Multiple Shooting

Monique St-Maurice
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/gradreports

Part of the Mathematics Commons, and the Statistics and Probability Commons

Recommended Citation
St-Maurice, Monique, "Multiple Shooting" (1985). All Graduate Plan B and other Reports. 1252.
https://digitalcommons.usu.edu/gradreports/1252

This Report is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
MULTIPLE SHOOTING

by

Monique St-Maurice

A report submitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Mathematics (Plan B)

UTAH STATE UNIVERSITY
Logan, Utah

1985
TABLE OF CONTENTS

LIST OF TABLES.....................................................................................iii
ABSTRACT..............................................................................................iv

Chapter

I. MULTIPLE SHOOTING METHOD.......................................................1
   1. INTRODUCTION..............................................................................1
   2. METHOD.......................................................................................2

II. EXAMPLE..........................................................................................7
   1. CASE 1: m=11, s\(0\)=0.............................................................7
   2. CASE 2: m=5, s\(0\)=0...............................................................11
   3. CASE 3: m=11, s\#0.................................................................13

REFERENCE............................................................................................15
APPENDIXES.........................................................................................16
   1. APPENDIX A: COMPUTER PROGRAM........................................17
   2. APPENDIX B: BIBLIOGRAPHY...................................................22
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Results for $m=11$, after 9 iterations</td>
<td>8</td>
</tr>
<tr>
<td>2. Results for $m=11$, after 1 iteration</td>
<td>9</td>
</tr>
<tr>
<td>3. Results for $m=11$, after 5 iterations</td>
<td>10</td>
</tr>
<tr>
<td>4. Results for $m=5$, after 16 iterations</td>
<td>12</td>
</tr>
<tr>
<td>5. Results for $m=5$, after 9 iterations</td>
<td>12</td>
</tr>
<tr>
<td>6. Results for $m=11$ and $s^0 \neq 0$, after 9 iterations</td>
<td>14</td>
</tr>
</tbody>
</table>
ABSTRACT

Multiple Shooting

by

Monique St. Maurice, Master of Science

Utah State University, 1985

Major Professor: Dr. Chris. Coray
Department: Mathematics

The purpose of this report was to study the Multiple Shooting method, a numerical method to solve boundary value ordinary differential equation. A FORTRAN program was written to solve the specific problem

\[ y'' = -y, \quad y(0) = 0, \quad y(\pi/2) = 1 \]

on a microcomputer, using the microsoft FORTRAN compiler and the 8087 coprocessor.

(24 pages)
CHAPTER I
MULTIPLE SHOOTING METHOD

1. INTRODUCTION

The subject of this report is "Multiple Shooting", a numerical method to solve boundary value ordinary differential equations. We will only consider the case where the boundary conditions are separated.

The multiple shooting method, as described by Stoer and Bulirsch [1], is the central reference in this work.

The first part of this report is a description of the multiple shooting method. The second part contains the results obtained when running the program on the example

\[ y'' = -y, \quad y(0) = 0 \quad \text{and} \quad y(\pi/2) = 1. \]

Finally, two appendixes conclude the report. Appendix A is the FORTRAN program, and Appendix B is a bibliographical list of publications on the subject of multiple shooting.
2. METHOD

The multiple shooting method is a procedure for solving a boundary value problem
\[ y' = F(x, y). \]

One seeks a solution \( y(x) \) satisfying a boundary condition of the form
\[ Ay(a) + By(b) = c. \]

In practice the boundary conditions are usually separated:
\[ Ay(a) = c, \quad By(b) = d. \]

Here, \( a, b \) are given numbers, \( A, B \) square matrices of order \( n \), and \( c, d \) vectors in \( \mathbb{R}^n \).

In what followed, \( y \) and \( s_k \) are vectors of dimension \( m \), \( x_k \) are real numbers.

