Macro: slider_crank.mac

Macro	slider_crank.mac
Description	Demonstrates a slider-crank mechanism (four bar mechanism)
CM version	Any
See also	macro: crank_rocker.mac

What the macro does

This macro creates a “stick diagram” of a slider-crank mechanism (a type of four bar mechanism). This means that the links are represented by simple lines. The crank and coupler links are just a single line each. The slider is represented by a single point. These links are shown on the left in figure 1. Here the links are assembled correctly by the modeller. On the right of the figure is the result of rotating the crank to simulate the motion.

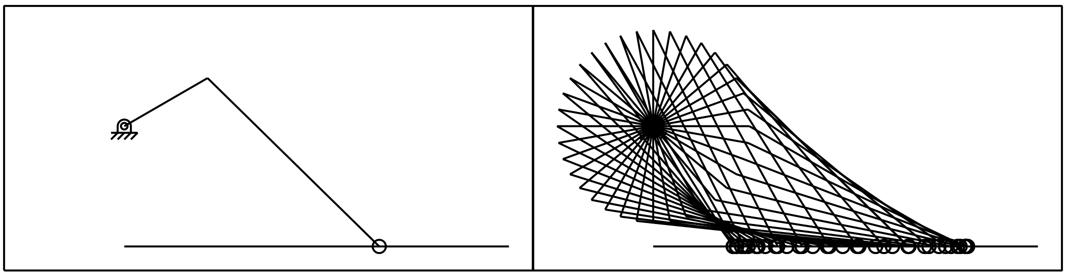

Figure 1: Slider-crank (four bar) mechanism

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.

A number of geometric objects are used.

`p0`	fixed pivot point
`lcrank`	line to represent the crank
`lcoupler`	line to represent the coupler
`pcoupler`	point at the end of the coupler (i.e. the slider)
`lslider`	line to represent the slide rail

These objects are declared as global variables at the start of the macro (lines 0012–0014) and are defined in function `setup` (lines 0032–0049). The definition is in terms of real variables declared in lines 0016 and 0017, and given values in lines 0025–0028. Each line goes between two end-points; for example, the first end-point of line `lcrank` is denoted by `lcrank:e1`, and the second end-point by `lcrank:e2`.

Line `lslider` represents the slider rail and does not move. Lines `lcrank` and lcoupler represent the moving links of the mechanism. Each moving line is embedded in a model space. A model space is essentially a transformation with which a number of geometric entities can be associated. If the transform changes, then the entities move together.

Two model spaces are used. These are declared in line 0015 and are defined in the `setup` function in lines 0037 and 0039. The translation components of each model space are initially zero (first two argument in each use of function `mod2`). The rotation angles are set to non-zero values: this is mainly for convenience here so that the objects appear rotated on the screen and so are easier to identify.

Note that models space `mcoupler` is embedded into model space `mcrank` by including `mcrank` as the fourth argument in line 0039. This means that if `mcrank` moves, the `mcoupler` moves with it. A simple hierarchy of model spaces has been constructed as shown in figure 2.

In the definitions of the geometric objects in the `setup` function, it is seen that the moving lines and the point `pcoupler` are each embedded in an appropriate space. The fixed line, `lslider`, is placed in world space.

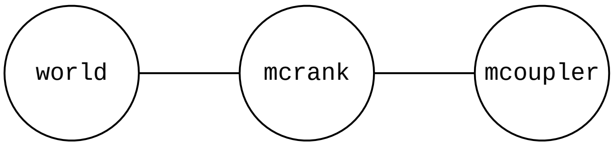

Figure 2: Hierarchy of model spaces

Initially the geometric objects are not connected. The `setup` function uses the `pivot` function twice in lines 0047 and 0048. The first of these makes an adjustment to the translation part of model space `mcrank`. This is to bring `lcrank:e1`, the first end-point of line `lcrank` onto `p0`, the pivot point. The second adjusts model space `mcoupler` to bring `lcoupler:e1`, the first end-point of line `lcoupler` onto `lcrank:e2`, the second end-point of line `lcrank`.

What each use of `pivot` does is to fix the translation components of the relevant model space. All that can change is its rotation angle.

It remains to put the end of the coupler onto the slide rail. A constraint rule is introduced. This is done in the `assemble` function (lines 0052–0057). The constraint rule says that point `lcoupler:e2` needs to be on line `lslider`. In line 0056, the `on` function finds the distance between these objects. The rule is becomes true when this distance is zero.

The `var` list (line 0054) just contains `mcoupler`. Since the translation components of each of this are fixed by one of the `pivot` commands, all that can be changed is the rotation angle. Hence there is just one degree of freedom that can be used to resolve the constraint rule.

