 0.2500 & -0.2500 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 16 \\ 0 & 1 & 16 \end{bmatrix}$$

Linear Analysis

As noted in the lecture notes, **Systems of Two First Order Equations**, the original **state variable**, $\mathbf{u}$, is translated to the new state variable, $\mathbf{v} = \mathbf{u} - \mathbf{u}_e$, resulting in the new linear system of differential equations centered at the origin:

$$\begin{pmatrix} \dot{v}_1 \\ \dot{v}_2 \end{pmatrix} = \begin{pmatrix} -\frac{13}{8} & \frac{3}{4} \\ \frac{1}{4} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{2}$$

or more simply

$$\dot{\mathbf{v}} = \mathbf{A}\mathbf{v}.$$

For this problem we seek solutions $\mathbf{v}(t) = \xi e^{\lambda t}$, and the result is the *eigenvalue problem*:

$$(\mathbf{A} - \lambda\mathbf{I})\xi = \mathbf{0}, \qquad \text{where} \qquad \det|\mathbf{A} - \lambda\mathbf{I}| = 0,$$

provides the **characteristic equation** for the eigenvalues and ξ are the corresponding eigenvectors. For this system with distinct real eigenvalues, λ_1 and λ_2, and corresponding eigenvectors, $\xi^{(1)}$ and $\xi^{(2)}$, the solution of System (2) satisfies:

$$\mathbf{v}(t) = c_1\xi^{(1)}e^{\lambda_1 t} + c_2\xi^{(2)}e^{\lambda_2 t},$$

where c_1 and c_2 are arbitrary constants.

Once again MatLab is an excellent program for solving this eigenvalue problem. The MatLab command

```
[v,d] = eig(A)
```

produces the eigenvalues on the diagonal of matrix d with the corresponding eigenvectors appearing as columns of matrix v. For this example MatLab produces:

$$d = \begin{bmatrix} -1.7500 & 0 \\ 0 & -0.1250 \end{bmatrix} \qquad v = \begin{bmatrix} -0.9864 & -0.4472 \\ 0.1644 & -0.8944 \end{bmatrix}, \tag{3}$$

which gives the general solution to System (2) as

$$\mathbf{v}(t) = c_1 \begin{pmatrix} -0.9864 \\ 0.1644 \end{pmatrix} e^{-1.75t} + c_2 \begin{pmatrix} 0.4472 \\ 0.8944 \end{pmatrix} e^{-0.125t}.$$

Solution to the System of Linear Differential Equations

Since $\mathbf{u}(t) = \mathbf{v}(t) + \mathbf{u}_e$, it follows that the general solution to (1) is

$$\mathbf{u}(t) = c_1 \begin{pmatrix} -0.9864 \\ 0.1644 \end{pmatrix} e^{-1.75t} + c_2 \begin{pmatrix} -0.4472 \\ -0.8944 \end{pmatrix} e^{-0.125t} + \begin{pmatrix} 16 \\ 16 \end{pmatrix}.$$

The specific solution to an initial value problem, where $\mathbf{u}(0) = \mathbf{u}_0$, is readily found using MatLab and solving the vector equation:

$$c_1\xi^{(1)} + c_2\xi^{(2)} + \mathbf{u}_e = \mathbf{u}_0.$$

We use our example above with $\mathbf{u}_0 = [5, 25]^T$ (or $\mathbf{v}_0 = [-11, 9]^T$) to illustrate the appropriate MatLab commands. (Assume that the eigenvectors are stored in v as presented in (3).) Let $\mathbf{c} = [c_1, c_2]^T$ be vector for which MatLab is solving, then

```
v0 = [-11, 9]';
c = linsolve(v,v0)
```

produces the solution $\mathbf{c} = [14.5050, -7.3962]^T$, so the unique solution to the IVP is

$$\mathbf{u}(t) = 14.5050 \begin{pmatrix} -0.9864 \\ 0.1644 \end{pmatrix} e^{-1.75t} + 7.3962 \begin{pmatrix} 0.4472 \\ 0.8944 \end{pmatrix} e^{-0.125t} + \begin{pmatrix} 16 \\ 16 \end{pmatrix}.$$

The MatLab's numerical solver, *ode23*, extends easily to systems of 1st order differential equations. Below we show how to both use the numerical solver and the exact solution above to graph the $\mathbf{u}(t)$. First the right hand side of System (1) is made into a MatLab function

```
function yp = greenhouse(t,y)
% Greenhouse DE (rhs)
yt1 = -(13/8)*y(1) +(3/4)*y(2) + 14;
yt2 = (1/4)*y(1) - (1/4)*y(2);
yp = [yt1,yt2]';
end
```

