Macro: solve3_simple.mac

Macro	solve3_simple.mac
Description	Solves a pre-defined set of three simultaneous linear equations in three unknowns
CM version	Any

What the macro does

The macro solves the following pre-defined set of three simultaneous linear equation in the three unknowns x, y and z.

$$\begin{aligned} x + y - 8z &= -3 \\ x + 2y + z &= 9 \\ x - y + 2x &= 1 \end{aligned}$$

It is easily checked that the exact solution is the following.

$$x = 2 \qquad y = 3 \qquad z = 1$$

These equation are presented to the constraint modeller as "constraints". A constraint is an expression which is zero when it is true. So the macro specifies the above three equations using three "rule" commands. The expressions associated with these rule commands are the following.

$$\begin{aligned} x + y - 8z + 3 \\ x + 2y + z - 9 \\ x - y + 2x - 1 \end{aligned}$$

These expressions are all zero when they are true. Effectively the constraint modeller constructs the following function of the three variables

$$f(x,y,z) \;=\; \sqrt{[(x+y-8z+3)^2 \;+\; (x+2y+z-9)^2 \;+\; (x-y+2x-1)^2]}$$

and then tries to find where a minimum occurs. Clearly the function $f(x,y,z)$ is non-negative. If values for x, y, z can be found for which the function takes the minimum value of zero, then these values form a solution of the original equations.

Normally one would expect to allow all three of the variables to be changed. The macro allow one or more of the variables to be fixed. This means that their values are not changed while the modeller searches for a minimum. Usually this means that an exact solution cannot be found. For example, if one of the variables is fixed, then there are three equations in just two unknowns. This represents an over-defined system and it is likely that they are incompatible (unless the fixed variable just happens to have the right value). What the modeller finds is a sort of "best compromise" solution. This is purely in the sense that it minimises the function f.

Figure 1 shows the graphics screen after the optimisation process is complete. The values of the variable are displayed. Also is the latest "truth" value. This is the final value obtained for the function $f(x, y, z)$. By default, the modeller stops searching when this truth value is around 10^{-5}.

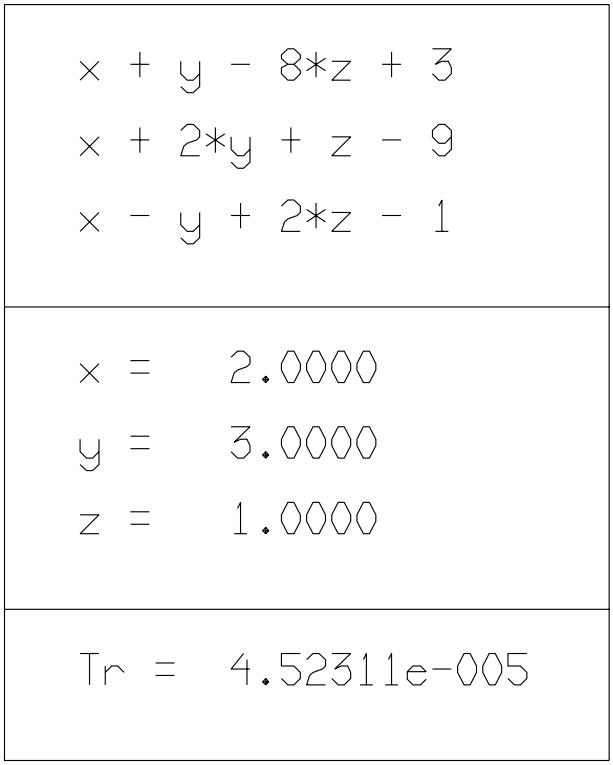

Figure 1: Graphics screen after successful solution of the three equations

How the main part of the macro works

The listing of the macro is given below. The lines of the macro are numbered for ease of reference. As is often the case, much of the macro consists of commands to make the application look good on the screen.

The main "heart" of the macro is the function called `solve` defined between lines 0073 and 0082 of the macro. The variable it uses are `x`, `y`, `z` which are declared as (global) variables in line 0009. The constraint rules are defined in lines 0077, 0078, 0079. The modeller is told by the `var` statement in line 0075 that it can change the value of each of `x`, `y`, `z` when trying to resolve the constraint rules.

The function can be called by typing in the following command

```
solve()
```

which is of course simply the name of the function (followed by brackets).

When the function is called, the modeller first of all evaluates the constraint expressions using the current values of `x`, `y`, `z`. If each rule expression is zero, then it does nothing further. If not, then the modeller tries to vary each of `x`, `y`, `z` to try to minimise the sum of the squares of the constraint expressions.

