SYSDIFEQ – Systems of First-order Differential Equations

© 2019 Valentín Albillo

Abstract

SYSDIFEQ is a program written in 1980 for the HP-41C programmable calculator to obtain an approximate numerical solution for a system of $N$ simultaneous first-order differential equations using a fourth-order Runge-Kutta method. Two worked examples included.

Keywords: numerical solution, first-order differential equations, system, Runge-Kutta, programmable calculator, HP-41C, HP42S

1. Introduction

SYSDIFEQ is a 137-step RPN program that I wrote in 1980 for the HP-41C programmable calculator (will also run in the HP-41CV/CX and the HP42S with trivial or no changes, see Note 1), to obtain an approximate numerical solution (using a $4^{th}$-order Runge-Kutta method) for a system of $N$ simultaneous first-order differential equations of the general form:

\[
y_1' = f_1(x, y_1, y_2, ..., y_n) \\
y_2' = f_2(x, y_1, y_2, ..., y_n) \\
... \\
y_n' = f_n(x, y_1, y_2, ..., y_n)
\]

with the initial conditions: $x = x_0$, $y_1 = y_1(x_0)$, ..., $y_n = y_n(x_0)$

where the $f_i$ are user-defined functions of the variables $x, y_1, y_2, ..., y_n$ and $y_1, y_2, ..., y_n$ are functions of $x$.

We seek to compute the values $y_i(x_0 + h)$, where $h$ is an arbitrary increment of the independent variable $x$ and the index $i = 1, 2, ..., n$. The input consists of:

a) the equations: $y_i' = f_i(x, y_1, y_2, ..., y_n)$  
   b) the increment: $h$ (generally small)  
   c) the initial values: $y_i(x_0)$

and the output is $y_i(x_0 + h), y_i(x_0 + 2h), ...$, for $i = 1, 2, ..., n$.

To this purpose, the $4^{th}$-order Runge-Kutta method of Gill is applied as shown in the following pseudocode:

\[
\text{for } j = 1 \text{ to } 4 \\
\text{for } i = 0 \text{ to } n \\
\quad k_{i,j} = f_i(y_{0,j-1}, y_{1,j-1}, ..., y_{n,j-1}) \\
\text{next } i \\
\text{for } i = 0 \text{ to } n \\
\quad y_{i,j} = y_{i,j-1} + h \left( a_1 k_{i,j} + a_2 k_{i,j-1} + a_3 k_{i,j-1} + a_4 k_{i,j-1} \right) \\
\quad q_{i,j} = y_{i,j} + 3 \left( a_1 k_{i,j} + a_2 k_{i,j-1} + a_3 k_{i,j} + a_4 k_{i,j} \right) - c_j k_{i,j} \\
\text{next } i \\
\text{next } j \\
\text{output } y_i \text{ for } i = 0 \text{ to } n
\]

This procedure is repeated as often as desired to yield $y_i(x_0 + h), y_i(x_0 + 2h), ...$

The notation $y_0 = x$ is used for notational convenience and to simplify the form of the process.

Initially $q_{i,0} = 0$, $y_{i,0} = y_i(x_0)$, thereafter $q_{i,j}(x_i) = q_{i,4}(x_{i-1})$
1.1 N-th order differential equations

A single differential equation of orden $n$: 

$$y^{(n)} = f(x, y, y', y'', \ldots, y^{(n-1)})$$

subject to initial conditions 

$$x = x_0, y = y_0, y' = y'_0, \ldots, y^{(n-1)} = y^{(n-1)}_0$$

can be reduced to a system of $n$ first-order differential equations and solved using this program.

