Macro: lfit_simple.mac

Macro	lfit_simple.mac
Description	Demonstrates fitting a line to pass between two given points
CM version	Any

What the macro does

This macro attempts to fit a straight line between two given points. Figure 1 shows three stages in the process. On the left is the initial state showing the two points and the line. On the right is the final result with the line having moved between the points.

Two constraint rules are applied. One says that one end of the line has to lie at one of the points; the other says that the other end must lie at the other point. The constraint modeller tries to resolve these constraints by adjusting the position and rotation of the line. The middle part of figure 1 shows various trial positions as the modeller attempt the resolution.

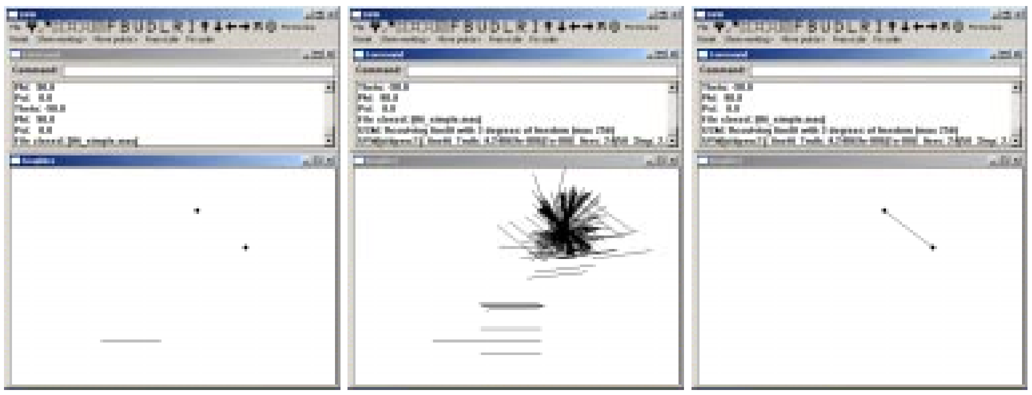

Figure 1: Graphics screen before, during and after fitting the line

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.

At the beginning of the macro (lines 0009–0010) a number of global variables are declared. Of these, `mm` is a "model space" and this is effectively a transformation. The two points are `p1` and `p2` and the line is `ll`.

A user-defined function (UDF) called `setup` is defined in lines 0012–0022. Line 0014 defines the model space. This is a two-dimensional model space. Entities can be embedded into such a space. The space has a transformation associated with it. This combines translation in the x- and y-directions, and a rotation about the z-axis (the axis perpendicular to the plane). It is also possible to define a global scale factor for the model space. Initially the scale factor is unity. The command at line 0014 defined the model space `mm` with zero translation in x and y (the first two arguments of the `mod2` function), and zero angle of rotation (the third argument). The two points are defined in lines 0015 and 0016. Then the line `ll` is defined. The arguments of to the `lin` function are the (x, y, z)-coordinates of its two end-points. These are chosen slightly strangely, but this is so that the line initially appears away from the two points. The line passes between the points $(-8, -3, 0)$ and $(-3, -3, 0)$; thus is has length 5 units. The final argument to the `lin` is the model space `mm`. This means that the line is embedded in this model space and when the model space is changed, the line moves. Lines 0018–0021 of the macro change the "font" of the points so that they appear as dots on the screen, and the colours of these points and of the line.

Next in the macro file comes the function `linefit` (lines 0025–0034). This creates two constraint rules (lines 0031 and 0032). These use the function `on` which is a built-in function of the modeller. It finds the distance (in world space) between two geometric objects. The first rule gives the expression which is the distance between the first end-point of line `ll` (denoted by `ll:e1`) and the point `p1`. If this distance is zero, then the end of the line lies at the point. Similarly the second rule expression is for the distance between `ll:e2`, the second end-point of the line, and the point `p2`.

The modeller needs to know what it can change when trying to resolve the constraints. This is done with the `var` list (line 0027). This only gives the model space `mm`. There are several components to this model space, but, by default, the only ones which are free are the two components of translation and the angle of rotation.

The function `linefit` takes one argument when it is called. This is specified by the `inp` statement (line 0029) which says that there is one argument which is called `code` within the function and is an integer as specified in line 0028. This variable is used with the `rpnt` function in line 0033. If `code` is 1, then `rpnt` clears the screen and repaints it; if it is zero, then the screen is repainted without clearing the screen; and if it is -1, then the `rpnt` function does nothing.

The `linefit` function could be invoked by entering the following command in the command window.

```
linefit( 0 )
```

However, to help the user, a menu is created.