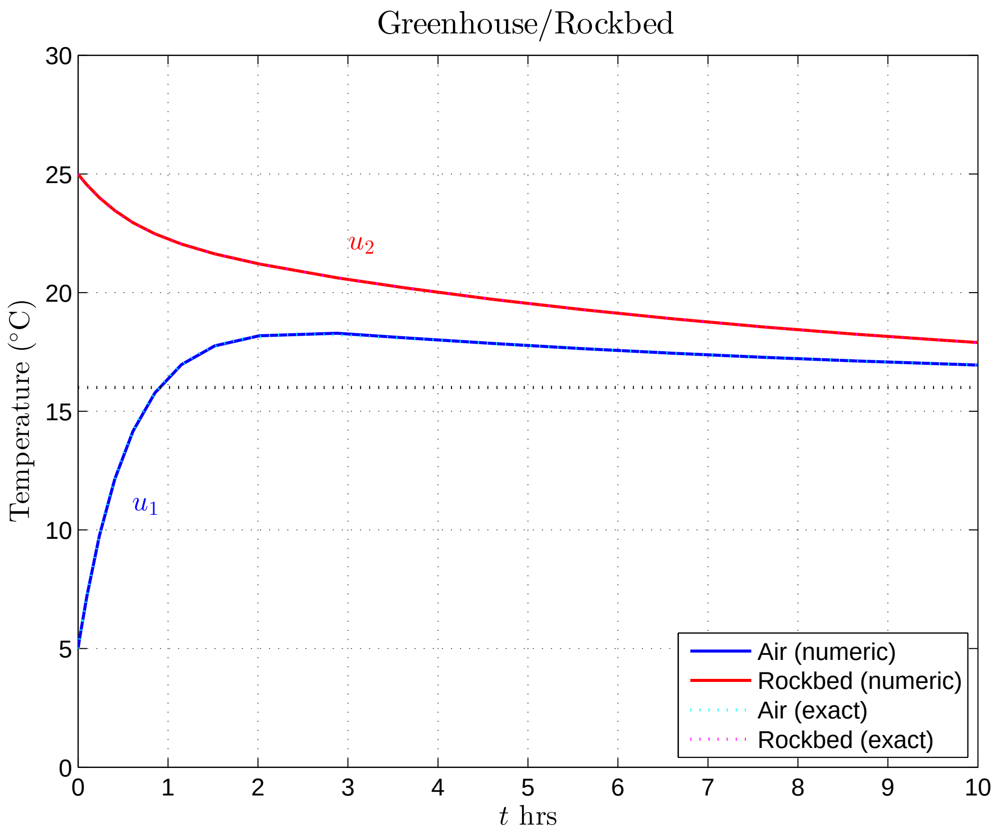
The function plotting the numerical and exact solutions is given by

```
mytitle = 'Greenhouse/Rockbed';          % Title
xlab = '$t$ hrs';                        % X-label
ylab = 'Temperature ($^\circ$C)';                    % Y-label

u0 = [5,25]';
[t,u] = ode23(@greenhouse,[0,10],u0); % simulate heat with ode23
tt = linspace(0,10,200);
u1 = -14.3077*exp(-1.75*tt)+3.3077*exp(-0.125*tt)+16; % solution u1
u2 = 2.3846*exp(-1.75*tt)+6.6154*exp(-0.125*tt)+16;   % solution u2

plot(t,u(:,1),'b-','LineWidth',1.5);  % Plot greenhouse air (numeric)
hold on                               % Plots Multiple graphs
plot(t,u(:,2),'r-','LineWidth',1.5);  % Plot greenhouse rocks (numeric)
plot(tt,u1,'c:','LineWidth',1.5);  % Plot greenhouse air, u1
plot(tt,u2,'m:','LineWidth',1.5);  % Plot greenhouse rocks, u2
plot([0 10],[16 16],'k:','LineWidth',1.5);  % Plot equilibrium
grid                                  % Adds Gridlines
text(0.6,11,'$u_1$','color','blue','FontSize',14,...
    'FontName','Times New Roman','interpreter','latex');
text(3,22,'$u_2$','color','red','FontSize',14,...
    'FontName','Times New Roman','interpreter','latex');
legend('Air (numeric)','Rockbed (numeric)','Air (exact)',...
    'Rockbed (exact)',4);

axis([0 10 0 30]);  % Defines limits of graph
```

