Semi-Linear Elliptic Ordinary Differential Equations

A.Laura Carbajal
Research Experience for Undergraduates Program
Northern Arizona University
1998

Advisor
John M. Neuberger

Table of Contents

Semi-Linear Elliptic Ordinary Differential Equations

A. Laura Carbajal, Northern Arizona University
Dr. John M. Neuberger, Northern Arizona University

Preface

This report depicts my investigations of Semi Linear Elliptic Ordinary Differential Equations, during Northern Arizona University's Research Experience for Undergraduate Program. The project required an intense amount of background reading, as well as effort in my part to pursue new knowledge and combine it with knowledge acquired during my college career. In general I have found that the data collected during the program has been interesting, and at time exciting to my advisor and myself.

Some of the data included in this report may or may not be relevant, but I have included the data as a resource for future individuals.

--A. Laura Carbajal--

Introduction

2. What is a Semi-Linear Elliptic Ordinary Differential Equation?

To be able to define what a Semi-Linear Elliptic Ordinary Differential Equation, we must first establish/define the different components of the title. Let's begin with the term Semi-Linear. A Semi-Linear Equation is of the form L(u)=N(u), where L(u) is a Linear Operator and N(u) is a Nonlinear Function of u. Semi-Linear Equations are a subset of Non-Linear Equations, simply because if an Ordinary Differential Equation is Linear then it is not a Non Linear. Semi-Linear Differential Equations are Differential Equations which are linear in the derivatives of the unknown function and Nonlinear in theunknown function itself. For example, $y''(x)=(y(x))^3$ is linear in y" but cubic in y. Elliptic Equations in the cases I've looked at are PDE where the differential operator is the Laplacian, i.e., $\Delta y = f(y)$, where $\Delta y(x_1,...,x_n)=y_{x1x1}+...+y_{xnxn}$. Ordinary Differential Equation, are Differential Equations with only one independent variable. In the ODE case, $\Delta y=y''$, since there is only one independent variable and hence $\Delta y=y_{1\,x1x1}= y''$. For example, y"= -(λy+y^3), is a Semi-Linear Elliptic Ordinary Differential Equation, since y" is a Linear Operator and -(λy+y^3) is a Non-Linear Function.

3. Methods of Looking at Semi-Linear Elliptic Ordinary Differential Equations:

In this section, I will discuss IVP, Euler's Mehtod, BVP, Shooting Methond, MI, Eigenvalues, Eigenvectors, Eigenfunctions, λ, Bifurcation Digarams, Graphs of Solutions of BVP, Differences Between IVP and BVP.

a. Initial Value Problems

Initial Value Problems (IVPs) have been used in a variety of mathematical areas. They are an important part of the Mathematics world, having the capacity to model many things. By definition an IVP for an nth order differential equation $F\left(x,y,\frac{dy}{dx},\dots,\frac{d^n y}{dx^n}\right)=0$ means finding a solution to the differential equation on an interval I, that satisfies at x_0 the initial conditions $y(x0)=y0$, $\frac{dy}{dx}(x0)=a_1,\dots,\frac{d^{n-1} y}{dx^{n-1}}(x0)=a_{n-1}$ where $x0\in I$ and $a_0, a_1,\dots,a_{n-1}$ are given constants. In the case of a first order ODE, the initial conditions reduce to the single requirement $y(x_0)=y_0$. In the case of a second order differential equation, the initial conditions have the form $y(x_0)=y_0$,

$\frac{dy}{dx}(x_0)=c$. The term initial condition come from mechanics, in which $y(x_0)=y_0$ can represent the location and $\frac{dy}{dx}(x_0)=y'(x_0)$ the velocity, of the object, at time x_0.

b. Euler's Method

Euler's Method is a method of computing points on a solution curve to an IVP, numerically. At times one can find an explicit equation for the solution curve, but often this is not possible. The basic concept behind Euler's method is pick a starting point (corresponding to the initial value), and calculate the slope at that point using the differential equation. This slope tells you which direction to take. Move a short distance (Δy_1), from the initial value and calculate the slope at that point using the differential equation. Using the new slope move a new short distance(Δy_2), and so on.

c. Boundary Value Problem

Boundary Value Problems (BVP) are problems which seeks to determine a solution to a differential equation subject to conditions on the unknown function specified at two or more values of the independent variable. The BVP we will consider are ODE, where instead of one or more conditions at a single point x_0, we specify two values to be the conditions at the endpoints(boundary) of the interval. These conditions are called boundary conditions. In the case of a second order differential equation, the intial conditions have the form $y(x_0)=y_0$. Our two-point boundary value problem of ordinary differential equations invves a second order equation, an intial condition an an endpoint condition. The region I is the interval ($x0$, x_n) and the boundary consists of the two endpoints. A PDE Dirichlet problem requires that the Laplace equation $F_{xx} + F_{yy} = G(x)$, be satisfied inside some region $\mathbb{R}$ of the xy plane and that F(x,y) assume specified values on the boundary of $\mathbb{R}$.