The `cycle` function is used to simulate a cycle of the mechanism running. This function is defined in lines 0060–0074. Within a loop, the rotation angle of space `mcrank` is repeatedly incremented (line 0066) and the mechanism is reassembled (line 0067). To track the end of the coupler, the point `pcoupler` is transformed into world space (by the `transf` function, line 0069). The result is assigned to a member of a global array called `qq` (declared in line 0018). Its colour and font are then set and the screen is repainted (line 0071). The argument for th `rpnt` function is provided by the argument passed in to the `cycle` function. The x-coordinated of the tracked point is obtained and stored in the array `xx` (line 0072).

Function `find_vel_acc` is defined in lines 0077–0081. This uses the built-in `deriv` function to evaluate numerically the first and second derivatives of the x-motion held in array `xx`. Note that the `cycle` function needs to have been run before these derivatives can be found. The first argument of the `deriv` function is either 1 or 2 depending on whether the first or second derivative is required. The third argument represents the time step between points in the array, `xx`, being differentiated.

The results can be output to a text (ASCII) file using the function `do_output` defined in lines 0084–0107. Line 0089 opens the file using the built-in function `fopen`. This is on "channel" 2 as specified by the first argument to `fopen`. The second argument, also 2, specifies that the file is to be opened for writing. For convenience, lines 0090 and 0091 output (to the file, channel 2) the name of the file and the current date and time. Again for convenience, each of these output lines starts with a dollar symbol. Since a dollar symbol in the macro is interpreted as starting a comment, the symbol is output using `asc(36)` which produces the character whose ASCII value is 36. The function `fwriteln` makes the required output and then goes to a new line. Line 0092 simply outputs a blank line.

The results are output using a loop (lines 0094–0100). For convenience, the `fwrite` function is used to output values individually. This makes the output without going to a new line afterwards. After the values are output, line 0099 forces the start of a new line. In each of lines 0095–0098, the character string represents a formatting string as used in the C language. The modeller interprets any string in an output command which begins with the percent symbol as representing a formatting string to control the output of the next argument. In line 0095, the string `%3d` asks for the integer value to be output in a field of size 3. In lines 0096–0098, the string `%12.5lf"` asks for the real number to be output in a field of size 12 using 5 decimal places.

Finally in the macro, the initial set-up is made and a simple menu is created.

GM
May 2013

Listing of macro

```
$ =======================================================================
$   slider_crank.mac
$ =======================================================================
$   Slider crank mechanism
$   revised: May 2013
$ =======================================================================

dec int   npoint;                                   $ number of points
npoint = 36;                                        $ make it 36

dec string file_name;                               $ name of output file
dec geom   p0;                                      $ fixed pivot point
dec geom   lcrank, lcoupler, lslider;               $ lines for links/rail
dec geom   pcoupler;                                $ end of coupler point
dec mod2   mcrank, mcoupler;                        $ model spaces
dec real   len_crank, len_coupler, len_slider;      $ link lengths
dec real   yoffset;                                 $ offset value
dec geom   qq[npoint];                              $ array of points
dec real   xx[npoint];                              $ array of pos
dec real   vv[npoint];                              $ array of vel
dec real   aa[npoint];                              $ array of acc
dec real   tstep;                                   $ time step

file_name = "slider_crank.out";                     $ set the file name
len_crank = 4;                                      $ length of crank
len_coupler = 10;                                   $ length of coupler
len_slider = 16;                                    $ length of slide rail
yoffset = -5;                                       $ offset of rail
tstep = 0.1;                                        $ time step


function setup                                      $ start of function
{
   p0 = pnt(0,0,0);                                 $ define point p0
   cfont(7,p0);                                     $ change its font
   ccol(blue(),p0);                                 $ and colour
   mcrank = mod2(0,0,30);                           $ crank model space
   lcrank = lin( 0,0,0, len_crank,0,0, mcrank );    $ crank line
   mcoupler = mod2(0,0,-45,mcrank);                 $ coupler model space
   lcoupler = lin(0,0,0,len_coupler,0,0,mcoupler);  $ coupler line
   pcoupler = pnt( len_coupler, 0, 0, mcoupler );   $ end of coupler point
   lslider = lin(0,yoffset,0,len_slider,yoffset,0); $ line for slide rail
   ccol( red(), lcrank );                           $ make crank red
   ccol( green(), lcoupler, pcoupler );             $ and coupler green
   ccol( magenta(), lslider );                      $ and rail magenta
   cfont( 4, pcoupler );                            $ make font a circle
   pivot( mcrank, lcrank:e1, p0 );                  $ join crank to p0
   pivot( mcoupler, lcoupler:e1, lcrank:e2 );       $ and crank to coupler
}                                                   $ end of function

