Solving Differential Equations with PLAS

by

Edward M. Rosen
EMR Technology Group

Introduction

The PLAS (Power Law Analysis and Simulation) program was created by Antonio E. N. Ferreira, Instructor at the Department of Chemistry and Biochemistry at the University of Lisbon, Portugal. It was designed specifically to aid in the modeling of biological systems. The program is freely available for download and non-commercial use and is particularly attractive for use in an educational environment.

A full description of this approach, called Power-Law Modeling, can be found in Reference (1). The PLAS program is provided on a CD with the book (Version 1.0).

The program (Version 1.2) may be also be downloaded directly from the web site indicated in Reference (2). A link to the Power Law Formalism Web Site is provided there.

An Example Problem

Consider the following set of (stiff) differential equations taken from Reference (3):

$dy_1/dt = k_1 * (y_2 - y_1 * y_2 + y_1 - k_2 * y_1^2)$

$dy_2/dt = (-y_2 - y_1 * y_2 + y_3)/k_3$

$dy_3/dt = k_4 * (y_1 - y_3)$

where:

k1 = 77.27
k2 = 0.000008375
k3 = 77.27
k4 = 0.161

Initial conditions are:

$y_1(0) = 4.0$
$y_2(0) = 1.1$
$y_3(0) = 4.0$

The exact solution (Reference (3)) at time = 300 is given as:

y1 = 4.41830

y2 = 1.29024

y3 = 3.01929

The PLAS Program

The PLAS program is extremely easy to use and set up. Figure 1 is the input of the example problem. A full screen editing capability is provided. Comments can be entered directly into the input file without use of any special symbols. Transformations can be added in any order. Multiplications are entered without * and powers are indicated with ^. Note that the initial time, step size for the integration and final time are added with the reserved symbols, t0, hr and tf.

The output is provided by graphs and tables. The full syntax is described in a pull down menu. An option allows for the selection of the integration method (Taylor or BDF).

One cannot print the the output graphs or tables directly. Instead keystroke Ctrl+C must be used to copy and then to paste the results into another program for printing.

PLAS found the following solution (Version 1.0, BDF/Gear) to the Example problem using a step size of hr = 0.001:

y_1 = 4.418453
y_2 = 1.290236
y_3= 3.019847

Using IMSL

It is interesting to compare the results of PLAS with a spreadsheet implementation of IMSL routine DDASPG (Petzold-Gear Method). The VBA array function DAEV shown in Figure 2 was utilized. This was invoked from the spreadsheet shown in Figure 3. The array function DAEV in turn calls FORTRAN subroutine DDTHREE (shown in Figure 4) which was compiled with Digital's Visual Fortran and placed into the Emrdll directory as DDTHREE. dll. The general procedure for calling IMSL routines from a spreadsheet has been described in Reference (4).

Conclusions

The PLAS system provides a very simple but powerful means of integrating simultaneous differential equations (initial value problems). It is freely available for non-commercial use.

References

1. Voit, Eberhard, *Computational Analysis of Biochemical Systems* 2000, Cambridge University Press, Cambridge, England

2. PLAS (Power Law Analysis and Simulation) web site: http://correio.cc.fc.ul.pt/~aenf/plas.html

3 . Chan, Ian Yau Nam, *Numerical Solution of Stiff Differential System*s, Ph. D. Thesis in Chemical Engineering, Princeton University, January 1978

4. Rosen, E. M. "On the Use of IMSL Routines in Excel 7.0" *CACHE News* , No 49 Fall 1999

Three Stiff Equations from
Numerical Solution of Stiff Differential Systems
CHAN IAN YAU NAM
Ph.D. Thesis in Chemical Engineering, Princeton University 1978

```
Differential Equations
======================

y1' = k1 (y2 - y1 y2 + y1 - k2 y1 y1)

y2' = (-y2 - y1 y2 + y3)/k3

y3' = k4 (y1 - y3)

Initial Values
==============

y1 = 4
y2 = 1.1
y3 = 4.0


Equation Parameters
===================
k1 = 77.27

k2 = 8.375 10^-6

k3 = 77.27

k4 = 0.161

Integration Controls: Starting time, Time Increment and Final Time
==================================================================

t0 = 0
hr = 0.001
tf= 300
```