d. Shooting Method

The Shooting Method is a way to use Euler's Method for solving Initial Value Problems(IVP), to generate one or more solutions to a BVP. As you vary $y'(x_0)=c$, one may find IVP solutions which also satisfy the Boundary Value Problem(BVP). The basic concept of this method is to vary $y'(x_0)=c$, and check if the IVP solutions satisfies the BVP. In this way, one may obtain one or possibly more solutions to the given BVP. For example, suppose you positioned a target 100 yards away and you fire a rifle. Are you more likely to hit the target if you fire once or if you fire many times varying your aim(Δy) slightly? The answer, is to fire many shots, thus the probability of hitting the target increases. It is not to say that you cannot hit the target with one shot, but your probability of being successful will increase if the number of shots increase. This analogy applies to the shooting method, the more iterations(shots) you perform the more likely you will be to obtain a solution to the BVP.

e. Morse Index(MI) and Co-Morse Index(CMI)

The Morse Index(MI), in its simpliest form means the number of "down" directions, i.e., $x^2 - y^2 - z^2$ and $x^2 + y^2 - z^2$, have MI 1 and MI 2 critical points (0,0,0), respectively. Thus, the Co-Morse Indes is the number of "up" directions.

f. Eigenvalues, Eigenvectors and Eigenfunctions

Eigenvalues are numbers. Eigenvectors are vectors. If $Ax=\lambda x$ for some $\lambda \in \mathbb{R}$, then x is an eigenvector of A corresponding to the eigenvalues λ. As and example, $-y''=\lambda y$ on [0,1], y(0)=0=y(1) is an eigenvalue (ODE) problem where $\lambda_k=(k\pi)^2$

is an eigenvalue with corresponding eigenfunction $y(x)=\text{Sin}k\pi x$.

g. What is λ?

Given an ODE $y''+f(\lambda,y)$ such as $y'' = -(\lambda y + y^3)$, the parameter λ can be varied to create a family of BVPs. One should realize that as you change λ, you change the the BVP and hence obtain different solutions.

h. Bifurcation Diagrams

On our Bifurcation Diagrams, the x-axis represents λ ($x=\lambda$) and the y-axis represents c ($y=c$). The parameter λ is varied to create a family of BVP. It is important to realize that as you change λ, you change the BVP solution and obtain different solutions. Each branch of the Bifurcation Diagram represents a differernt family of solutions corresponding to ordered pairs (λ,c). A bifurcation point is a value at which slight changes in the equation can cause a major differences in the system. Often, a bifurcation brings about a change in the stability, causing a stable situation to become unstable. A minor change in a parameter of the function resulst in major differences in the behavior of the system. For example, if we let $f(y)=-(\lambda y+y^3)$, then we obtain the following diagram.

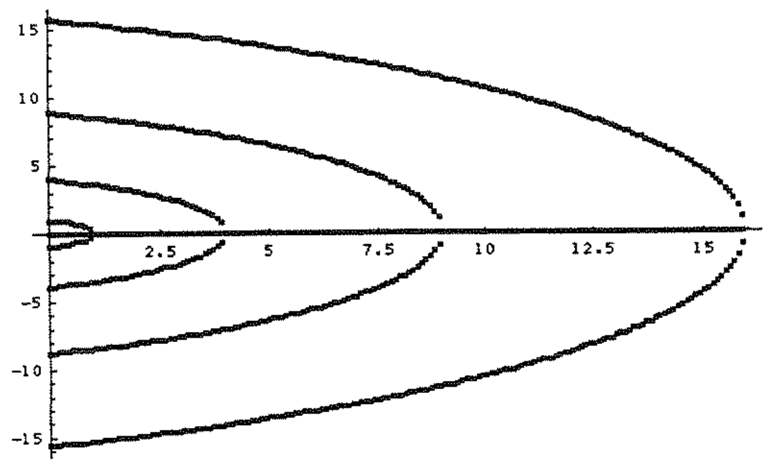


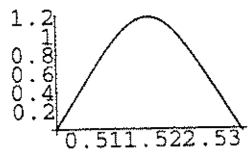
figure 1.1

```
n =70;

f[y_, λ_]:=-( λ*y+y^3);

a=0;b=Pi;dx = (b-a)/n;

y0 = 0;

deltac = .1 ;stopc=15.9999;

delta λ = .1; stop λ=15.9999;
```

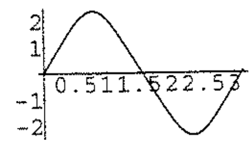
i. Graphs of Solutions of the BVP given by f(y)= -(λy+y^3)

These graphs do not represent what theseveral solutions look like, note the number of times the solutions changes signs. For example, at (0,1), where λ=0 and c=1, the solution changes sign once(in the positive direction), which gives up Morse Index (MI) 1 ,i.e., Saddle Point. At the point (0,4) our solution changes sign twice(once positive and once negative), and is of Morse Index 2, i.e., saddle point At the point, (0,9) it changes three times(twice positve, once negative), giving us Morse Index 1, i.e.,saddle point. At (0, 16), our result is four changes (two positive, two negative), implying we have Morse Index 2, i.e., a saddle point.

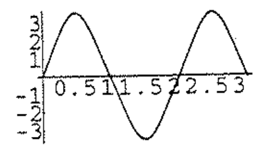
λ=0



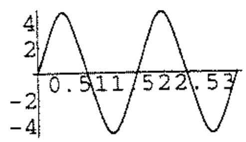
c=1



c=4



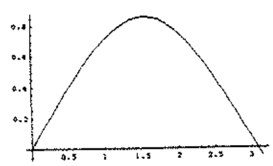
c=9



c=16

For λ=.5 we, only have one solution, where c=.8 to that problem It changes sign, once and is positive. It's MI , implying that there is a maximum.