The basic idea of this method is to guess some initial values
\[ y(x_k) = s_k, \quad k = 1, 2, \ldots, m \]
at several predetermined points
\[ a = x_1 < x_2 < \ldots < x_m = b \]
and, then solve \( m-1 \) initial value problems. We do not expect that this solution will work, but we can adjust the guesses by iteration using Newton's method, so that we hopefully converge to the exact solution \( y(x) \).
We will now see how this procedure is carried out. Let us choose a partition of \([a, b]\) with \(a = x_1 < x_2 < \ldots < x_m = b\). To each point \(x_k\), we associate a guess \(s_k\), such that

\[ s_k = y(x_k), k=1,2,\ldots,m. \]

We let \(y(x_{k+1};x_k, s_k)\) be the solution of the initial value problem

\[ y' = F(x,y), \quad y(x_k) = s_k, \quad \text{at the point } x_{k+1}. \]

We would like to have

\[ y(x_{k+1};x_k, s_k) - s_{k+1} = 0, \quad k=1,2,\ldots,m-1, \]

and also

\[ r(s_1, s_m) = 0, \]

with

\[ r(y(a), y(b)) = 0 \]

being the boundary conditions. For notation, let

\[ F_k(s_k, s_{k+1}) = y(x_{k+1};x_k, s_k) - s_{k+1}. \]

We can represent all the conditions mentioned above with a system of equations of the form

\[
\begin{align*}
F_1(s_1, s_2) &= y(x_2; x_1, s_1) - s_2 \\
F_2(s_2, s_3) &= y(x_3; x_2, s_2) - s_3 \\
\vdots &= \vdots \\
F(s) &= \\
F_m(s_{m-1}, s_m) &= y(x_m; x_{m-1}, s_{m-1}) - s_m \\
F_m(s_1, s_m) &= r(s_1, s_m)
\end{align*}
\]

\(1\)
By solving $m-1$ initial-value problems, we will compute $F(s)$. We want to use the Newton's formula to readjust our guesses iteratively, i.e.,

$$s_i^{i+1} = s_i^i - [DF(s_i^i)]^{-1} F(s_i^i), \quad i=0,1,...$$ (2)

We have already computed $F(s)$, but we still have to compute $DF(s)$. This is done as follows: the Jacobian matrix $DF(s)$ has the form

$$DF(s) = \begin{bmatrix}
G_1 & -I & 0 & 0 \\
0 & G_2 & -I & 0 \\
0 & 0 & G_3 & -I \\
0 & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots \\
A & 0 & 0 & B
\end{bmatrix}$$ (3)

where the matrices $A, B, G_k, k=1,2,...,m-1$, are Jacobian matrices of order $n \times n$. 
We use difference quotients in lieu of actual derivatives in the above matrices. These can be computed by solving an additional \((m-1)n\) initial-value problems. We compute \(s^{i+1}\) from \(s^i\) as follows:

If we let

\[
\begin{bmatrix}
\Delta s_1 \\
\vdots \\
\Delta s_m
\end{bmatrix}
\]

from equation (2), we get

\[
DF(s) \begin{bmatrix}
\Delta s_1 \\
\vdots \\
\Delta s_m
\end{bmatrix} = -F(s)
\]

We can write equation (5) as the following system of linear equations:

\[
\begin{align*}
G_1 \Delta s_1 - \Delta s_2 &= -F_1 \\
G_2 \Delta s_2 - \Delta s_3 &= -F_2 \\
&\vdots \\
G_{m-1} \Delta s_{m-1} - \Delta s_m &= -F_{m-1} \\
A \Delta s_1 + B \Delta s_m &= -F_m
\end{align*}
\]
Beginning with the first equation, we can express all $\Delta s_k$ successively in terms of $\Delta s_1$.

\[ \Delta s_2 = G_1 \Delta s_1 + F_1 \]
\[ . \]
\[ . \]
\[ . \]
\[ \Delta s_m = G_{m-1} G_{m-2} \ldots G_1 \Delta s_1 + \sum_{j=1}^{m-1} \left( \prod_{l=j+1}^{m} G_l \right) F_j \]

From the last equation we get,

\[ (A + BG_{m-1} G_{m-2} \ldots G_1) \Delta s_1 = w \]  
(8)

where

\[ w = -(F_m + B \sum_{m-1} \ldots + B G_{m-1} G_{m-2} \ldots G_2 F_1). \]