```

Figure 3: Listing of macro slider_crank.mac (part 1)

```
function assemble                                      $ start of function
{
   var mcoupler;                                       $ coupler angle varies

   rule( pcoupler on lslider );                        $ put coupler on rail
}                                                      $ end of function


function cycle                                         $ start of function
{
   dec int i, code;                                    $ local variables
   inp code;                                           $ one argument

   loop( i, 0, npoint )                                $ loop for cycle
   { mcrank:a = i*360/npoint;                          $ increment crank angle
     assemble();                                       $ call assemble
     qq[i] = transf( pcoupler );                       $ get end of coupler
     ccol( cyan(), qq[i] );                            $ change colour
     cfont( 6, qq[i] );                                $ and its font
     rpnt(code);                                       $ repaint graphics
     xx[i] = qq[i]:x;                                  $ get x coordinate
   }
}                                                      $ end of function


function find_vel_acc                                  $ start of function
{
   vv = deriv( 1, xx, tstep );                         $ first derivative
   aa = deriv( 2, xx, tstep );                         $ second derivative
}                                                      $ end of function


function do_output                                     $ start of function
{
   dec int  i;                                         $ declare local int

   fwriteln( 0, "Opening file:", file_name );          $ message to screen
   fopen( 2, 2, file_name );                           $ open file to write
   fwriteln( 2, asc(36), "File:", file_name );         $ output file name
   fwriteln( 2, asc(36), "Date:", date() );            $ output date/time
   fwriteln( 2 );                                      $ blank line

   loop( i, 0, npoint )
   { fwrite( 2, "%3d", i );                            $ output counter
     fwrite( 2, "%12.5lf", xx[i] );                    $ output pos
     fwrite( 2, "%12.5lf", vv[i] );                    $ output vel
     fwrite( 2, "%12.5lf", aa[i] );                    $ output acc
     fwriteln( 2 );                                    $ end output line
   }
```

Figure 4: Listing of macro slider_crank.mac (part 2)

```

   fwriteln( 2 );                                   $ blank line
   fwriteln( 2, asc(36), "End of file" );           $ output end of file
   fwriteln( 2 );                                   $ blank line
   fclose( 2 );                                     $ close file
   fwriteln( 0, "File closed:", file_name );        $ write to screen
}                                                   $ end of function

graphics();                                         $ graphics window
setup();                                            $ call setup
assemble();                                         $ call assemble
rpnt();                                             $ repaint screen
zoom();                                             $ and zoom all
zoom(0.8);                                          $ zoom down a little

menu slider                                         $ create menu
{
   button Setup
   { setup();                                       $ call setup function
   }
   button Cycle
   { cycle(1);                                      $ call cycle function
   }
   button Vel/acc
   { find_vel_acc();                                $ find vel and acc
     fwriteln( 0, "Completed" );                    $ write to screen
   }
   button Output
   { do_output();                                   $ output values
   }
}

remmenu();                                          $ remove previous menu
addmenu( slider );                                  $ put up new menu

$ End of file

```

Figure 5: Listing of macro slider_crank.mac (part 3)

Listing of output file

```
$ File: slider_crank.out
$ Date: Thu May 30 08:50:42 2013

  0    12.66025    -4.02758   -19.61390
  1    12.15943    -5.95630   -18.96046
  2    11.46899    -7.76948   -17.30323
  3    10.60553    -9.36000   -14.50720
  4     9.59699   -10.60458   -10.38435
  5     8.48461   -11.35839    -4.69190
  6     7.32532   -11.45595     2.74083
  7     6.19342   -10.74258    11.52654
  8     5.17680    -9.17265    19.87212
  9     4.35890    -6.94595    24.66177
 10     3.78761    -4.50816    24.09400
 11     3.45726    -2.31147    19.83984
 12     3.32532    -0.57476    14.89444
 13     3.34231     0.71661    10.93295
 14     3.46864     1.67507     8.23629
 15     3.67733     2.41404     6.54295
 16     3.95145     3.01814     5.53914
 17     4.28095     3.54402     4.97835
 18     4.66025     4.02761     4.69362
 19     5.08648     4.49062     4.56646
 20     5.55837     4.94406     4.50241
 21     6.07529     5.39001     4.41649
 22     6.63637     5.82200     4.22339
 23     7.23969     6.22487     3.83398
 24     7.88135     6.57449     3.15842
 25     8.55459     6.83820     2.11577
 26     9.24899     6.97643     0.64891
 27     9.94987     6.94593    -1.25902
 28    10.63817     6.70437    -3.57206
 29    11.29075     6.21587    -6.19792
 30    11.88135     5.45621    -8.99537
 31    12.38199     4.41691   -11.79063
 32    12.76473     3.10751   -14.39729
 33    13.00349     1.55593   -16.63446
 34    13.07591    -0.19276   -18.33916
 35    12.96494    -2.07830   -19.37171
 36    12.66025    -4.02758   -19.61390

$ End of file
```

Figure 6: Listing of output file