For example, consider the $3^{rd}$-order differential equation:

$$y''' = 6y'' + 7y' - 8y - 4x + 7$$

subject to initial conditions $x_0 = 1, y(1) = 2, y'(1) = 3, y''(1) = 4$

and substitute: $y = y_1$, $y' = y_2$, $y'' = y_3$, and thus $y''' = y_3'$, so the equation may be written as:

$$y_3' = 6y_3 + 7y_2 - 8y_1 - 4x + 7$$

$$y_2' = y_3$$

subject to initial conditions $x_0 = 1, y_1(1) = 2, y_2(1) = 3, y_3(1) = 4$

$$y_1' = y_2$$

which is a system of 3 first-order differential equations that can be solved using this program.

Using four memory modules, in theory equations up to the $80^{th}$ order might be solved because after the last one all the others take only 5 bytes each to define.

1.2 Program characteristics

This program (labeled as “RK” for Range-Kutta) is 32-register, 224 bytes long so it exactly fits into a single magnetic card as long as it doesn’t have a final END instruction, let the final .END. of program memory do the task (else it won’t fit into a single card, a second card would be required.)

The program solves a system of $N$ first-order differential equations, where $N$ depends on the available memory:

<table>
<thead>
<tr>
<th>Number of modules</th>
<th>Number of equations</th>
<th>Average size eq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>2</td>
<td>18 bytes</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>63 bytes</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>74 bytes</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>72 bytes</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>71 bytes</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>41 bytes</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>25 bytes</td>
</tr>
</tbody>
</table>

Of course, the more equations in the system the more bytes it will probably take to define each. The program is also fast, the loops are designed to run as fast as possible by duplicating indirect addresses. Computing times depend mostly on the number and complexity of the functions you define but the following times are typical:

<table>
<thead>
<tr>
<th>Number of equations</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time per point</td>
<td>12 sec.</td>
<td>22 sec.</td>
<td>32 sec.</td>
<td>62 sec.</td>
</tr>
</tbody>
</table>
2. Program Listing

```
01 ♦LBL "RK" 29 STO 09 57 ISG 14 85 STO 15 113 ST+ 04
02 CLRG 30 STO 00 58 GTO 01 ♦ 86 RCL 10 114 ISG 13
03 FIX 0 31 ENTER↑ 59 “INC=?” 87 STO 14 115 GTO 11 ♦
04 CF 29 32 SQRT 60 PROMPT 88 ♦LBL 13 116 SF 29
05 SF 21 33 LASTX 61 STO 12 89 RCL IND 15 117 GTO 11 ♦
06 “N=?” 34 + 62 ♦LBL 14 90 RCL IND 13 118 ♦LBL 10
07 PROMPT 35 STO 01 63 .002 91 / 119 RCL 10
08 STO 04 36 / 64 .002 92 3 120 RCL 10
09 3 37 STO 02 65 6 93 FS? 29 121 ADV
10 * 38 RCL 04 66 STO 04 94 ST* Y 122 ♦LBL 02
11 22 39 ST- 05 67 STO 03 95 RCL IND 16 123 “Y”
12 + 40 20 68 ♦LBL 11 96 RCL IND 04 124 FIX 0
13 STO 05 41 + 69 RCL 10 97 * 125 ARCL X
14 SF 25 42 ST+ 11 70 STO 14 98 RCL IND 15 126 ├“=”
15 ARCL IND X 43 1 E3 71 RCL 11 99 - 127 FIX 7
16 1 44 / 72 STO 15 100 RCL IND 13 128 ARCL IND Y
17 STO 11 45 20 73 RCL 05 101 / 129 AVIEW
18 STO 07 46 + 74 STO 16 102 ST- IND 14 130 ISG X
19 STO 08 47 STO 14 75 ♦LBL 12 103 * 131 X<>Y
20 + 48 STO 10 76 XEQ IND 14 ♦ 104 + 132 ISG Y
21 FS7C 25 49 ♦LBL 01 77 RCL 12 105 ST- IND 16 133 GTO 02 ♦
22 GTO 00 ♦ 50 “Y” 78 * 106 1 134 GTO 14 ♦
23 “SIZE” 51 ARCL 12 79 STO IND 15 107 ST+ 15 135 ♦LBL 20
24 ARCL X 52 ├“=” 80 ISG 15 108 ST+ 16 136 1
25 PROMPT 53 PROMPT 81 X<>Y 109 ISG 14 137 RTN
26 ♦LBL 00 54 STO IND 14 82 STO 14 110 GTO 13 ♦ 138 END.
27 2 55 ISG 12 83 GTO 12 ♦ 111 FS7C 29
28 STO 06 56 X<>Y 84 RCL 11 112 GTO 10 ♦
```