This is a system of linear equations in the unknowns $\Delta s_1$. We can solve this by Gaussian elimination, and then substitute $\Delta s_1$ in the first equation of the system (7) to get $\Delta s_2$ and again in the successive equations to get all the $\Delta s_k$. Knowing all $\Delta s_k$, we may obtain the new iterates $s_{i+1}$ from (4), and compute a new approximate solution $y$. 
CHAPTER II

EXAMPLE

1. CASE 1: m=11, s^0=0

We considered the two point boundary value problem
\[ y'' = -y, \quad y(0) = 0, \quad y(\pi/2) = 1 \]

A FORTRAN program was written to solve this problem by multiple shooting. A Runge-Kutta subroutine in vector form was used to solve all the initial value problems. The Newton's iteration was started with \( s_k = 0 \), for \( k = 1, m \). The program was executed on a Leading Edge microcomputer using the Microsoft FORTRAN compiler and the 8087 coprocessor. The Newton's iteration was terminated when

\[ F_k(s_k, s_{k+1}) < 10^{-7} \]

for \( k = 1, m \).

\[ F_k(s_k, s_{k+1}) = y(x_{k+1}; x_k, s_k) - s_{k+1} \]

The program was first executed with \( m = 11 \); we chose 11 equally spaced points in the interval \([0, \pi/2]\). The solution was obtained in 9 iterations to an accuracy of 6 digits on both \( y \) and \( y' \). Since the analytic solution was known, we also computed a relative error on \( y \). The total computing time for this case was 16 seconds. The results are tabulated below.
### TABLE 1

Results for m=11, after 9 iterations

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Y'</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00000E+00</td>
<td>.00000E+00</td>
<td>.10000E+01</td>
<td>.00000E+00</td>
</tr>
<tr>
<td>.15708E+00</td>
<td>.15643E+00</td>
<td>.98769E+00</td>
<td>.19051E-06</td>
</tr>
<tr>
<td>.31416E+00</td>
<td>.30902E+00</td>
<td>.95106E+00</td>
<td>.96442E-07</td>
</tr>
<tr>
<td>.47124E+00</td>
<td>.45399E+00</td>
<td>.89101E+00</td>
<td>.65645E-07</td>
</tr>
<tr>
<td>.62832E+00</td>
<td>.58779E+00</td>
<td>.80902E+00</td>
<td>.10141E-06</td>
</tr>
<tr>
<td>.78540E+00</td>
<td>.70711E+00</td>
<td>.70711E+00</td>
<td>.16859E-06</td>
</tr>
<tr>
<td>.94248E+00</td>
<td>.80902E+00</td>
<td>.58779E+00</td>
<td>.29470E-06</td>
</tr>
<tr>
<td>.10996E+01</td>
<td>.89101E+00</td>
<td>.45399E+00</td>
<td>.20069E-06</td>
</tr>
<tr>
<td>.12566E+01</td>
<td>.95106E+00</td>
<td>.30902E+00</td>
<td>.25069E-06</td>
</tr>
<tr>
<td>.14137E+01</td>
<td>.98769E+00</td>
<td>.15643E+00</td>
<td>.30174E-06</td>
</tr>
<tr>
<td>.15708E+01</td>
<td>.10000E+01</td>
<td>.78842E-07</td>
<td>.17881E-06</td>
</tr>
</tbody>
</table>
The following two tables show the results after only 1 iteration (table 2), and after 5 iterations (table 3) for the case m=11.

