

Math 242: Modern Computational Mathematics (MCM)
Professor McKelvey
Instructions for `lsoda` R Package

1 Introduction

This assignment involving the numerical solution of systems of differential equations. We will use an R package called `odesolve` in this work.

The first step is to be sure this package is installed on the machine where you are using R. You may need to perform this installation on your personal machine, or on a Saint Olaf owned machine. On Saint Olaf machines you may need to re-install the package every time you log in. (This is according to IIT.) Fortunately, installation is very simple and quick.

2 Installation of `odesolve` Package.

To install the `odesolve` package, launch the R program. Then type in the command:

```
install.packages("odesolve")
```

A menu of CRAN sites will appear. (CRAN sites are repositories for R software packages.) Choose the one labeled "USA (IA)". The installation should proceed immediately.

To use the functions in the library, something necessary for this assignment, issue the following command after installation is complete:

```
library(odesolve)
```

The differential equations solvers of this package are now available to you.

3 Solving Differential Equations using R

The particular function we will use is called `lsoda`. I will describe this function in general and then provide two examples of its use.

The `lsoda` function takes six arguments of the form:

```
lsoda(ics, times, func, parms, rtol, atol)
```

where the meaning of each argument is given below.

ics An R vector of initial conditions. This vector should contain one entry for each variable in the differential equation.

times An R vector of times for which you want R to list the values of all the variables you are finding with the differential equation.

func A function representing the right hand side of the system of differential equations. More details on **func** will be given below.

parms An R vector of parameter values. These are the values of any constants that appear in the differential equation. You get to choose the order in which these constants appear, but the order you choose must be consistent with the definition of **func**. (See below for details.)

rtol A strictly positive floating point number indicating a relative tolerance for error in the solution algorithm. For our purposes, use 0.001.

atol A strictly positive floating point number indicating an absolute tolerance for error in the solution algorithm. For our purposes, use 0.001.

Let's talk more about the **func** argument. This is an R function that represents the right hand side of the differential equation. The definition of **func** always begins with the following line:

```
yourfuncname <- function(t,y,params) {
```

where you can replace the name with whatever you want.

The **t** argument represents the time, something we will usually (but not always, see HW) ignore in our work. The **y** argument is an R vector of all the variables in our differential equation. The values of the entries in **y** are what get updated as time moves forward. Lastly, the **parms** argument is a vector of parameter values, exactly the same vector as the **parms** argument of **lsoda**.

The object returned by **func** is a little strange. It is an R *list* consisting of two elements. The first element is an R vector consisting of the values of the derivatives of all the **y**'s, or put another way, it is the left hand side of the differential equation. The second element of the list will always been an empty vector (denoted **c()** in R). This will, I hope, be made clearer by the examples.

3.1 The Matrix Returned by **lsoda**.

The differential equation solver **lsoda** returns an R matrix. Each row of the matrix corresponds to one of the times specified in the **times** vector that was given to **lsoda** when it was called. The first column of the matrix gives the time for that row. The remaining columns show the values of the various functions **y** that are calculated by the solver. The second column gives the values for the first variable, the third column gives the values for the second variable, and so on.

These columns can then be used to produce various plots, etc., that might be of interest to you.

3.2 The First Example

Suppose we are interested in solving the simple differential equation

$$\dot{y} = \alpha \cdot y, \quad y(0) = 1$$

for various values of α and time values on the interval $[0, 2]$.

In this example, the only variable in the problem is y . The value *alpha* is a constant, and thus a parameter. Distinguishing between variables and parameters is important when using `lsoda`.

The first step is to write `func`, the function that represents the right hand side of the differential equation. We will name this function `simprhs`. The following code will do the trick:

```
simprhs <- function(t,y,params) {  
  retv <- params[1]*y[1]   #alpha is the first parameter (params[1])  
                           #y is the first (only) variable (y[1])  
  list(dy=retv,global=c()) #the list to be returned  
}
```

The R variable `retv` holds the “returned vector,” the vector of right hand side values, which consists of exactly one value in this simple case. The `dy` and `global` terms in the last line of the function serve only to give names to the list’s fields.