How the rest of the macro works

Lines 0037–0081 create the menu with its buttons and submenus.

The submenu `Show working` (lines 0047–0054) calls `linefit` with the argument set to zero and to unity. The former allows all the attempts at repositioning the line to be shown (as in the middle of figure 1, but this tends to slow down the resolution process. The latter again shows all the attempts but as an animation.

Lines 0055–000072 enable the user to change the positions of the two points. Here the built-in function `dodig` is used. This gives a prompt to the user and then waiting for the user to move the cursor and click one of the mouse buttons. The function `getdig` retrieves the position of the cursor and uses this to redefine the appropriate point. As the point has been redefined, its colour and font have to be redefined.

Initially the modeller succeeds in fitting the line between the original points. This is because the points are 5 units apart which is also the length of the line. Once the points are moved, the modeller places the line midway between the points. The model space `mm` also has a component called `mm:sigma` which represents a (global) scale factor. Allowing this to vary enables the line to be stretched or shrunk. Lines 0073–0081 allow the user to free or fix this scale factor.

In line 0084, the `remmenu` command is used to remove any existing user menu, and then, in line 0085, the `addmenu` command is used to show the newly defined menu.

After this, the `setup` function is called to define the initial positions of the geometric objects. The the graphics window is created and the screen repainted. The `zoom` command repaints the graphics at full size for the screen, and the second `zoom` command shrinks it to a somewhat smaller size.

GM
May 2013

Listing

```
$ =========================================================================
$   LFIT_SIMPLE.MAC
$ =========================================================================
$   Fitting a line between two points
$   October 1997
$   Revised: May 2013
$ =========================================================================

dec  mod2    mm;                               $ global variables
dec  geom    p1, p2, ll;

function  setup                                $ start of function
{
   mm = mod2( 0, 0, 0 );                       $ define model space
   p1 = pnt( 4, 0, 0 );                        $ define first point
   p2 = pnt( 0, 3, 0 );                        $ define second point
   ll = lin( -8, -8, 0, -3, -8, 0, mm );       $ define the line
   cfont( 6, p1, p2 );                         $ points are dots
   ccol( green(), ll );                        $ line is green
   ccol( red(), p1 );                          $ first point is red
   ccol( yellow(), p2 );                       $ second point is yellow
}                                              $ end of function


function linefit                               $ start of function
{
   var  mm;                                    $ var list: just mm
   dec  int code;                              $ declare local integer
   inp  code;                                  $ use it as argument

   rule( ll:e1 on p1 );                        $ first constraint rule
   rule( ll:e2 on p2 );                        $ second constraint rule
   rpnt( code );                               $ repaint the screen
}                                              $ end of function


menu lfit_menu                                 $ start new menu
{
   button Fit the line                         $ define button
   { linefit(-1);                              $ call linefit function
     rpnt(1);                                  $ clear and repaint screen
   }
   button Reset                                $ define button
   { setup();                                  $ call setup function
     rpnt(1);                                  $ clear and repaint screen
   }
   submenu Show working>                       $ define submenu
   { button Overlay                            $ define button
     { linefit(0);                             $ call linefit function
     }
```

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

```
     button Animate                            $ define button
     { linefit(1);                             $ call linefit fucntion
     }
   }
   submenu Move points>                        $ define submenu
   {  button Red point                         $ define button
      { if( dodig( "Give new red point" ) )    $ go into digitize mode
        { p1 = getdig();                       $ get digitize, redefine p1
          ccol( red(), p1 );                   $ reset colour of p1
          cfont( 6, p1 );                      $ and its font
          rpnt(0);                             $ repaint without clearing
        }
      }
      button Yellow point                      $ define button
      { if( dodig( "Give new yellow point" ) ) $ go into digitize mode
        { p2 = getdig();                       $ get digitize, redefine p2
          ccol( yellow(), p2 );                $ reset colour of p2
          cfont( 6, p2 );                      $ and its font
          rpnt(0);                             $ repaint without clearing
        }
      }
   }
   button Free scale                           $ define button
   { free( mm:sigma );                         $ free global scale in mm
     fwriteln( 0, "Scale freed" );             $ output message to screen
   }
   button Fix scale                            $ define button
   { fix( mm:sigma );                          $ fix global scale in mm
     fwriteln( 0, "Scale fixed" );             $ output message to screen
   }
}

remmenu();                                     $ remove any previous menu
addmenu( lfit_menu );                          $ add new menu

setup();                                       $ call setup function

graphics();                                    $ create graphics window
rpnt(1);                                       $ clear and repaint
zoom();                                        $ zoom all
zoom( 0.75 );                                  $ and zoom down a little

$  End of file

```

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