**TABLE 2**

Results for m=11, after 1 iteration

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Y'</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00000E+00</td>
<td>.00000E+00</td>
<td>.10000E+01</td>
<td>.00000E+00</td>
</tr>
<tr>
<td>.15708E+00</td>
<td>.15643E+00</td>
<td>.98769E+00</td>
<td>.19051E-06</td>
</tr>
<tr>
<td>.31416E+00</td>
<td>.30519E+00</td>
<td>.92689E+00</td>
<td>.12544E-01</td>
</tr>
<tr>
<td>.47124E+00</td>
<td>.44279E+00</td>
<td>.84475E+00</td>
<td>.25300E-01</td>
</tr>
<tr>
<td>.62832E+00</td>
<td>.56612E+00</td>
<td>.74382E+00</td>
<td>.38275E-01</td>
</tr>
<tr>
<td>.78540E+00</td>
<td>.67249E+00</td>
<td>.62705E+00</td>
<td>.51478E-01</td>
</tr>
<tr>
<td>.94248E+00</td>
<td>.75970E+00</td>
<td>.49769E+00</td>
<td>.64920E-01</td>
</tr>
<tr>
<td>.10996E+01</td>
<td>.82607E+00</td>
<td>.35925E+00</td>
<td>.78613E-01</td>
</tr>
<tr>
<td>.12566E+01</td>
<td>.87047E+00</td>
<td>.21534E+00</td>
<td>.92577E-01</td>
</tr>
<tr>
<td>.14137E+01</td>
<td>.89235E+00</td>
<td>.69661E-01</td>
<td>.10683E+00</td>
</tr>
<tr>
<td>.15708E+01</td>
<td>.89173E+00</td>
<td>-.74173E-01</td>
<td>.12142E+00</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Y'</td>
<td>Error</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>.0000E+00</td>
<td>.0000E+00</td>
<td>.99995E+00</td>
<td>.0000E+00</td>
</tr>
<tr>
<td>.15708E+00</td>
<td>.15643E+00</td>
<td>.98764E+00</td>
<td>.52679E-04</td>
</tr>
<tr>
<td>.31416E+00</td>
<td>.30900E+00</td>
<td>.95100E+00</td>
<td>.55747E-04</td>
</tr>
<tr>
<td>.47124E+00</td>
<td>.45396E+00</td>
<td>.89095E+00</td>
<td>.58822E-04</td>
</tr>
<tr>
<td>.62832E+00</td>
<td>.58775E+00</td>
<td>.80896E+00</td>
<td>.62470E-04</td>
</tr>
<tr>
<td>.78540E+00</td>
<td>.70706E+00</td>
<td>.70705E+00</td>
<td>.66344E-04</td>
</tr>
<tr>
<td>.94248E+00</td>
<td>.80896E+00</td>
<td>.58772E+00</td>
<td>.70733E-04</td>
</tr>
<tr>
<td>1.0996E+01</td>
<td>.89094E+00</td>
<td>.45393E+00</td>
<td>.75598E-04</td>
</tr>
<tr>
<td>1.2566E+01</td>
<td>.95098E+00</td>
<td>.30896E+00</td>
<td>.81104E-04</td>
</tr>
<tr>
<td>1.4137E+01</td>
<td>.98760E+00</td>
<td>.15638E+00</td>
<td>.87331E-04</td>
</tr>
<tr>
<td>1.5708E+01</td>
<td>.99991E+00</td>
<td>-.44635E-04</td>
<td>.94423E-04</td>
</tr>
</tbody>
</table>
2. CASE 2: m=5, \( s^0 = 0 \)

The order of the differential equation we are trying to solve determines the dimension \( n \) of the vectors \( s_k \), and also the dimension of the matrices \( A, \), \( B, \) and \( G_k \). In our example \( n = 2 \). \( A, B \) and \( G \) are all 2x2 matrices.

The number of points we choose in the interval \([a, b]\) determines the number of initial-value problems we need to solve to compute \( F(s) \), and also the number of matrices \( G_k \) we need to compute to form the big Jacobian Matrix \( DF(s) \). In our example, \( m = 11 \). In addition to this, each small \( n \times n \) matrix \( G_k \) requires an additional \( n \times (m-1) \) initial-value problems to be solved. In our example, we had to solve 20 initial-value problems to obtain \( DF(s) \).