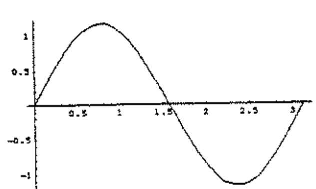
λ=.5



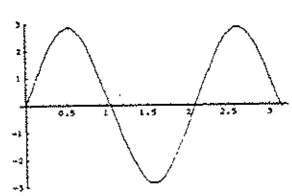
c=.8

For λ= 3 we have three, with MI 1, 1, 2 respectively.

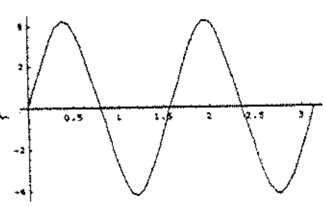
λ=3



c=2.25



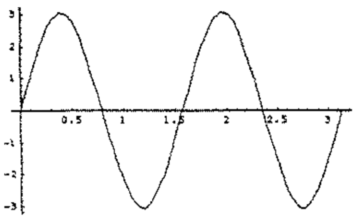
c=7.65



c=14.55

For λ=9, MI 2,

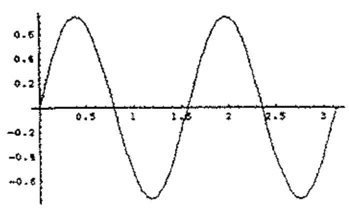
λ=9



c=11.3

MI 2

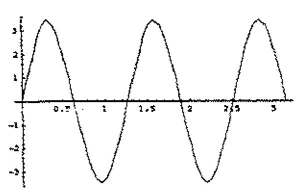
λ=15.5



c=2.95

MI 2,

λ=16



c=16

j. Difference Between IVP'and BVP's? Are they the same thing? Why and why not?

It is tempting to say that Initial Value Problems(IVP) and Boundary Value Problems(BVP) are one and the same,but they are not.

An example of an IVP is y" = f(y) (i.e., y"(x) = f(y(x)),, $\forall$ x$\in$ [0,π],

where y(0)=0

and y'(0)=c

It is a theorem that under certain condiotns on f, given I$\in$R, there exists a unique solution to this problem.

This is where we require the use of Euler's Method, i.e., Euler's method requires picking a starting point (x_0, y_0) (corresonding to the initial value), and calulating the slope at that point using the differential equation. We then proceed, by varying c by small increments, obtaining, a different solutions. This is exactly what has been previously stated, but in a more technical manner.

An example of a BVP for example is y" = f(y) (i.e., y"(x)=f(y(x)), $\forall$ x$\in$ [0,π],

where y(0)= 0

and y(π)= 0.

This is similar to the IVP example except that we are looking for solutions which satisfy y(0)=0 and y(π)=0 . It is correct to say that BVP solutions are solutions to some IVP, but not all IVP solutions will be solutions to our BVP. To be able to explain why this happens, I will refer back to my explaination of the shooting method. The Shooting Method is a way of using Euler's Method for solving IVPs to generate one or more solutions to a BVP. By varying y'(x_0)=c, there is a possiblity of finding IVP soultions which also satisfy the BVP. The basic concept is to vary y'(x_0), and check if the IVP solutions satisfy the BVP. Again, the analogy of the fixed target holds.

3. Results various f(y) attempted

f(y)=-(λ*y+y^7)

In this bifurcation diagram, we observe the same eigenvalues as when we looked at the funcation f(y)=-(λ*y+y^3). But in this graph, the branches are stretched out and occur more frequently then they did when f(y)=-(

λ*y+y^3) (figure 1.1).

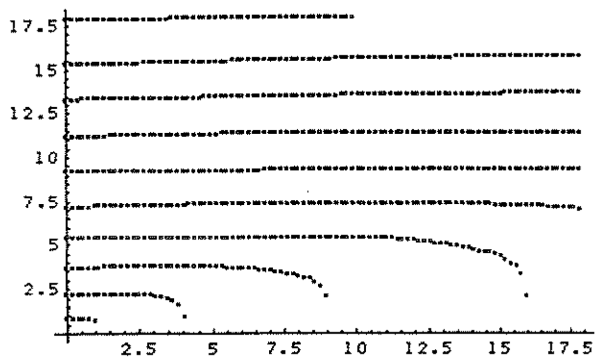


figure 2.1

n =85;

f[y_, λ_]:=-(λ*y+y^7);

a=0;b=Pi;dx = (b-a)/n;

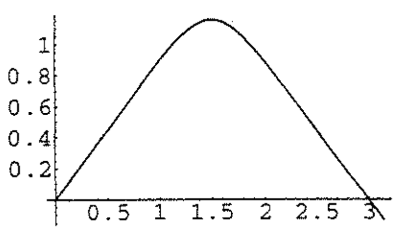
y0 = 0;

deltac = .1 ;stopc=18;

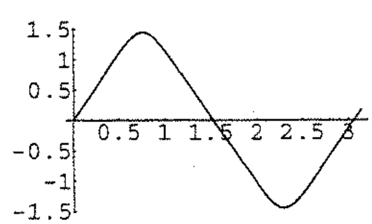
delta λ = .19; stop λ=18;