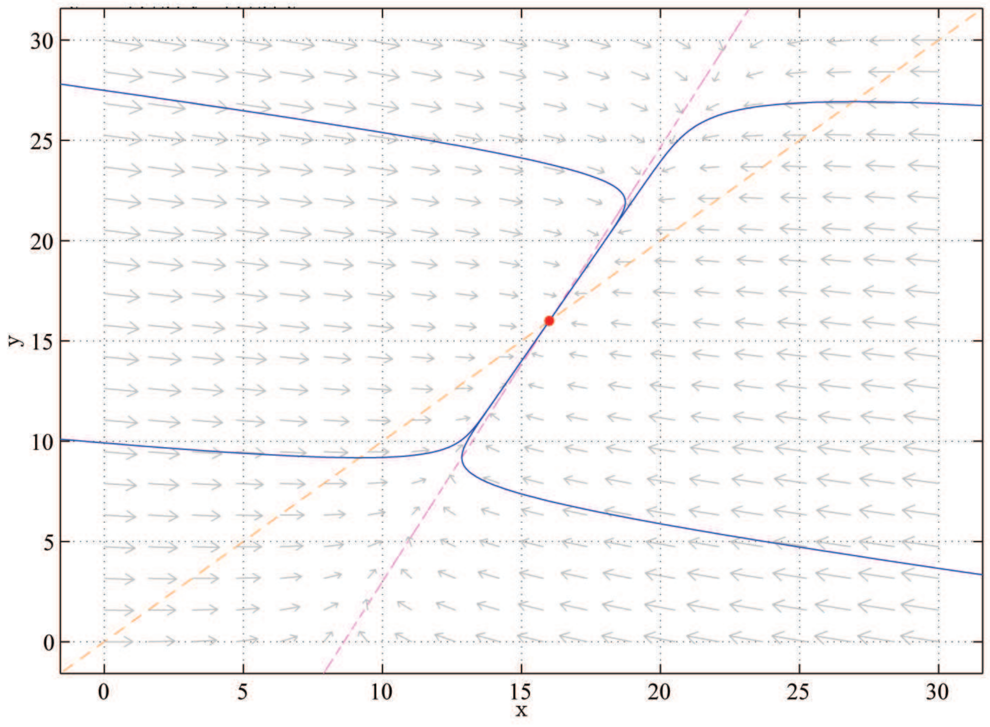


The graph shows how well the numerical routine *ode23* in MatLab tracks the solution to System (1). As we saw in the lecture notes, the heat transfers rapidly into the air compartment, then slowly the solution tends toward the equilibrium solution.

Phase Portrait - 2D

This final section shows how to create **two dimensional phase portraits** and **direction fields**. You begin by downloading the MatLab files for pplane and dfield by John Polking from Rice University. The current version is *pplane8*, which is invoked by having this m-file in your current directory and typing `pplane8` in the command window of MatLab.

After invoking *pplane8* a window appears where you type in your differential equation, including the limits on your **state variables**. This will generate a new window showing the direction field for the phase portrait. With the mouse you can click at any point and have a solution trajectory be drawn. Default for this is both directions in time. (To get a forward **trajectory** only one select **Solution direction** under the **Options** menu.) One of the most powerful features is the ability of this program to find **equilibria** and determine the **eigenvalues** near that equilibrium. This feature is found under the **Solutions** menu saying **Find an equilibrium point**. Select this feature, then click on the graph where you think an equilibrium point may be. A large **red dot** appears at the equilibrium, and a new window opens telling the coordinates of the equilibrium, the nature of the equilibrium, and the eigenvalues and eigenvectors of the linearized system near the equilibrium. Another useful feature under the **Solutions** menu is to **Show nullclines**. The nullclines tell where **trajectories** are either horizontal or vertical. The intersection of two different color nullclines shows the location of an equilibrium. Below is a figure showing the direction field for our example. Four solutions were added along with the equilibrium and the nullclines.