We are now read to use `lsoda` to solve this differential equation for various values of the parameter α . Let’s begin with $\alpha = -0.3$. The call to `lsoda` that solves our problem in this case is:

```
lsoda(c(1),c(0,0.5,1,1.5,2),simprhs,c(-0.3),rtol=0.001,atol=0.001) and the fol-  
lowing is returned from R:
```

```
      time      1  
[1,]  0.0 1.0000000  
[2,]  0.5 0.8602627  
[3,]  1.0 0.7401386  
[4,]  1.5 0.6374113  
[5,]  2.0 0.5477588
```

We see that this is a matrix, the first column gives a listing of times and the second column gives y values corresponding to these times. To create plots of these values you would start by giving the output of the `lsoda` call a name, say `results`, and then plotting the first column of `results` against the second column. Here is how you would do that:

```
results <- lsoda(c(1),c(0,0.5,1,1.5,2),simprhs,c(-0.3),rtol=0.001,atol=0.001)  
plot(results[,1],results[,2],type="l")
```

The true solution to our differential equation, with $\alpha = -0.3$ is $y(t) = e^{-0.3t}$, which is an exponentially decreasing function, consistent with our results.

If we were interested in the solution when our parameter α has a different value, say 2.1, we would simply call `lsoda` a second time with this parameter value. Such a call would be:

```
lsoda(c(1),c(0,0.5,1,1.5,2),simprhs,c(2.1),rtol=0.001,atol=0.001)
```

3.3 A Second Example

In this example we will solve a more complicated system of differential equations involving two variables (x and y) and four parameters (a , b , c and d). This system is known as the Lotka-Volterra system and represents a predator/prey population dynamic system.

If x represents the population of prey, say rabbits, and y represents the population of predators, say foxes, the following has been proposed as a system that describe how the populations of both prey and predators will evolve over time.

$$\begin{aligned}\dot{x} &= ax - bxy \\ \dot{y} &= -cy + dxy\end{aligned}$$

As with the first example, the first thing we need to do is write the function (**func**) that describes the right hand side of our system of differential equations. In this case we have two variables, so we will decide that x is the first variable, and y is the second. Similarly, we have four parameters and we will decide they appear in alphabetical order, so a will be the first parameter, b the second, and so on. If we name this function `lotkarhs`, the following R code will correctly define our right hand side function.

```
lotkarhs <- function(t,y,params) {  
retv <- {c(params[1]*y[1] - params[2]*y[1]*y[2],  
          -1.0*params[3]*y[2] + params[4]*y[1]*y[2]) }  
list(dy=retv,global=c())  
}
```

The curly brackets on lines 2 and 3 of this code are there because the vector definition takes more than one line.

Once we agree on initial values for x and y , and decide on parameter values, we are ready to call `lsoda`.

Let's agree that the initial conditions are $x(0) = 40$ and $y(0) = 10$. Let's also agree on the following parameter values: $a = 2$, $b = 0.15$, $c = 0.1$, and $d = 0.002$.

Lastly, let's agree that we are interested in values of time going from zero to 10, and we want reports every 0.5 time periods. The following R command will give us the information we seek:

```
{lsoda(c(40,10),seq(0,10,by=0.5),lotkarhs,c(2,0.15,0.1,0.002),  
rtol=0.001,atol=0.001)}
```

The curly brackets are necessary because this expression takes more than a single line.

The result of calling `lsoda` in this case is the matrix:

	time	1	2
[1,]	0.0	40.00000	10.00000
[2,]	0.5	51.50343	9.95623
[3,]	1.0	66.19411	10.04366
[4,]	1.5	83.97590	10.29790
[5,]	2.0	103.66232	10.76026
[6,]	2.5	122.65362	11.46379

[7,]	3.0	136.34936	12.41907
[8,]	3.5	140.02014	13.57540
[9,]	4.0	131.39892	14.79910
[10,]	4.5	112.87085	15.91263
[11,]	5.0	89.88775	16.75630
[12,]	5.5	68.07407	17.24586
[13,]	6.0	50.38217	17.39818
[14,]	6.5	37.28908	17.28352
[15,]	7.0	28.02867	16.98216
[16,]	7.5	21.64979	16.55720
[17,]	8.0	17.31550	16.05741
[18,]	8.5	14.40074	15.51715
[19,]	9.0	12.48051	14.95923
[20,]	9.5	11.28048	14.39907
[21,]	10.0	10.63086	13.84720