The solutions to the different branches are as follow:

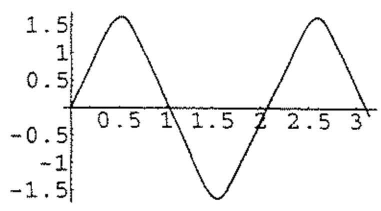
λ=0



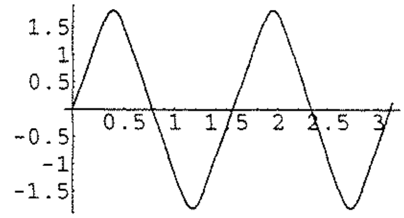
c=.9



c=2.2



c=3.7



c=5.2

f(y) = -λ*Exp[y]

In this graph, after many trial, we concluded that the solutons ranged from [0, .4]. I was particularly fasinated with this graph due the limited range of solutions. When we first began, evaluating this graph through *Mathematica* code, I was under the impression that there was not any solutions in the interval I had set for the code. Then I realized that there might be some solutions in a very limited interval that I had not looked at. So I decided to run this equations through the *Mathematica* code that solve IVP, this program yield some solutions, but what it did give me was a rough estimate of where the solutions were located.

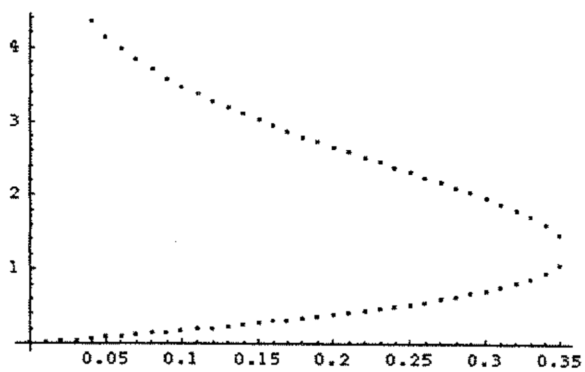
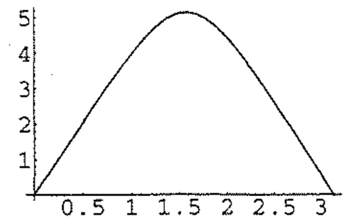


figure 3.1

```
n =100;

f[y_, lam_]:=-lam*Exp[y];

a=0;b=Pi;dx = (b-a)/n;

y0 = 0;

deltac = .01 ;stopc=4.8;
```

deltalam = .01; stoplam=.42;

Solutions to the IVP look like:



They never change sign, and they are all one hump.

f(y) = -(λy +Exp[-y]

In this graph we looked at the different branches that were created it doesn't seem to follow any of the pattern. none of the branches start and the eigenvalues. We found this graph particularly interesting because looks like the branches are convering to a asyntote.

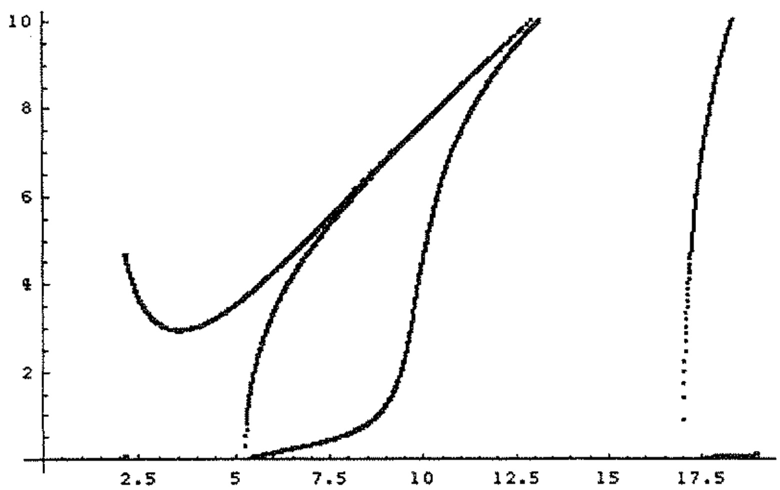


figure 4.1

```
n = 80;
f[y_, λ_] := -(λ*y + Exp[-y]);
a = 0; b = Pi; dx = (b - a)/n;
y0 = 0;
deltac = .01 ; stopc = 10;
deltaλ = .01; stopλ = 19;
```

This graph is a close up of figure 1.1. In this graph we wanted the it to go from [2, 3.2]. We wanted to see if indeed there was some solutions there and what they looked like.

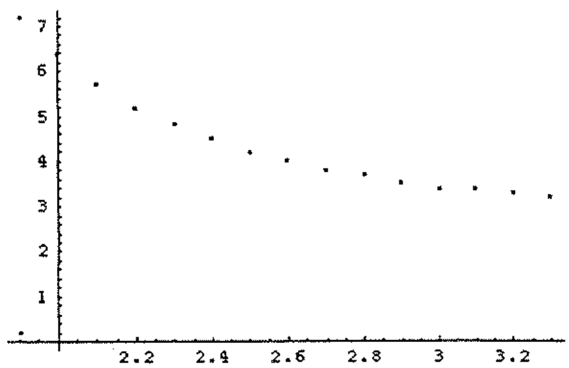


figure 4.2

```
n = 50;
f[y_, λ_] := -(λ * y + Exp[-y]);
a = 0; b = Pi; dx = (b - a) / n;
y0 = 0;
deltac = .1 ; stopc = 9;
deltaλ = .1; stopλ = 3.4;
```

Here we have yet another close up of the first except that in this bifurcation diagram we wanted it to go from [0,2]. We can see that there are some solutions here, but not as many as there are later in the graph of the 1.1 The interesting thing here, is that λ's range was from [0,2] and the only solutions graphed were from [0,1].

