LSYSGS – Systems of Linear Equations - Gauss-Seidel

© 2019 Valentín Albillo

Abstract

LSYSGS is a program written in 1980 for the HP-41C programmable calculator to solve a system of N linear algebraic equations in N unknowns using the iterative method of Gauss-Seidel. Two worked examples included.

Keywords: *system of linear equations, iterative method, Gaus-Seidel, overrelaxation, programmable calculator, HP-41C, HP42S*

1. Introduction

LSYSGS is a *129-step RPN* program that I wrote in 1980 for the *HP-41C* programmable calculator (will also run *as-is* in the *HP-41CV/CX* and in the *HP42S* with trivial changes, see ***Note 1***) to obtain a numerical solution for a system of *N* algebraic linear equations in N unknowns using the method of *Gauss-Seidel*, an iterative process for approximating its solution. This procedure is useful for equations which can be so arranged that the diagonal elements of the coefficient matrix dominate the other elements (see ***Note 3***). Briefly, one wishes to solve:

$$\left.\begin{matrix} a_{1,1}x_1 + \ldots + a_{1,n}x_n = b_1 \\ \ldots \\ a_{n,1}x_1 + \ldots + a_{n,n}x_n = b_n \end{matrix}\right\} \Rightarrow \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

The *Gauss-Seidel* algorithm proceeds as follows:

- choose an arbitrary vector x^0. This program selects a vector with all its components equal to *0*.
- successively alter the i_{th} component of the trial vector so that the i_{th} equation is satisfied. The corresponding elements are changed by the computed increment.
- when all the changes (rounded to an specified number of decimal places) are *0*, the program stops and outputs the approximate solution. Else, repeat until all rounded increments become *0*.

This can be mathematically expressed as follows:

1. At the beginning, assign: $x_1 = x_2 = \ldots x_n = 0.$
2. Then compute:

$$\begin{aligned} x_1 &= x_1 - w*(a_{1,1}\,x_1 + a_{1,2}\,x_2 + \ldots + a_{1,n}\,x_n - b_1) / a_{1,1} \\ x_2 &= x_2 - w*(a_{2,1}\,x_1 + a_{2,2}\,x_2 + \ldots + a_{2,n}\,x_n - b_2) / a_{2,2} \\ &\ldots \qquad \ldots \qquad \ldots \\ x_n &= x_n - w*(a_{n,1}\,x_1 + a_{n,2}\,x_2 + \ldots + a_{n,n}\,x_n - b_n) / a_{n,n} \end{aligned}$$

The $\boldsymbol{x_i}$ have been changed by an increment. If all increments are *0*, then the computed solution is output. Otherwise the procedure is repeated with the new $\boldsymbol{x_i}$ as the next trial vector. The accuracy depends on the user-selected number of decimal places and the input matrix is ***not*** disturbed by the procedure and can be reused.

The constant ***w*** is a value between *1* and *2* selected by the user which often speeds convergence drastically: an optimum ***w*** can achieve convergence as much as 10 times faster. On the other hand, a bad ***w*** might result in longer execution times or even divergence. If ***w*** = *1* then the method is called a *relaxation* method, while if ***w*** > *1* and ***w*** < *2* it's then known as *overrelaxation*. See ***Examples***.

2. Program Listing

```
01 ♦LBL "S"
02 CLRG
03 FIX 0
04 CF 29
05 "N=?"
06 PROMPT
07 STO 00
08 X^2
09 LASTX
10 ST+ X
11  +
12  9
13  +
14 STO 06
15  1
16  -
17 RCL 00
18  2
19  +
20 1 E5
21  /
22  +
23 STO 07
24 SF 25
25 RCL IND 06
26 FS?C 25
27 GTO 00 ▶
28 "SIZE "
29 RCL 06
30  1
31  +
32 ARCL X
33 PROMPT
34 ♦LBL 00
35 RCL 00
36 1 E3
37  /
38  1
39  +
40 STO 02
41 STO 03
42 RCL 00
43  10
44  +
45 STO 04
46 ♦LBL 01
47 "A"
48 ARCL 02
49 "├,"
50 ARCL 03
51 "├=?"
52 PROMPT
53 STO IND 04
54 ISG 04
55 X<>Y
56 ISG 03
57 GTO 01 ▶
58 "B"
59 ARCL 02
60 "├=?"
61 PROMPT
62 STO IND 04
63 ISG 04
64 X<>Y
65 RCL 00
66 ST-03
67 ISG 02
68 GTO 01 ▶
69 "W=?"
70 PROMPT
71 STO 01
72 "FIX=?"
73 PROMPT
74 STO 09
75 FIX IND X
76 RCL 00
77  9.009
78  +
79 STO 08
80 CF 00
81 ♦LBL 02
82 RCL 07
83 STO 04
84 RCL 06
85 STO 02
86 RCL 08
87 STO 05
88 ♦LBL 04
89 RCL IND 02
90 DSE 02
91 ♦LBL 03
92 RCL IND 02
93 RCL IND Z
94  *
95  -
96 DSE 02
97 DSE Y
98 GTO 03 ▶
99 RCL IND 04
100  /
101 RCL 01
102  *
103 ST+ IND 05
104 FS? 04
105 VIEW IND 05
106 RND
107 X≠0?
108 SF 00
109 RCL 08
110 DSE 04
111 DSE 05
112 GTO 04 ▶
113 FS?C 00
114 GTO 02 ▶
115 RCL 09
116  10
117 ♦LBL 05
118 "X"
119 FIX 0
120 ARCL 03
121 "├="
122 FIX IND Y
123 ARCL IND X
124 PROMPT
125 ISG X
126 ♦LBL 06
127 ISG 03
128 GTO 05 ▶
129 .END.
```