Within function `solve`, at line 0081, is a call to the fucntion `show_values` which is defined earlier in the macro file. This function shows the values of variable on the screen. It is this that causes the values on-screen to change rapidly while the constraints are being resolved. This function is used here and called from within `solve` simply as a demonstration. Showing the values in this way is not normally done since it solows down the resolution process.

How the rest of the macro works

Also defined globally are a number of variables which are used to create graphical entities, in this case lines and text items. These are declared in lines 0009–0012, and line 0013 declared a text string which is used to help in text manipulation.

The lines are defined by the (x, y, z)-coordinates of their end-points in lines 0021–0026 of the macro, and line 0027 chnages their colour to green. These are lines which form the horizontal and vertical boundaries of the on-screen display (cf. figure 1).

In line 0029, a piece of text is assigned to the variable `stemp`; in line 0030, text variable `t1` is created and the piece of text is assigned to it; and, in line 0031, the colour is chnaged to yellow. This process is then repeated for variables `t2` and `t3`.

Between lines 0039 and 0054, a user-defined fucntion (UDF) is defined called `show_values`. Here text variables `tx`, `ty`, `tz` are defined which have text giving the values of the variables `x`, `y`, `z`.

Similarly, the UDF called `show_truth` (lines 0056–0062) sets up a variables called `tr` to show the last truth value. Here the built-in function called `truth` is used. This returns the minimum value (of the square root of the sum of the constraint expressions) obtained by the last resolution process.

The function `reset` (lines 0073–0082) is used to reset the values of the three variables all to zero. It then shows these values on-screen and used to built-in function `zoom` to ensure that the graphics fill the screen as well as possible.

The last part of the macro (starting at line 0087) sets up a user menu which is called `solve_menu`. The names of the various submenus and button appear on the screen. When the user clicks on one of the button, the commands associated with it are obeyed. The first button (lines 0089–0091) simply calls the `reset` function. The next button (lines 0092-0096) calls three UDFs, namely `solve` to resolve the constraints, and then `show_values` and `show_truth` to display the results. There is then a submenu (lines 0097–0116) which allows the user to fix or free each of the

three variables. If a variable is fixed, then the modeller does not try to change it during constraint resolution (even if it appears in the appropriate `var` list). Freeing a variable allows its value to be changed. The modeller only changes a variable during constraint resolution if it is named in the appropriate `var` list and it is free.

The submenu and buttons defined by lines 0117–0129 aloow the user to assign values of each of `x`, `y`, `z`. It maes use of the built-in `val` function which puts up a dialogue box giving the user a prompt and showing the current value. (This is only an aid for the user; the value of any of these variables could also be set by entering a command such as `x = 24` into the command window.)

Finally, in line 0136, the `remmenu` command is used to remove any existing user menu, and then, in line 0137, the `addmenu` command is used to show the newly defned menu.

GM
May 2013

Listing

```
0001   $ ========================================================================
0002   $  SOLVE3_SIMPLE.MAC
0003   $ ========================================================================
0004   $  Solving three simultaneous linear equations
0005   $  GM October 1997
0006   $  Revised May 2013
0007   $ ========================================================================
0008
0009   dec real   x, y, z;                                $ main (global) variables
0010   dec geom   l0, l1, l2, l3, l4, l5;                 $ geom objects for lines
0011   dec geom   tx, ty, tz, tt;                         $ geom objects for text
0012   dec geom   t1, t2, t3;                             $ geom objects for text
0013   dec string stemp;                                  $ temporary char string
0014
0015   graphics();                                        $ create graphics window
0016
0017   x = 0;                                             $ set variables to zero
0018   y = 0;
0019   z = 0;
0020
0021   l0 = lin( -2, -6, 0, 14, -6, 0 );                  $ define the lines
0022   l1 = lin( -2, -2, 0, 14, -2, 0 );
0023   l2 = lin( -2,  6, 0, 14,  6, 0 );
0024   l3 = lin( -2, 14, 0, 14, 14, 0 );
0025   l4 = lin( 14, -6, 0, 14, 14, 0 );
0026   l5 = lin( -2, -6, 0, -2, 14, 0 );
0027   ccol( green(), l0, l1, l2, l3, l4, l5 );           $ set colour to green
0028
0029   swrite( stemp, "x + y - 8*z + 3" );                $ define text string
0030   t1 = txt( 0, 12, 0, stemp );                       $ assign to text variable
0031   ccol( yellow(), t1 );                              $ set colour to yellow
0032   swrite( stemp, "x + 2*y + z - 9" );                $ ditto
0033   t2 = txt( 0, 10, 0, stemp );                       $ ditto
0034   ccol( yellow(), t2 );                              $ ditto
0035   swrite( stemp, "x - y + 2*z - 1" );                $ ditto
0036   t3 = txt( 0,  8, 0, stemp );                       $ ditto
0037   ccol( yellow(), t3 );                              $ ditto
0038
0039   function show_values                               $ start of function
0040   {
0041      swrite( stemp, "x = %8.4lf     ", x );          $ define text string
0042      tx = txt( 0, 4, 0, stemp );                     $ assign to text variable
0043      ccol( rgb(1,1,0), tx );                         $ change colour to yellow
0044
0045      swrite( stemp, "y = %8.4lf     ", y );          $ ditto
0046      ty = txt( 0, 2, 0, stemp );                     $ ditto
0047      ccol( rgb(1,1,0), ty );                         $ ditto
0048
0049      swrite( stemp, "z = %8.4lf     ", z );          $ ditto
0050      tz = txt( 0, 0, 0, stemp );                     $ ditto
```