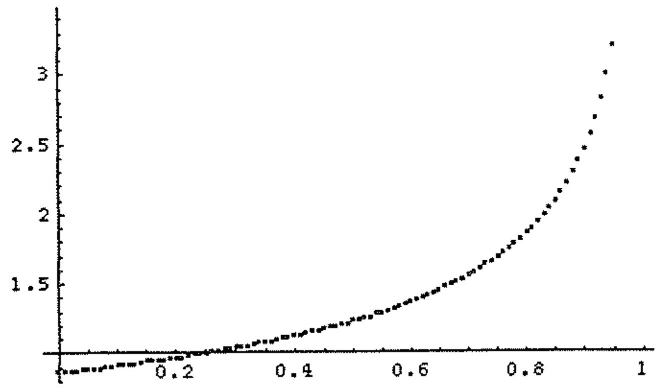


figure 4.3

```
n = 10;
f[y_, λ_] := -(λ * y + Exp[-y]);
a = 0; b = Pi; dx = (b - a) / n;
y0 = 0;
deltac = .01 ; stopc = 9;
deltaλ = .01; stopλ = 2;
alist = {}
For[λ = 0, λ <= stopλ, λ += deltaλ,
```

We did finally find solutions that ranged for [0,2], which concludes that the soultions are continious.

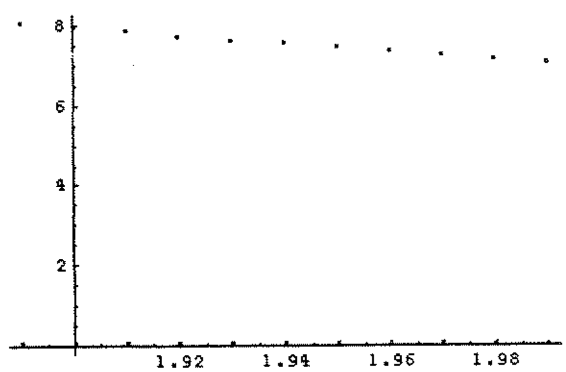


figure 4.4

```
n = 40;
f[y_, λ_] := -(λ * y + Exp[-y]);
a = 0; b = Pi; dx = (b - a)/n;
y0 = 0;
deltac = .01 ; stopc = 9;
deltaλ = .01; stopλ = 2;
```

The solutions to the BVP, have the usual one hump, two hump and three hump depending on the branch you are on.

f(y) = -λy + Exp[y]

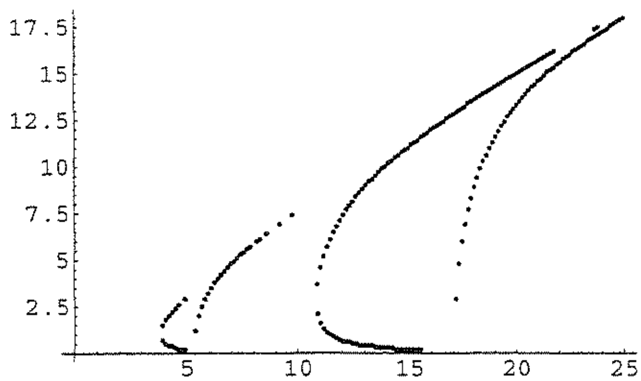
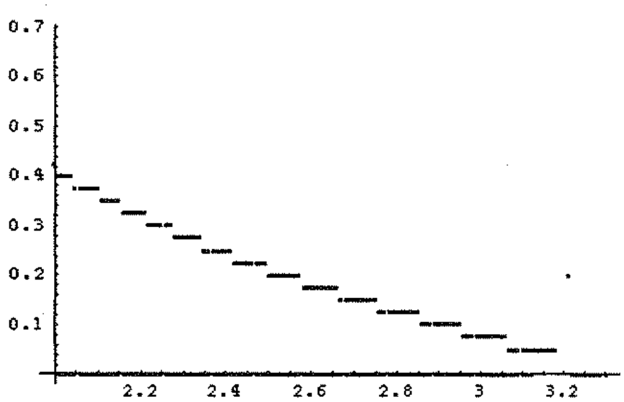


figure 6.1

```
n = 85;
f[y_, λ_] := -λ * y + Exp[y];
a = 0; b = Pi; dx = (b - a)/n;
y0 = 0;
deltac = .1 ; stopc = 25;
deltaλ = .145; stopλ = 25;
```



n =80;

f[y_, λ_]:=- λ*y + Exp[-y];

a=0;b=Pi;dx = (b-a)/n;

y0 = 0;

deltac = .025 ;stopc=2.5;

delta λ = .01; stop λ=4;

The solutions to the BVP are as usual one, two, three and so on.