*The symbols ♦ and ▶ are purely cosmetic, to indicate branching; ├ is the **Append** alpha function; ⬚ = see footnote.*

2.1 *Program characteristics*

This program (labeled as "***S***" for *Gauss-**S**eidel*) is 32-register, 224-byte 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 job instead (else it won't fit into a single card, a second card would be required.)

A system of up to 15 equations can be solved if using all 4 memory modules *(HP-41C)*. If none are plugged-in it can solve systems up to 3x3. In general, the required size to solve NxN systems is **SIZE** N^2**+2N+10**.

2.2 *Contents of the storage registers*[1]

Register	***Contents***	***Register***	***Contents***	***Register***	***Contents***
00	N	*06*	$a_{i,j}$, b_i count.	*N+9*	x_n
01	w	*07*	pivots	*N+10*	$a_{1,1}$
02	row c.	*08*	auxiliar	...	...
03	column c.	*09*	FIX mode	N^2+2N+9	b_n
04	pivots	*10*	x_1		
05	x_i count.	...	...		

[1] If you're able to use *synthetic instructions* you can save two additional registers for other purposes by using registers **M** and **N** instead of R_{08} and R_{09} , respectively. To that effect, perform the following changes in the program listing above:

- change all instructions STO 08, RCL 08 to **STO M**, **RCL M** and all STO 09, RCL 09 to **STO N**, **RCL N**, respectively.
- change line *12* **9** to *12* **7** , line *43* **10** to *43* **8** and line *77* **9.009** to *77* **7.007**
- change line *116* **10** to *116* **8**. The size required for NxN systems will now be **SIZE** N^2**+2N+8**.

3. Usage Instructions

To solve a system of N linear algebraic equations in N unknowns perform the following *Steps*:

Step 1: In **RUN** Mode, execute these commands:

- If you want to see each approximation while the program runs: $\boxed{\texttt{SF 04}}$ $\boxed{\texttt{CF 21}}$.
- If you'd rather see just the final computed solution: $\boxed{\texttt{CF 04}}$ $\boxed{\texttt{CF 21}}$.

	$\boxed{\texttt{XEQ}}$ "S"	→	*N=?*	*{ asks for the number of equations in the system}*
N	$\boxed{\texttt{R/S}}$	→	*Ai, j=?*	*{ asks for the coefficients of each equation in turn }*
$a_{i,j}$	$\boxed{\texttt{R/S}}$	→	..	
...	$\boxed{\texttt{R/S}}$	→	*Bi=?*	*{ ditto }*
b_i	$\boxed{\texttt{R/S}}$	→	...	
...	$\boxed{\texttt{R/S}}$	→	*Bn=?*	*{ once all coefficient have been entered ...}*
b_n	$\boxed{\texttt{R/S}}$	→	*W=?*	*{ ... it then asks for the value of **w**, must be between 1 and 2 }*
w[1]	$\boxed{\texttt{R/S}}$	→	*FIX=?*	*{ asks for the number **d** of decimal places wanted }*
d	$\boxed{\texttt{R/S}}$	→	...	

The computation starts and if you chose to see the convergence of the approximations they will be shown in the display while the program runs, otherwise the default *flying goose* indicator will appear. After a while, once convergence is achieved to the user-specified number of digits, the computed unknowns will be output, properly labeled and rounded to that many digits[2].

...	→	*X1=(its value)*	*{ the value of the first unknown }*
$\boxed{\texttt{R/S}}$	→	*X2=(its value)*	*{ the value of the second unknown }*
		...	
$\boxed{\texttt{R/S}}$	→	*Xn=(its value)*	*{ the value of the last unknown }*

Step 2: In case of non-convergence or to stop the procedure at any time, press $\boxed{\texttt{R/S}}$. To try another value of ***w***, another number of digits ***d***, or to solve 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 $\boxed{\texttt{SIZE}}$ *nnn* and then press $\boxed{\texttt{R/S}}$ to resume. In general, if your system has N equations, a minimum of $N^2+2N+10$ registers are required.
- input of the $\boldsymbol{a_{i,j}}$ may use the stack and $\boldsymbol{a_{i,i}}$ must not be *0*, rearrange the equations if necessary to ensure that. Don't disturb the stack while the output of the solution is taking place.
- accuracy and running time both depend on the selected number of digits ***d***. As a guideline, use ***d=5*** to speed things up.
- unlike *Gauss elimination*–based algorithms, the coefficient matrix is not altered by the procedure and can be used in further computations.