Nonlinear Example

The *pplane8* program is particularly useful for **systems of two nonlinear differential equations**. A competition model for two species y_1 and y_2 competing for the same resource satisfies the following system of differential equations:

$$\begin{aligned}\dot{y}_1 &= 0.1y_1(1 - 0.022y_1 - 0.03y_2),\\ \dot{y}_2 &= 0.12y_2(1 - 0.037y_1 - 0.024y_2).\end{aligned}$$

Though it is not the case for this system of differential equations, it can be quite challenging finding the equilibria for a nonlinear system of differential equations. MatLab does have a powerful tool for solving nonlinear systems of equations to find where they are zero, and it is called *fsolve*. The MatLab function *fsolve* requires entering a function $\mathbf{f}(\mathbf{x})$, which can be a vector function, and an initial guess, $\mathbf{x}_0$, then it tries to find the closest $\mathbf{x}$ that solves $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. If we define the right hand side of the DE above as the following MatLab function:

```
function z = compet( y )
% competition model: fsolve for equilibria
zt1 = 0.1*y(1)*(1-0.022*y(1)-0.03*y(2));
zt2 = 0.12*y(2)*(1-0.037*y(1)-0.024*y(2));
z = [zt1,zt2];
end
```

If we enter the following in MatLab:

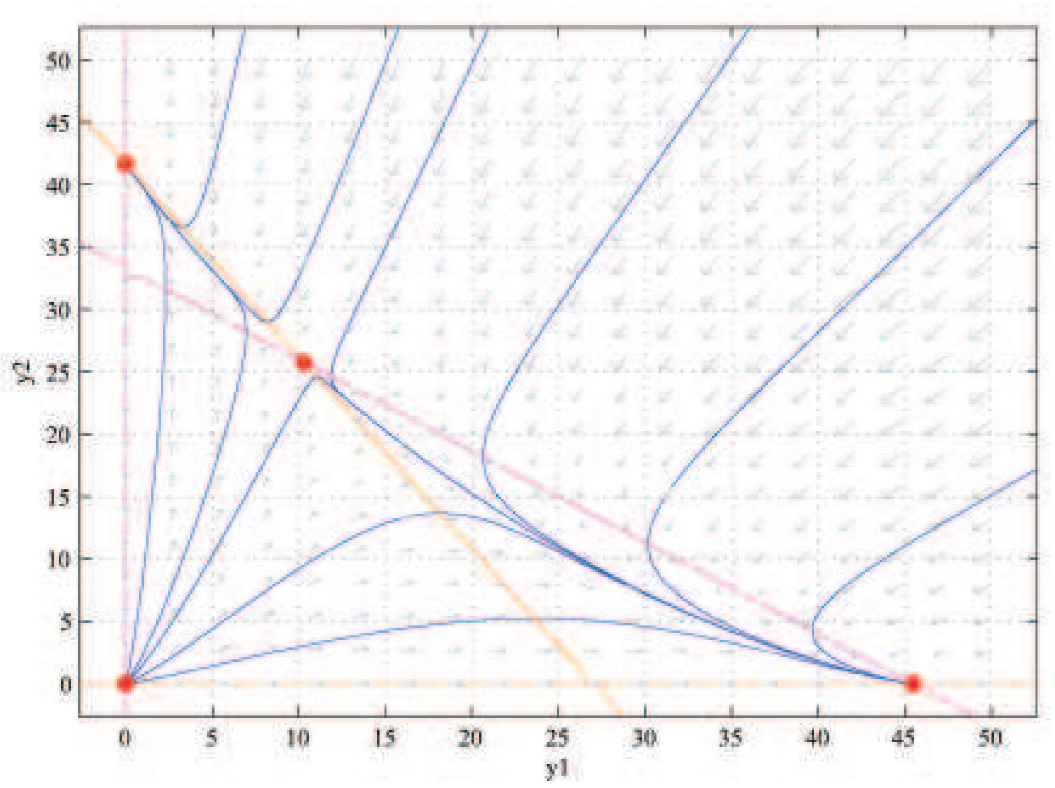
```
ye = fsolve(@compet,[10,20]);
```

then MatLab gives the **unstable equilibrium**

$$\mathbf{y}_e = [10.3093, 25.7732]^T.$$

There are **3** other equilibria, which could be found by similarly using our *compet* function with different initial guesses.

More easily, one can employ *pplane* as noted above. It readily finds equilibria and the eigenvalues associated with each equilibrium. For example, pointing near $\mathbf{y}_e = [10.3093, 25.7732]^T$, *pplane8* finds this equilibrium, and tells you that it is a saddle node with eigenvalues $\lambda_1 = 0.01638$ and $\lambda_2 = -0.1133$ (along with their corresponding eigenvectors). Below we show the figure produced by *pplane8*.


y2
y1
0
5
10
15
20
25
30
35
40
45
50