It is easy to see that the amount of computation increases rapidly when \( m \) and \( n \) get larger. Then it seems like a good idea to try to reduce the number of points we choose in the interval \([0, \pi/2]\). We ran the program with 5 points instead of 11 and obtained a solution to an accuracy of 6 digits on both \( y \) and \( y' \) in 16 iterations. The results are tabulated below.
TABLE 4
Results for m=5, after 16 iterations

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Y'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>1.0000E+01</td>
</tr>
<tr>
<td>0.3927E+00</td>
<td>0.38268E+00</td>
<td>0.92388E+00</td>
</tr>
<tr>
<td>0.78540E+00</td>
<td>0.70711E+00</td>
<td>0.70711E+00</td>
</tr>
<tr>
<td>0.11781E+01</td>
<td>0.92388E+00</td>
<td>0.38268E+00</td>
</tr>
<tr>
<td>0.15708E+01</td>
<td>0.10000E+01</td>
<td>-.29253E-07</td>
</tr>
</tbody>
</table>

We also include the results for the case m=5, after 9 iterations, so that comparison with the case m=11 can be made.

TABLE 5
Results for m=5, after 9 iterations

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Y'</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>1.0000E+01</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>0.3927E+00</td>
<td>0.38267E+00</td>
<td>0.92385E+00</td>
<td>0.37320E-04</td>
</tr>
<tr>
<td>0.78540E+00</td>
<td>0.70708E+00</td>
<td>0.70708E+00</td>
<td>0.39710E-04</td>
</tr>
<tr>
<td>0.11781E+01</td>
<td>0.92385E+00</td>
<td>0.38266E+00</td>
<td>0.33108E-04</td>
</tr>
<tr>
<td>0.15708E+01</td>
<td>0.99996E+00</td>
<td>-.17124E-04</td>
<td>0.40002E-04</td>
</tr>
</tbody>
</table>
3. CASE 3: m=11, s^0≠0

We were interested in finding out if a different guess to start the Newton's iteration would give different results. We ran the program for the case m=11, but with different initial guesses. We chose the starting vectors as follows:

\[
\begin{align*}
  s_1 &= (0,1) \\
  s_2 &= (.1,.9) \\
  s_3 &= (.2,.8) \\
  s_4 &= (.3,.7) \\
  s_5 &= (.4,.6) \\
  s_6 &= (.5,.5) \\
  s_7 &= (.6,.4) \\
  s_8 &= (.7,.3) \\
  s_9 &= (.8,.2) \\
  s_{10} &= (.9,.1) \\
  s_{11} &= (1,0)
\end{align*}
\]

It was surprising to obtain the solution in exactly the same number of iterations as for the other starting vectors since those guesses are much closer to the exact solution than the first ones. That last result is tabulated in table 6.
TABLE 6
Results for $m=11$ and $s^0 \neq 0$, after 9 iterations

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Y'$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00000E+00</td>
<td>.00000E+00</td>
<td>.10000E+01</td>
<td>.00000E+00</td>
</tr>
<tr>
<td>.15708E+00</td>
<td>.15643E+00</td>
<td>.98769E+00</td>
<td>.19051E-06</td>
</tr>
<tr>
<td>.31416E+00</td>
<td>.30902E+00</td>
<td>.95106E+00</td>
<td>.96442E-07</td>
</tr>
<tr>
<td>.47124E+00</td>
<td>.45399E+00</td>
<td>.89101E+00</td>
<td>.65645E-07</td>
</tr>
<tr>
<td>.62832E+00</td>
<td>.58779E+00</td>
<td>.80902E+00</td>
<td>.10141E-06</td>
</tr>
<tr>
<td>.78540E+00</td>
<td>.70711E+00</td>
<td>.70711E+00</td>
<td>.84294E-07</td>
</tr>
<tr>
<td>.94248E+00</td>
<td>.80902E+00</td>
<td>.58779E+00</td>
<td>.22103E-06</td>
</tr>
<tr>
<td>.10996E+01</td>
<td>.89101E+00</td>
<td>.45399E+00</td>
<td>.13379E-06</td>
</tr>
<tr>
<td>.12566E+01</td>
<td>.95106E+00</td>
<td>.30902E+00</td>
<td>.18802E-06</td>
</tr>
<tr>
<td>.14137E+01</td>
<td>.98769E+00</td>
<td>.15643E+00</td>
<td>.60348E-07</td>
</tr>
<tr>
<td>.15708E+01</td>
<td>.10000E+01</td>
<td>.97182E-07</td>
<td>.00000E+00</td>
</tr>
</tbody>
</table>
REFERENCE