The symbols ♦ and ♦ are purely cosmetic, to indicate branching: ├ is the Append alpha function

2.1 Contents of the storage registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
<th>Register</th>
<th>Contents</th>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>a₁ = 2</td>
<td>10</td>
<td>add.f,y</td>
<td>20</td>
<td>y₀ = x</td>
</tr>
<tr>
<td>01</td>
<td>a₂ = 3.41+</td>
<td>11</td>
<td>add.k</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>02</td>
<td>a₃ = 0.58+</td>
<td>12</td>
<td>h</td>
<td>N+20</td>
<td>y₂</td>
</tr>
<tr>
<td>03</td>
<td>a₄ = 6</td>
<td>13</td>
<td>add.a</td>
<td>N+21</td>
<td>k₀</td>
</tr>
<tr>
<td>04</td>
<td>N, add.b</td>
<td>14</td>
<td>add.f,y var.</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>05</td>
<td>add q = 2N+22</td>
<td>15</td>
<td>add k var.</td>
<td>2N+21</td>
<td>k₀</td>
</tr>
<tr>
<td>06</td>
<td>b₁ = 2</td>
<td>16</td>
<td>add q var.</td>
<td>2N+22</td>
<td>q₀</td>
</tr>
<tr>
<td>07</td>
<td>b₂ = 1</td>
<td>17</td>
<td>free</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>08</td>
<td>b₃ = 1</td>
<td>18</td>
<td>free</td>
<td>3N+22</td>
<td>q₀</td>
</tr>
<tr>
<td>09</td>
<td>b₄ = 2</td>
<td>19</td>
<td>free</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. Usage Instructions

To solve a system of N first-order differential equations perform the following Steps:

**Step 1:** in RUN Mode: \( \text{GTO .138} \) → \{ switch to PRGM Mode \}.

If there are definitions left from previously solved systems, execute:

\[ \text{DEL 999} \] → \{ you should see: 137 RTN \}

**Step 2:** still in PRGM Mode, define the equations. Every equation \( f_i \) must be defined under a numerical label whose number is \( i+20 \). This is: \( f_1 \) must be defined under LBL 21, \( f_7 \) under LBL 27, \( f_{14} \) under LBL 34, and so on. Each definition must be terminated with a RTN instruction.

To define each equation, \( y_i \) can be found stored in register \( R_{i+20} \) and the convention \( y_0 = x \) is used. This is: \( x (= y_0) \) can be found in \( R_{20} \), \( y_1 \) in \( R_{21} \), ..., \( y_{14} \) in \( R_{34} \), and so on.

You may use \( R_{17}, R_{18} \) and \( R_{19} \) as scratch registers if you wish but do not disturb the contents of any other registers from register \( R_0 \) up to \( R_{3N+22} \), both included. See Examples.