Conclusion

I have included all the Bifurcaton diagrams and the Mathematica code I used during the the project. It is not to say that this project is complete, because true research is never complete. Many interesting queston arose during the REU program, that I was not able to explore, due to lack of time. While we did not make any breakthroughs mathematics, at a minimum I have gained a enormous amount of knowledge. I have personally learning about some relatively advance topics in Differential Equations, Analysis and Linear Algebra, amoung many other topic that I was introduced to for the first time and have learned the odds and ends of Mathematica. Although, we encountered many problems with code and equipment failure, I have learned how to cope with frustration and disappointment. At a minimum, I have gained a respectable proficiency in Mathematica programing, which is of great importance in the field of Mathematics. I don't claim that these are the only things that I have learned in eight weeks, to name them all I would require another eight weeks. Along with the mentioned, there are other things I learn that I hold particularly valuable to me.

I would also like to take this time to thank my advisor, for his countless patience and sharing of his knowledge. It is without saying that I would not have enjoyed the eight weeks to the extent I did without his constant talks and support.

4. Appendix A - *Mathematica* Code

This program will allow you to creates graphs of solutions for a BVP. If you wish to change have your diagrams range from positive to negative, that is y=[-c, c]; change your c to -stopc on line 11 and 23. This will create a diagram ranging from negative to positive c.

```
Clear[a,b,x,y,n,i,alist,y0,yp0,yim2,yim1,f,dx];
```

Code	**Meaning of code**
`Clear[deltac, stopc, c];`	
`n = ;`	***Number of iterations***
`f[y_, lam_]:=;`	***Function***
`a=0;b=Pi;dx = (b-a)/n;`	***Interval $[0,\pi]$, ∇x=change interval/iterations***
`y0 = 0;`	***f(x0)=y0, inital value***
`deltac =; stopc=;`	***∇c= increments of change in c, stopc= stopping pt***
`deltalam =; stoplam=;`	***$\nabla\lambda$= increments of change in λ***
`For[lam = 0, lam<= stoplam, lam+= deltalam,`	***For loop, for λ***
`Print[ "lam = ", lam];`	***Print λ***
`For [c = deltac, c<=stopc, c += deltac,`	***For loop, in loop, for c***
`yp0 = c;`	***y'0=c, slope at y0=c***
`yim2 = y0 // N; yim1 = y0 + dx*yp0 // N;`	***yi minus2=y0 divided by N; yiminus1=y0+Δx*y'0, convert to a number***
`Clear[alist];`	***Clear alist when you change λ***
`alist = {{a,y0},{a+dx,yim1}};`	***alist conatins "solution points"***
`For[i=2,i<=n,i++,`	***For loop,for i=increment, initial= 2 and increment by 1***
`x = a + i*dx;`	***x=previous point+increment* ∇x***
`y = (dx^2 )*f[yim1, lam] + 2*yim1 - yim2 // N;`	***y=$(\nabla x)^2$*f(yi-1,λ)+2(yi-1)-yi-2, convert to a number***
`alist = Append[alist,{x,y}];`	***append alist with (a,y0 , a+∇x,yiminus1)=x coordinate, &(x,y) as y coordinate***
`temp = yim1; yim1 = y; yim2 = temp;`	***yi-1=yi-2=y=temp***
`];`	***Close for loop for i***
`ysign = Sign[y];`	***sign of y(pos or neg)=to temp storage Signy***

```
    If[c == deltac, test = ysign, Null];          is c exactly ∇c, test if the sign of y changes

    If[test==ysign,Null;,                         is test = to ysign?
      ListPlot[alist,PlotJoined->True];           if the statement is true plot the point
      test = ysign;                               if test is equal to ysign
      Print["c=",c,"    and  y(",b,")=",y];       print c and y
    ];                                            close if statement
  ];                                              close for loop c
];                                                close for loop λ
```

This program creates Bifurcation Diagrams for BVPs. If you wish to change have your diagrams range from positive to negative, that is y'(0)[-c, c]; change your c to -stopc on line 11 and 23. This will create a diagram ranging from negative to positive c. For this program I will not explain every line, I will only explain the lines that are new/not included in the previous program.

```
Clear[a,b,x,y,n,i,alist,y0,yp0,yim2,yim1,f,dx];
Clear[deltac, stopc, c];

n =;
f[y_, lam_]:=;
a=0;b=Pi;dx = (b-a)/n;
y0 = 0;
deltac =  ;stopc=;
deltalam = ; stoplam=;
alist = { };                                   create alist

For[lam = 0, lam<= stoplam, lam+= deltalam,


  For [c = deltac, c<=stopc, c += deltac,

    yp0 = c;

    yim2 = y0 // N; yim1 = y0 + dx*yp0 // N;

    For[i=2,i<=n,i++,

      x = a + i*dx;
      y = (dx^2 )*f[yim1, lam] + 2*yim1 - yim2 // N;

      temp = yim1; yim1 = y; yim2 = temp;
    ];

    ysign = Sign[y];

    If[c == deltac, test = ysign, Null];
```

```
    If[test==ysign,Null;,

      test = ysign;
      Print["c=",c,"     and  lam = ",lam];
      alist = Append[alist, {lam, c}];
```

Add value of alist to equal x coordinate, and value of{ λ,c} to equal y coordinate to alist

```
    ];
  ];
];
ListPlot[alist];
```