APPENDIXES
1. APPENDIX A: COMPUTER PROGRAM

This is a FORTRAN program to solve a second order differential equation by multiple shooting.

```fortran
REAL LINS(2,2),numy,numyp
DIMENSION GUESS(2,11),STORE(2,11),F(2,11),DF(25,25),A(2,2),
> B(2,2),W(2),GG(2,2),AB(2,2),WW(2,2),V(2),Delta(2),
> STAR(2),GF(2),G(2,2)
OPEN (2,file='mltshoo.out',form='formatted',status=
> 'new',access='sequential')

Input.

Xint=0
Xfin=3.14159265/2.
BigH=3.14159265/20.

We are working with a first guess to start Newton method.

Computing F(s) by solving 10 initial values problems by RungeKutta.

Do 100 IBRO=1,100
  xint=0
  Do 5 I=1,10
    x=xint
    Gl=GUESS(1,I)
    G2=GUESS(2,I)
    Call RKUTTA(X,Gl ,G2,NUMY,NUMYP)
    Store(1,I)=NUMY
    Store(2,I)=NUMYP
    F(1,I)=Store(1,I)-GUESS(1,I+1)
    F(2,I)=Store(2,I)-Guess(2,I+1)
  5 Continue

Convergence test

If (IBRO.LT.2) go to 200
Do 80 I=1,10
  If (F(1,I).GT.(1.E-7).or.F(2,I).GT.(1.E-7) go to 200
80 Continue
```
Write (2,1600) IBRO
1600 Format (15x,'Iteration #',I3)
Write (2,2000)
Write (2,1200)
Write (2,4000)
1200 Format(12x,'X',14x,'Y',14x,'YP',14x,'Error')
4000 Format (/)
2000 Format (/,/)

xint=0
Error=0
Write (2,1000) xint,Guess(1,1),Guess(2,1),Error
Do 11 I=1,10
xint=xint+bigh
Error=ABS((sin(xint)-store(1,I))/store(1,I))
Write (2,1000) Xint,store(1,I),store(2,I),Error
11 Continue
1000 Format (5x,4E15.5)
Go to 300

c We start computing the new guesses.

200 F(1,11)=Guess(1,1)
F(2,11)=Guess(1,11)-1.

c Computing the Jacobian matrix DF(s)
c First we compute the matrices Gk,k=1,m-1
c We need to solve 2 more initial values problems for each matrix G
c
Xint=0
SMDELTA=0.001
Do 10 I=1,10
x=xint
ALPHA=GUESS(1,I)+SMDELTA
G1=Guess(1,I)
G2=Guess(2,I)
BETA=GUESS(2,I)+SMDELTA
CALL RKUTTA(X,ALPHA,G2,NUMY,NUMYP)
DF(2*I-1,2*I-1)=(NUMY-Store(1,I))/SMDELTA
DF(2*I,2*I-1)= (NUMYP-Store(2,I))/SMDELTA
CALL RKUTTA(X,G1,BETA,NUMY,NUMYP)
DF(2*I-1,2*I)=(NUMY-Store(1,I))/SMDELTA
DF(2*I,2*I)= (NUMYP-STORE(2,I))/SMDELTA
XINT=XINT+BIGH
10 CONTINUE
A(1,1)=1  
A(1,2)=0  
A(2,1)=0  
A(2,2)=0  
B(1,1)=0  
B(1,2)=0  
B(2,1)=1  
B(2,2)=0

Computing the vector W

W(1)=0  
W(2)=0  
GG(1,1)=DF(19,19)  
GG(1,2)=DF(19,20)  
GG(2,1)=DF(20,19)  
GG(2,2)=DF(20,20)  

Do 15 I=9,1,-1  
Do 20 J=1,2  
GF(J)=0

Do 25 K=1,2  
GF(J)=GF(J)+GG(J,K)*F(K,I)  
25 Continue  
20 Continue

W(1)=W(1)+GF(1)  
W(2)=W(2)+GF(2)

