

%(\*)(1) *for t=0:0.02:4*

This is the start of a for-loop (the end of the loop is denoted by a matching end-statement). The variable is the loop-variable and is incremented once in each loop.

In, for example, *t=a:b:c*, then *a* is the initial value of *t*, the value that *t* takes in the first pass of the loop. The value *c* is the final value that *t* takes and is used in the final loop. The value *b* is the incrementing amount of the loop variable *t*.

When we have *t=0:0.02:4*, then *t* takes the values,

There are 200 separate values in all.

%(\*)(2) *plot([0 0 3.5],[0 0 2],’k:’)*

This function plots two lines. A vertical line from the point (0,0) to the point (0,2) and a horizontal line from the point (0,2) to the point (2,3.5). These lines will be dotted lines and of the colour black. The horizontal line represents the locus of one end of the red line.

%(\*)(3) *plot(cos(alpha),sin(alpha),’k’)*

The variable *alpha* is a vector of 200 equi-spaced values between 0 and . (See line 5 of simulate.m, above).

*cos(alpha)* represents an array of cosine values of all the individual values of the vector *alpha*. Similarly for *sin(alpha)* – an array of sine values.

The plot function then uses these *cos*- and *sin*-values to plot a circle, the line style of the circle being dotted and of colour black. This circle is the locus of the other end of the red line.

%(\*)(4) *if t==0*

This is an *if-statement*. The *if-statement* tests the logical expression , and will execute the following statement (the statement following the *if-statement*) if true. This means that if the value of *t* is equal (or equivalent) to the value of , then the following statement would be executed, otherwise it would not.

Should the *if-statement* include an *else-clause*, then the statement following the *else* would be executed if the tested expression was false.

The code means that initially there is a pause of 2 seconds during the first pass of the loop (when ) and thereafter, there is a pause of seconds during each successive pass of the loop.

%(\*\*)(1) *axis equal, axis manual*

*axis equal* is a scaling feature. It means that units on the *x*-axis are the same size as units on the *y*-axis. This would be a Unit aspect ratio of . Normally, Matlab would adjust the unit aspect ratio in order to fit in both ranges of *x*- and *y*-values within a standard-sized graph window. Without this happening (*axis equal*) the plotted circle would have ended up as an ellipse.

*axis manual* freezes graph scaling at the current limits so that any future plots are constrained to fit within these limits.

Without this line (*axis manual*), providing that we had *hold on,* then the graph’s scale would be extended to accommodate all plots.

Under the current circumstances, with the present circle, it would make no difference whatsoever if we did not have the line *axis equal* in our code since specifying values for axis limits also sets axis scaling to manual and the scale setting with axis([-1 3.5 -1.2 2.2]) provides limits to include all graphs.

%(\*\*)(2) *clf*

This clears the (current) figure window.

Without this command, running the m-script again would result in no plot output. The plot window would have had to have been deleted first of all.

Using the *clf* command saves having to delete the graph window between each run of the m-script.