Plot on same coordinate plane all the values in alist

This program gives you solutions to a IVP. In this program you don't vary lamda(λ). It's a good program if you wish to find the solutions on one set λ.

```
Clear[a,b,x,y,n,i,alist,y0,yp0,yim2,yim1,f,dx];
Clear[deltac, stopc, c];
n =;

lambda = ;
f[y_]:=;
a=0;b=Pi;dx = (b-a)/n;
y0 = 0;
deltac = ;stopc=;
test = 1;
```

test for ysign to be positive

```
For [c = deltac, c<=stopc, c += deltac,
```

c incremented by ∇c

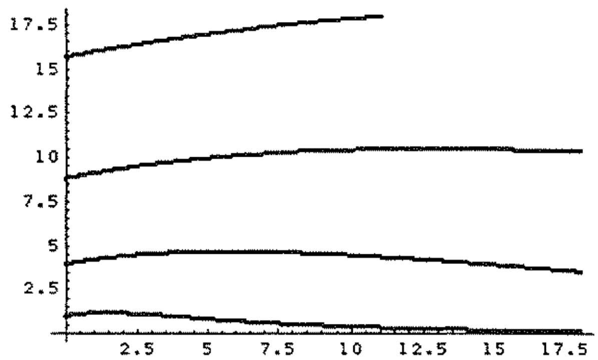
```
  yp0 = c;

  yim2 = y0 // N; yim1 = y0 + dx*yp0 // N;

  Clear[alist];
  alist = {{a,y0},{a+dx,yim1}};

  For[i=2,i<=n,i++,

    x = a + i*dx;
    y = (dx^2 )*f[yim1] + 2*yim1 - yim2 // N;

    alist = Append[alist,{x,y}];
    temp = yim1; yim1 = y; yim2 = temp;
  ];

  ysign = Sign[y];
  If[test==ysign,Null;,
    ListPlot[alist,PlotJoined->True];
    test = ysign;
```

```
  Print["c=",c,"     and  y(",b,")=",y];
 ];
];
```

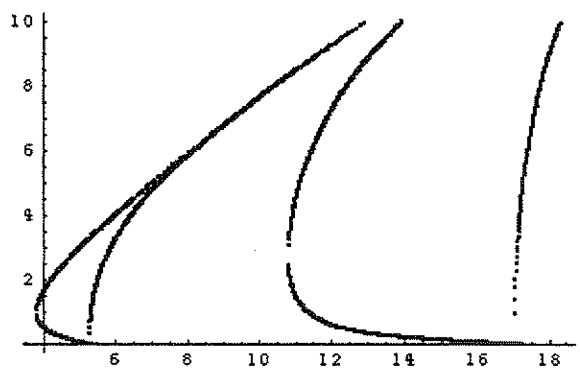
This program solves equations without any parameter λ, or varying c. It solves one at a time.

```
Clear[a,b,y,x,y0,ym1,ym2,yp0,f,dx,dy,i,n,alist];
f[y_]:=-y;
a=0;                              left endpoint
b=Pi;                             right endpoint
n=10;                             number of divisions
dx=(b-a)/n;                       Δx
y0 = 0;                           initial value of y at a
ym2 = y0;                         y-2= y0
yp0 = 1;                          y'o = 1
ym1  = yp0 * dx;                  y-1 = y'0*Δx
alist = {{ a,ym2},{a+dx,ym1}};
For[i=2,i<=n,i++,
    y = f[ym1]*(dx)^2+2*ym1 - ym2;
              x = a + dx*i;
    alist = Append[alist,{x,y}];
    ym1 = y;
    ym2 = ym1;
    ];

Print["y(",b,")=",y // N];
ListPlot[alist,PlotJoined->True];
```

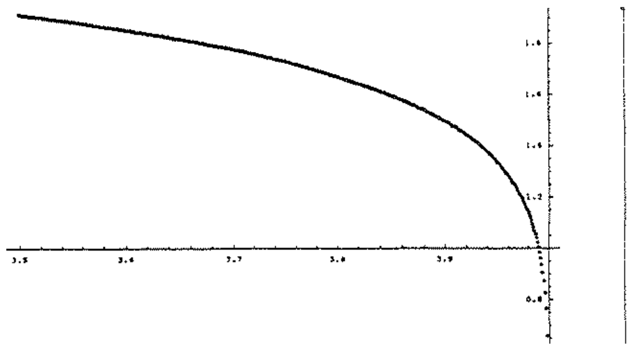
5. Appendix B



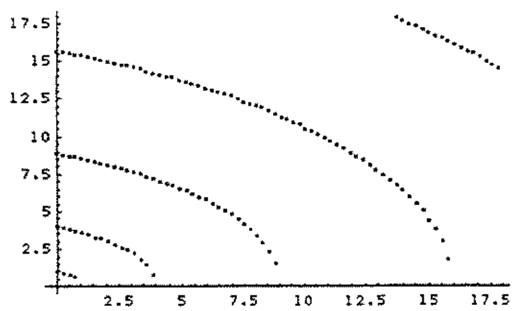

```
n = 60;
f[y_, λ_] := -(- λ * y + y^3);
a = 0; b = Pi; dx = (b - a) / n;
y0 = 0;
deltac = .01 ; stopc = 18;
delta λ = .1; stop λ = 18;
```