AB(1,1)=DF(2*I-1,2*I-1)  
AB(1,2)=DF(2*I-1,2*I)  
AB(2,1)=DF(2*I,2*I-1)  
AB(2,2)=DF(2*I,2*I)

Do 30 I=1,2  
Do 35 J=1,2  
WW(I,J)=0

Do 40 K=1,2  
WW(I,J)=WW(I,J)+AB(I,K)*GG(K,J)  
40 Continue  
35 Continue  
30 Continue
Do 45 j=1,2
Do 50 k=1,2
GG(j,k)=WW(j,k)
50 Continue
45 Continue
15 Continue

W(1)=W(1)+F(10,1)
W(2)=W(2)+F(10,2)
Do 55 I=1,2
V(I)=0
Do 60 J=1,2
V(I)=V(I)+B(I,J)*W(J)
60 Continue
55 Continue
W(1)=-(f(1,11)+V(1))
W(2)=-(f(2,11)+ V(2))

We now multiply B*GG and add A to get linear system to solve

Do 65 I=1,2
Do 70 J=1,2
LINS(I,J)=0
Do 75 K=1,2
LINS(I,J)=LINS(I,J)+B(I,K)*GG(K,J)
75 CONTINUE
LINS(I,J)=LINS(I,J)+A(I,J)
70 Continue
65 Continue

Solving the system for the unknown vector DeltaS1

DENO=LINS(2,1)*LINS(1,2)-LINS(1,1)*LINS(2,2)
DELTA(2)=(LINS(2,1)*W(1)-LINS(1,1)*W(2))/DENO
DELTA(1)=(W(1)-(LINS(1,2)*DELTA(2)))/LINS(1,1)
Guess(1,1)=Guess(1,1)+DELTA(1)
GUESS(2,1)=Guess(2,1)+DELTA(2)

Do 81 I=1,10
G(1,1)=DF(2*I-1,2*I-1)
G(1,2)=DF(2*I-1,2*I)
G(2,1)=DF(2*I,2*I-1)
G(2,2)=DF(2*I,2*I)
Do 85 J=1,2
STAR(J)=0
DO 90 k=1,2
  STAR(J)=STAR(J)+G(J,K)*DELTA(K)
90    CONTINUE
DELTA(J)=STAR(J)+F(J,I)
GUESS(J,I+1)=GUESS(J,I+1)+DELTA(J)
85    CONTINUE
81    CONTINUE
100   Continue
300   STOP
END

Program to solve system of two diff.equations by Runge-Kutta
Subroutine RKUTTA(x,ys,zs,numy,numyp)
REAL KONE,KTWO,KTHREE,KFOUR,LONE,LTWO,LTHREE,LFOUR,numy,numyp

F(x,y,z)=z
G(x,y,z)=-y

H=3.14159265/140
N=7

Do 5 I=1,n
  y=ys
  z=zs

  kone=F(x,y,z)
  Lone=G(x,y,z)
  ktwo=F(x+h/2.,y+h/2.*kone,z+h/2.*lone)
  Ltwo=G(x+h/2.,y+h/2.*kone,z+h/2.*lone)
  kthree=F(x+h/2.,y+h/2.*ktwo,z+h/2.*ltwo)
  Lthree=G(x+h/2.,y+h/2.*ktwo,z+h/2.*ltwo)
  kfour=F(x+h,y+h*ktwo,z+h*lthree)
  Lfour=G(x+h,y+h*ktwo,z+h*lthree)
  ys=ys+h/6.*(kone+2.*ktwo+2.*kthree+kfour)
  ZS=ZS+H/6.*(LONE+2.*LTWO+2.*LTHREE+LFOUR)
  x=x+h
5    CONTINUE
numy=ys
numyp=z
RETURN
End
2. APPENDIX B: BIBLIOGRAPHY


Lipton, David; Poterba, James; Sachs, Jeffrey; Summers, Lawrence. Multiple shooting in rational expectation models. Econometrica, 50, (1982), 1329-1333.