Figure 2: Listing of macro solve3_simple.mac (part 1)

```
0051      ccol( rgb(1,1,0), tz );                    $ ditto
0052
0053      rpnt();                                    $ clear and repaint screen
0054   }                                             $ end of function
0055
0056   function show_truth                           $ start of function
0057   {
0058      swrite( stemp, "Tr = ", truth() );         $ define text string
0059      tt = txt( 0, -4, 0, stemp );               $ assign to text variable
0060      ccol( red(), tt );                         $ change colout to red
0061      rpnt();                                    $ clear and repaint screen
0062   }                                             $ end of function
0063
0064   function reset                                $ start of function
0065   {
0066      x = 0;                                     $ set value to zero
0067      y = 0;                                     $ ditto
0068      z = 0;                                     $ ditto
0069      show_values();                             $ call function
0070      zoom();                                    $ zoom to fill screen
0071   }                                             $ end of function
0072
0073   function solve                                $ start of function
0074   {
0075      var       x, y, z;                         $ allow x, y, z to change
0076
0077      rule( x + y - 8*z + 3 );                   $ constraint rull
0078      rule( x + 2*y + z - 9 );                   $ constraint rull
0079      rule( x - y + 2*z - 1 );                   $ constraint rull
0080
0081      show_values();                             $ call function
0082   }                                             $ end of function
0083
0084   reset();                                      $ call function
0085   rpnt();                                       $ clear and repaint screen
0086
0087   menu solve_menu                               $ start menu defintion
0088   {
0089      button Reset                               $ new button
0090      { reset();
0091      }
0092      button Solve                               $ new button
0093      { solve();                                 $ call function
0094        show_values();                           $ call function
0095        show_truth();                            $ call function
0096      }
0097      submenu Fix/free>                          $ new submenu
0098      { button Fix x                             $ new button
0099        { fix(x);                                $ fix the variable x
0100        }
```

Figure 3: Listing of macro solve3_simple.mac (part 2)

```
0101        button Fix y                                $ new button
0102        { fix(y);                                   $ fix the variable y
0103        }
0104        button Fix z                                $ new button
0105        { fix(z);                                   $ fix the variable z
0106        }
0107        button Free x                               $ new button
0108        { free(x);                                  $ free the variable y
0109        }
0110        button Free y                               $ new button
0111        { free(y);                                  $ free the variable y
0112        }
0113        button Free z                               $ new button
0114        { free(z);                                  $ free the variable z
0115        }
0116      }
0117      submenu Set>                                  $ new submenu
0118      { button Set x                                $ new button
0119        { x = val( "New value for x", x );          $ redefine x
0120          show_values();                            $ call function
0121        }
0122        button Set y                                $ new button
0123        { y = val( "New value for y", y );          $ redefine y
0124          show_values();                            $ call function
0125        }
0126        button Set z                                $ new button
0127        { z = val( "New value for z", z );          $ redefine z
0128          show_values();                            $ call function
0129        }
0130      }
0131    }
0132
0133    remmenu();                                      $ remove any previous menu
0134    addmenu( solve_menu );                          $ show the new menu
0135
0136    $ End of file
0137
```

Figure 4: Listing of macro solve3_simple.mac (part 3)