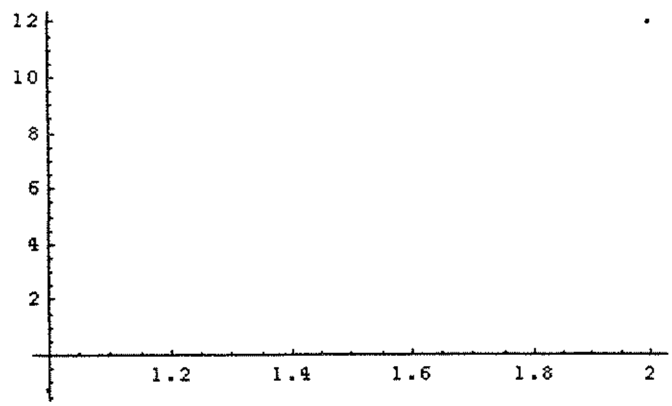

```
n = 80;
f[y_, λ_] := - λ * y + Exp[y];
a = 0; b = Pi; dx = (b - a) / n;
y0 = 0;
deltac = .01 ; stopc = 10;
delta λ = .01; stop λ = 19;
```



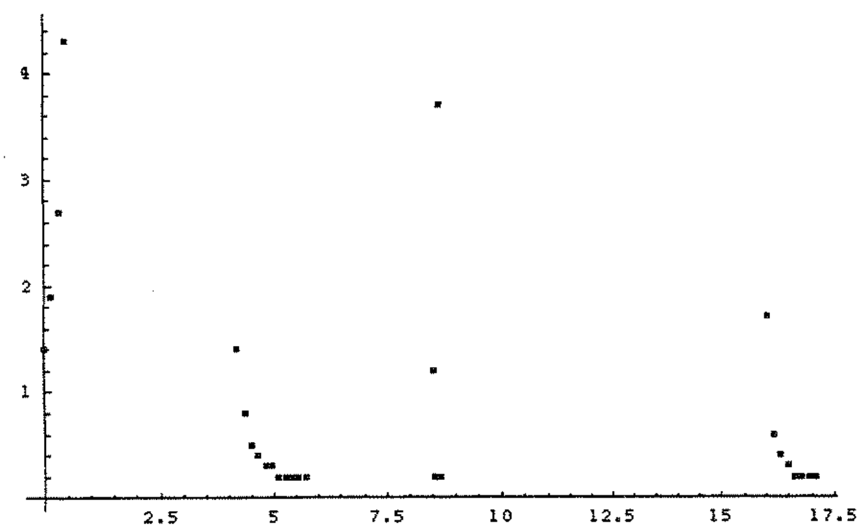
```
n =100;
f[y_, λ_]:=-( λ*y+y^7);
a=0;b=Pi;dx = (b-a)/n;
y0 = 0;
deltac = .001 ;stopc=2.5;
delta λ = .001; stop λ=4.5;
```



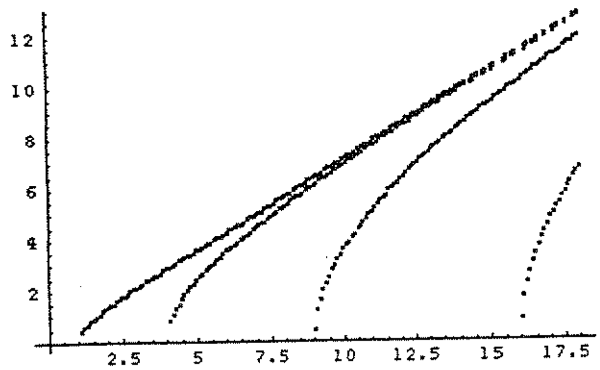

```
n =60;
f[y_, λ_]:=-( λ*y+y^3);
a=0;b=Pi;dx = (b-a)/n;
y0 = 0;
deltac = .1 ;stopc=18;
delta λ = .2589; stop λ=18;
```




```
n = 75;
f[y_, λ_] := - λ * y + Exp[y];
a = 0; b = Pi; dx = (b - a)/n;
y0 = 0;
deltac = .01 ; stopc = 5;
delta λ = .01; stop λ = 14;
```




```
n = 80;
f[y_, λ_] := -( λ * y + Abs[y]^.5);
a = 0; b = Pi; dx = (b - a)/n;
y0 = 0;
deltac = .1 ; stopc = 18;
delta λ = .15; stop λ = 18;
```




```
n =85;
f[y_, λ_]:=(- λ*y+y^3);
a=0;b=Pi;dx = (b-a)/n;
y0 = 0;
deltac = .1 ;stopc=18;
delta λ = .1; stop λ=18;
```

References

Agnew, Ralph P. (1960). Differential Equations. McGraw-Hill Book Company, Inc, The Maple Press Company. York, PA. .332-340.

McCallum, William G. (1996). Multivariable Calculus. Preliminary ed. John Wiley &Sons, Inc. New York, NY. 334.

Neuberger, John M. (1995). "Existence of a Sign-Changing Solution to a Superlinear Dirichlet Problem." and (1997) "A numerical method for finding sign-changing solutions of superlinear Dirichlet problems."

Scheid, Francis. (1968). Theory and Problems of Numerical Analysis. McGraw-Hill Book Company, Inc. New York, NY.

Spiegel, Murray R. (1981), Applied Differential Equations. 3rd ed. Prentice-Hall Inc. Englewood Cliffs, NJ.