[1] Be careful with your choice of ***w***, as it might cause non-convergence; a value of ***1.2*** or ***1.3*** will do fine in most cases.
[2] If you have a printer attached, you may want to change line *124* **`PROMPT`** to *124* **`AVIEW`** and execute $\boxed{\text{SF 21}}$ so that each unknown $\boldsymbol{x_i}$ is automatically printed without stopping. Otherwise you'll have to press $\boxed{\textbf{R/S}}$ to display the next $\boldsymbol{x_i}$.

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: $\boxed{-6 x_1 + 2 x_2 - 2 x_3 = -16, \quad 4 x_1 - 3 x_2 + 5 x_3 = 11, \quad -8 x_1 + 2 x_2 + 7 x_3 = -13}$.
using ***w* = 1** and then ***w* = 1.22**, taking note of the respective running times. Specify 4 decimal digits.

We don't want to see the successive approximations so, in **RUN** Mode:

$\boxed{\texttt{CF 04}}$ $\boxed{\texttt{CF 21}}$ $\boxed{\texttt{XEQ}}$ "S" → *N=?* *{ asks for the number of equations in the system }* 3 $\boxed{\texttt{R/S}}$
→ *A1,1=?* -6 $\boxed{\texttt{R/S}}$ → *A1,2=?* 2 $\boxed{\texttt{R/S}}$ → *A1,3=?* -2 $\boxed{\texttt{R/S}}$ → *B1=?* -16 $\boxed{\texttt{R/S}}$
→ *A2,1=?* 4 $\boxed{\texttt{R/S}}$ → *A2,2=?* -3 $\boxed{\texttt{R/S}}$ → *A2,3=?* 5 $\boxed{\texttt{R/S}}$ → *B2=?* 11 $\boxed{\texttt{R/S}}$
→ *A3,1=?* -8 $\boxed{\texttt{R/S}}$ → *A3,2=?* 2 $\boxed{\texttt{R/S}}$ → *A3,3=?* 7 $\boxed{\texttt{R/S}}$ → *B3=?* -13 $\boxed{\texttt{R/S}}$
→ *W=?* 1 $\boxed{\texttt{R/S}}$ → *FIX=?* 4 $\boxed{\texttt{R/S}}$ → *{ after 85 seconds }*
→ *X1=3.0000* $\boxed{\texttt{R/S}}$ → *X2=2.0000* $\boxed{\texttt{R/S}}$ → *X3=1.0000*

Using ***w*=1.22** instead we get the same results but in only *50 seconds*, i.e.: 41% faster.

4.2 *Example 2*

Solve the test system: $\boxed{3 x_1 + x_2 = 5, \quad x_1 + 2 x_2 = 5}$
using ***w* =1**. Specify 4 decimal digits, select to see the successive approximations and note the running time.

in **RUN** Mode:

$\boxed{\texttt{SF 04}}$ $\boxed{\texttt{CF 21}}$ $\boxed{\texttt{XEQ}}$ "S" → *N=?* *{ asks for the number of equations in the system }* 2 $\boxed{\texttt{R/S}}$
→ *A1,1=?* 3 $\boxed{\texttt{R/S}}$ → *A1,2=?* 1 $\boxed{\texttt{R/S}}$ → *B1=?* 5 $\boxed{\texttt{R/S}}$
→ *A2,1=?* 1 $\boxed{\texttt{R/S}}$ → *A2,2=?* 2 $\boxed{\texttt{R/S}}$ → *B2=?* 5 $\boxed{\texttt{R/S}}$
→ *W=?* 1 $\boxed{\texttt{R/S}}$ → *FIX=?* 4 $\boxed{\texttt{R/S}}$ → *2.5000* → *0.8333* → *2.0833* → *0.9722*
→ *2.0139* → *0.9954* → *2.0023* → *0.9992*
→ *2.0004* → *0.9999* → ...
{ finally, after 20 seconds } → *X1=1.0000* $\boxed{\texttt{R/S}}$ → *X2=2.0000*

Notes

*1.*This program can easily be adapted to run under other *RPN* versions with minimal changes, e.g. for it to run on the *HP42S* simply change the names of intructions such as **FIX 0** to **FIX 00**, ***** to **x**, **ST+/ST-/ST*/**... to **STO+/STO-/STOx/**... , **ISG X** to **ISG ST X** and so on. Also, the *HP42S* has a wider numeric range and greater precision (12-digit instead of 10-digit), so the results might differ slightly. See the ***HP42S Owner's Manual*** section titled *"Using HP-41C Programs"* for full details

*2.*The method used does not always converge. In such case, try to rearrange the system so that the diagonal elements exceed in value all other elements in their row. In particular, if they exceed their *sum* then convergence is guaranteed.

3. This program was published in *PPC Calculator Journal V7N4 pp21-22 (May 1980).*

References

Francis Scheid (1988). *Schaum's Outline of Theory and Problems of Numerical Analysis, 2nd Edition*.

Copyrights