**Step 3:** once the equations have been defined in program memory, switch back to RUN Mode and execute the program, as follows:

\[ \text{XEQ "RK"} \rightarrow N=? \ \{ \text{asks for the number of equations in the system} \} \]

\[ N \ \text{R/S} \rightarrow Y0=? \ \{ \text{remember the convention } y_0 = x_0 \text{, so it's asking for } x_0 \} \]

\[ x_0 \ \text{R/S} \rightarrow Y1=? \ \{ \text{asks for } y_1(x_0) \} \]

\[ y_1 \ \text{R/S} \rightarrow Y2=? \ \{ \text{asks for } y_2(x_0) \} \]

... ... ...

\[ y_n \ \text{R/S} \rightarrow \text{INC=?} \ \{ \text{asks for the increment [step] } h \} \]

**Note:** the increment (step) \( h \) should be adequately small. The smaller is \( h \), the greater is the accuracy and number of stages required to reach a given value of \( x \) (and so the longer it will take and the more rounding errors will accumulate). The error per stage is approximately proportional to \( h^5 \) so as a rule of thumb \( h=0.1 \) should give about 5 correct places at the end of 10 stages or so.

\[ h \ \text{R/S} \rightarrow \ \{ \text{computation begins} \} \]

After some time, the values of \( y_i(x_0 + h) \) will be output. If there's a printer attached and On, all values will be printed and spaced and computation will proceed to the following stage automatically. Otherwise, simply press \( \text{R/S} \) after each displayed value to see the next one or resume the procedure, like this:

\[ \rightarrow Y0=\text{its value} \ \{ \text{remember, } y_0 = x_0 \text{, so this is } x = x_0 + h \text{ in this stage} \} \]

\[ \rightarrow Y1=\text{its value} \ \{ \text{outputs } y_1(x_0 + h) \} \]

\[ \rightarrow Y2=\text{its value} \ \{ \text{outputs } y_2(x_0 + h) \} \]

... ... ...

\[ \rightarrow YN=\text{its value} \ \{ \text{outputs } y_n(x_0 + h) \text{, the last value in this stage} \} \]

\[ \rightarrow Y0=\text{its value} \ \{ \text{proceeds to compute the next stage, } y_i(x_0 + 2h) \} \]

and so on. To stop the procedure at any time, press \( \text{R/S} \). For another system, go to Step 1.
Notes:

- if after introducing N a message SIZE nnn does show up this means the current register allocation is insufficient to run the program, so simply execute SIZE nnn and then press R/S to resume.
  In general, if your system has N equations, a minimum SIZE of 3N+23 registers is required.

- do not disturb the stack while the output is taking place. If you want to simply view (not write down) each value, without having to press R/S each time to continue (you have no printer attached), then at any moment stop the computation by pressing R/S and execute CF 21, then R/S again to resume. Subsequent values will be displayed without stopping program execution.

- if you want to change the spacing (step) h during the process, wait for the program to stop or output something, then stop (if not yet halted) and input the new h like this:

  new value of h STO 12 R↓ R/S.

  the procedure will resume from where it left but this time using the new h.

- to change the number of decimals in the output, change line 127 FIX 7 to the desired display setting.

4. Examples

The following examples can be useful to check that the program is correctly entered and to understand its usage.

4.1 Example 1

Solve the system:

\[
\begin{align*}
y_1' &= \sin x - y_2 \\
y_2' &= e^x + y_3 \\
y_3' &= 1 - y_1 - y_2
\end{align*}
\]

with initial conditions:

\[
\begin{align*}
x_0 &= 0.230253487 \\
y_1 &= -0.258919089 \\
y_2 &= 1.487143417 \\
y_3 &= 0.973608574
\end{align*}
\]

using a step \( h = -0.102342187 \).

Define the equations: in RUN Mode, GTO .138, switch to PRGM Mode, DEL 999 → see: 137 RTN

Enter the definitions below, and then switch to RUN Mode, execute PACK, RAD and run the program:

\[
\begin{align*}
138 & \text{LBL 21} \\
139 & \text{RCL 20} \\
140 & \text{SIN} \\
141 & \text{RCL 22} \\
142 & \text{-} \\143 & \text{RTN} \\
144 & \text{LBL 22} \\
145 & \text{RCL 20} \\
146 & \text{E↑X} \\
147 & \text{RCL 23} \\148 & \text{+} \\149 & \text{RTN} \\
150 & \text{LBL 23} \\
151 & 1 \\
152 & \text{RCL 21} \\
153 & - \\
154 & \text{RCL 22} \\
155 & - \\
156 & \text{RTN} \\
\end{align*}
\]

XEQ "RK" → N=?