Figure 1
Input File to PLAS

```
Option Explicit

Declare Sub DDTHREE Lib "C:\Emrdll\DDTHREE.dll" _
(ByRef nL As Long, ByRef pp As Double, ByRef ttf As Double, _
ByRef hd As Double, ByRef yy As Double, ByRef ypr As Double, _
ByRef tout As Double)


Public Function DAEV(n, h, tf, prm, y)

Application.Volatile True

Dim tout As Double
Dim nL As Long
Dim nn, mm As Integer
Dim I, K As Integer
Dim sumypr As Single
Dim ttf As Double
Dim hd As Double
Dim tt As Single

nL = n
nn = n + 2

ReDim yy(1 To n) As Double
ReDim dd(1 To nn) As Single
ReDim ypr(1 To n) As Double

mm = 4
ReDim pp(1 To mm) As Double

ttf = tf
hd = h

For I = 1 To n
yy(I) = y(I)
Next I

For I = 1 To mm
pp(I) = prm(I)
Next I
```

Figure 2

VBA Array Function DAEV

```
Call DDTHREE(nL, pp(1), ttf, hd, yy(1), ypr(1), tout)

tt = tout

sumypr = 0
For I = 1 To n
sumypr = sumypr + Abs(ypr(I))
Next I


dd(1) = sumypr
dd(2) = tt

For I = 3 To nn
dd(I) = yy(I - 2)
Next I


DAEV = dd

End Function
```

Figure 2 (Cont)

Three Equations
Numerical Solution of Stiff Differential Systems Ian Yau Nam Chan Ph. D. Thesis Princeton, 1978

h	0.5
tf	300
n	3

Parameters (23)	
k1	77.27
k2	0.000008375
k3	77.27
k4	0.161

Steady State Indicator	**Time**	**y1**	**y2**	**y3**
sum of derivatives	0	4	1.1	4
0.871006191	300	4.41830254	1.290244818	3.0192826

Figure 3

Spreadsheet Implementation of the Example Problem

```
      SUBROUTINE DDTHREE (N, PRM, TF, HD, Y, YPR, TOUT)
!DEC$ ATTRIBUTES DLLEXPORT::DDTHREE
!DEC$ ATTRIBUTES ALIAS:'DDTHREE' ::DDTHREE
      USE NUMERICAL_LIBRARIES
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      EXTERNAL GCN
      INTEGER N, IDO
      DOUBLE PRECISION Y(N), T, TOUT, YPR(N)
      DOUBLE PRECISION  PRM(4)
      DOUBLE PRECISION TF, HD, SUMYPR, XN

      COMMON /TRANS/PP(4)

C     OPEN (UNIT =9, FILE ='C:\EMRDLL\DEBUG.OUT')

      DO 20 I = 1, 4
   20 PP(I) = PRM(I)

      XN = N

      IDO = 1
      T = 0.0
      NSTEP = TF/HD + 0.000001

      CALL DRV (N, Y, YPR)

      ISTEP = 0
   10 CONTINUE
      ISTEP = ISTEP + 1

      CALL DDASPG (N, T, T+HD, IDO, Y, YPR, GCN)

      IF (IDO .EQ. 3) GO TO 99

      IF ( ISTEP .LE. NSTEP ) THEN

         SUMYPR = 0
         DO 40 I = 1, N
   40    SUMYPR = SUMYPR + DABS(YPR(I))

         IF (SUMYPR .LE. 0.0000001* XN) THEN
            IDO = 3

            GO TO 10

         END IF


      END IF
```

Figure 4
FORTRAN Subroutine DDTHREE Calling IMSL Routine DDASPG

```
      IF (ISTEP .EQ. NSTEP) IDO = 3
      GO TO 10

   99 CONTINUE
      TOUT = T

      RETURN
      END

      SUBROUTINE DRV (N, Y, YPR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)

      INTEGER  N
      DOUBLE PRECISION  Y(N), YPR(N)

      COMMON /TRANS/PP(4)

      XK1 = PP(1)
      XK2 = PP(2)
      XK3 = PP(3)
      XK4 = PP(4)


      YPR(1) = XK1 * (Y(2) - Y(1) * Y(2) + Y(1) - XK2* Y(1)**2)
      YPR(2) = (-Y(2) - Y(1) * Y(2) + Y(3))/XK3
      YPR(3) = XK4 * (Y(1) - Y(3))


      RETURN
      END
      SUBROUTINE GCN (N, T, Y , YPR, GVAL)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      INTEGER N
      DOUBLE PRECISION  T,  GVAL(N)
      DOUBLE PRECISION Y(N), YPR(N), RHS(N)

      CALL DRV (N, Y, RHS)

      DO 10 I = 1, N

   10 GVAL(I) = YPR(I) - RHS(I)

      RETURN
      END
```

Figure 4 (Cont)