3 R/S → Y0=?

0.230253487 R/S → Y1=?

-0.258919089 R/S → Y2=?

1.487143417 R/S → Y3=?

0.973608574 R/S → INC=?

-0.102342187 R/S → computing proceeds and then ...
... the output begins → \( Y0=0.1279113 \)  \{ remember, \( y_0 = x_0 \), so this is \( x = x_0 + h \) in this first stage\}

\[
\begin{align*}
\text{R/S} & \quad \rightarrow \quad Y1=-0.1364522 \quad \text{\{ outputs } y_1(x_0 + h) \} \\
\text{R/S} & \quad \rightarrow \quad Y2=1.2640152 \quad \text{\{ outputs } y_2(x_0 + h) \} \\
\text{R/S} & \quad \rightarrow \quad Y3=0.9918304 \quad \text{\{ outputs } y_3(x_0 + h), \text{ the last value in this stage } \} \\
\text{R/S} & \quad \rightarrow \quad Y0=0.0255691 \quad \text{\{ this is } x = x_0 + 2h \text{ in this second stage} \} \\
\text{R/S} & \quad \rightarrow \quad Y1=-0.0258989 \quad \text{\{ outputs } y_1(x_0 + 2h) \} \\
\text{R/S} & \quad \rightarrow \quad Y2=1.0514656 \quad \text{\{ outputs } y_2(x_0 + 2h) \} \\
\text{R/S} & \quad \rightarrow \quad Y3=0.9996729 \quad \text{\{ outputs } y_3(x_0 + 2h), \text{ the last value in this stage, and so on ... } \}
\end{align*}
\]

To check the accuracy obtained, the exact solution is:

\[
\begin{align*}
y_1 &= 1 - e^x \\
y_2 &= e^x + \sin x \\
y_3 &= \cos x
\end{align*}
\]

\[
\text{thus: } y_1(x_0 + h) = 0.136452195, \quad y_2(x_0 + h) = 1.264014981, \quad \text{so we got 7-8 correct places}
\]

\[
y_3(x_0 + h) = 0.991830497
\]

4.2 Example 2

Solve the system:

\[
\begin{align*}
y_1' &= y_1 - y_2 + e^x - y_4 - x \\
y_2' &= y_1 - \sin x + e^x \\
y_3' &= \cos x - y_3 - y_4 - x \\
y_4' &= y_3 - e^{-x} - 1 \\
y_5' &= (y_5 + \sin x - y_4)^2
\end{align*}
\]

with initial conditions:

\[
\begin{align*}
x_0 &= 0 \\
y_1 &= 1 \\
y_2 &= 1 \\
y_3 &= 2 \\
y_4 &= y_5 &= 0
\end{align*}
\]

Using a step \( h = 0.1 \), find the values of \( y_1 .. y_5 \) for \( x = 1 \).

Define the equations: in \textbf{RUN Mode}, \textbf{GTO .138}, switch to \textbf{PRGM Mode}, \textbf{DEL 999} → see: \textbf{137 RTN}

Enter the 5 definitions below, and then switch to \textbf{RUN Mode}, execute \textbf{PACK}, \textbf{RAD} and run the program:

\[
\begin{array}{cccccc}
138 & \bullet \text{LBL} & 21 & 150 & \bullet \text{LBL} & 22 \\
139 & \text{RCL} & 21 & 151 & \text{RCL} & 21 \\
140 & \text{RCL} & 22 & 152 & \text{RCL} & 20 \\
141 & - & 153 & \text{SIN} & 154 & \text{RCL} & 20 \\
142 & \text{RCL} & 20 & 155 & \text{RCL} & 20 \\
143 & \text{E} \uparrow \text{X} & 156 & \text{RCL} & 20 \\
144 & + & 157 & \text{RCL} & 24 \\
145 & \text{RCL} & 24 & 158 & \text{RTN} \\
146 & - & 159 & \text{LBL} & 23 \\
147 & \text{RCL} & 20 & 160 & \text{RCL} & 20 \\
148 & - & 161 & \text{LCL} & 24 \\
149 & \text{RTN} & 162 & \text{LCL} & 25 \\
169 & \text{LCL} & 24 & 170 & \text{RCL} & 23 \\
178 & \text{LCL} & 25 & 179 & \text{RCL} & 25
\end{array}
\]

\[
\begin{align*}
\text{XEQ } \text{"RK"} & \quad \rightarrow \quad N=? \\
5 & \quad \text{R/S} \quad \rightarrow \quad \text{SIZE 38 } \quad \star^1 \\
0 & \quad \text{R/S} \quad \rightarrow \quad Y0=? \\
0 & \quad \text{R/S} \quad \rightarrow \quad Y1=? \\
1 & \quad \text{R/S} \quad \rightarrow \quad Y2=? \\
1 & \quad \text{R/S} \quad \rightarrow \quad Y3=? \\
2 & \quad \text{R/S} \quad \rightarrow \quad Y4=? \\
0 & \quad \text{R/S} \quad \rightarrow \quad Y5=? \\
0 & \quad \text{R/S} \quad \rightarrow \quad \text{INC=?} \\
0.1 & \quad \text{R/S} \quad \rightarrow \quad \text{computation proceeds and then } ...
\end{align*}
\]

\[
\star^1 \text{ The example assumes there were less than 38 registers allocated so it prompts for the user to allocate them right now, before resuming.}
\]

6
... the output begins → Y0=0.1  { remember, y_0 = x_0 . so this is x = x_0 + t = 0.1 in this first stage}
R/S → Y1=1.0948375  { outputs y_1 (0.1) }
R/S → Y2=1.2050044  { outputs y_2 (0.1) }
R/S → Y3=1.8998416  { outputs y_3 (0.1) }
R/S → Y4=-0.0001665  { outputs y_4 (0.1) }
R/S → Y5=0.0003345  { outputs y_5 (0.1), the last value in this stage }
R/S → ...  { proceeds to compute the next stage, y_1 (0.2) }

After 10 such stages you should get:

\[
\begin{align*}
x &= 1.0000000 \\
y_1 &= 1.3817719 \\
y_2 &= 3.5597527 \\
y_3 &= 0.9081817 \\
y_4 &= -0.1585284 \\
y_5 &= 0.5573977
\end{align*}
\]

and the exact solution is:

\[
\begin{align*}
y_1 &= \sin x + \cos x \\
y_2 &= \sin x + e^x \\
y_3 &= \cos x + e^{-x} \\
y_4 &= \sin x - x \\
y_5 &= \tan x - x
\end{align*}
\]

so we got 5-6 correct places

Notes

1. This program will also run in the HP42S but as it has a wider numeric range and greater precision (12-digit instead of 10-digit), the results might differ slightly. Consult the HP42S Owner’s Manual for other minor differences (e.g.: naming).

2. This program was published in PPC Melbourne Chapter Technical Notes V1N2 pp23-28 (September 1980). However, there’s a missing minus sign ("−") between \( e^x \) and \( y_4 \) in the definition of the first equation of the second example.

3. It was also published in PPC Calculator Journal V8N1 pp17-19 (Jan./Feb. 1981). However, there’s also a missing minus sign ("−") between \( e^x \) and \( y_4 \) in the definition of the first equation of the second example.

References


Copyrights

Copyright for this paper and its contents is retained by the author. Permission to use it for non-profit purposes is granted as long as the contents aren’t modified in any way and the copyright is acknowledged.