3.3 More things to do with a computer, besides solving linear equations and ODEs

Mechanics is a physical subject. The concepts in mechanics do not depend on computers. But mechanics is also a quantitative subject; relevant amounts (of length, mass, force, moment, time, etc) are described with numbers, and relations are described using equations and formulas. Computers are very good with numbers and formulas. Thus, the modern practice of engineering mechanics uses computers. The most-needed computer skills for mechanics are:

- solution of simultaneous linear algebraic equations,
- plotting, and
- numerical solution of ODEs (Ordinary Differential Equations).

More basically, an engineer also needs the ability to routinely evaluate standard functions (x^3, $\cos^{-1}\theta$, *etc.*), to enter and manipulate lists and arrays of numbers, and to write short programs.

Classical languages, applied packages, and simulators

Programming in standard languages such as Fortran, Basic, C++, Java or Python probably takes too much time to use in solving simple mechanics problems. Thus an engineer needs to learn to use one or another widely available computational package (*e.g.,* MATLAB, SCI-LAB, OCTAVE, MAPLE, MATHEMATICA, MATHCAD, LABVIEW, etc). We assume that students have learned, or are learning such a package. Although none of the homework here depends on such, we also encourage you to play with packaged mechanics simulators (*e.g.,* INVENTOR, ADAMS, DADS, ODE, ALGODOO, etc) for testing and building your intuition.

How we explain computation

Solving a mechanics problem involves

1. Reducing a physical problem to a well posed mathematical problem;
2. Solving the math problem using some combination of pencil and paper and numerical computation; and
3. Giving physical interpretation of the mathematical solution.

This book is primarily about setup (a) and interpretation (c), neither of which particularly depends on what method is used to solve the equations. If a problem requires computation, the exact computer commands vary from package to package. Because we don't know which computer package you are using we show computer calculations using an informal pseudo-computer language. For reference, typical commands are summarized on page 4.

Required computer skills

Here, in a little more detail, are the primary computer skills you need.

- **Linear algebraic equations.** Many mechanics problems are statics or 'instantaneous mechanics' problems. These problems involve trying to find some forces or accelerations at a given configuration of a system. These problems can generally be reduced to the **solution of linear algebraic equations** of this general type: solve

$$\begin{aligned} 3\ x\ +\ \ 4\ y\ &=\ 8 \\ -7\ x\ +\ \sqrt{2}\ y\ &=\ 3.5 \end{aligned}$$

  for x and y. In practice the number of variables and equations can be quite large. Some computer packages will let you enter equations almost as written above. In our pseudo language we would write:

```
eqset =  {  3*x +       4*y = 8
           -7*x + sqrt(2)*y = 3.5  }
solve eqset for x and y
```

  Other packages may require you to set up your equations in matrix form

$$\underbrace{\begin{bmatrix} 3 & 4 \\ -7 & \sqrt{2} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{z} = \underbrace{\begin{bmatrix} 8 \\ 3.5 \end{bmatrix}}_{b} \quad \text{or} \quad Az = b$$

  which in computer-speak might look something like this:

```
A =  [ 3      4
      -7  sqrt(2) ]
b =  [ 8 3.5 ]'
solve  A*z=b for z
```

  where `A` is a 2×2 matrix, `b` is a column of 2 numbers (the `'` indicates that the row of numbers `b` should be transposed into a column), and the two elements of `z` are x and y. For systems of two equations, like above, a computer is hardly needed. But for systems of three equations, pencil and paper work is sometimes error-prone. Given the tedium, the propensity for error, and the availability of electronic alternatives, pencil and paper solution of four or more equations is an anachronism.

- **Plotting.** In order to see how a result depends on a parameter, or to see how a quantity varies with position or time, it is useful to see a **plot**. Any plot based on more than a few data points or a complex formula is far more easily drawn using a computer than by hand. Most often you can organize your data into a set of (x, y) pairs stored in an `x` list and a corresponding `y` list. A simple computer command will then plot x *vs* y. The pseudo-code below, for example, plots a circle using 100 points

```
npoints = [0 1 2 3 ... 100]
```

```
theta   = npoints * 2 * pi / 100
x       = cos(theta)
y       = sin(theta)
plot y vs x
```

where `npoints` is the list of numbers from 0 to 100, `theta` is a list of 100 numbers evenly spaced between 0 and 2π, and `x` and `y` are lists of 100 corresponding x, y coordinate points on a circle.

- **ODEs** The result of using the laws of dynamics is often a set of **o**rdinary **d**ifferential **e**quations which need to be solved. A simple example would be:

  Find x at $t = 5$ given that $\dfrac{dx}{dt} = x$ and that at $t = 0$, $x = 1$.

  The solution to this problem can be found easily enough by hand to be $x(5) = e^5$. But often the differential equations are just too hard for pencil and paper solution. Fortunately the **numerical solution of Ordinary Differential Equations** (ODEs) is already programmed into scientific and engineering computer packages. The simple problem above is solved with computer code equivalent to these informal commands:

  ```
  ODES =  { xdot  = x }
  ICS  =  { xzero = 1 }
  solve ODES with ICS until t=5
  ```

  which will yield a list of values for paired values for t and x the last of which will be $t = 5$ and x close to $e^5 \approx 148.4$.

Examples of informal computer commands

We use informal computer commands that are not as strict as any real computer package. You will translate informal commands, like those below, into commands your package understands. This reference table uses mathematical ideas which you may or may not know before you read this book, but these will be introduced in the text when needed.

`x=7`	Set the variable x to 7.
`omega=13`	Set ω to 13.
`u=[1 0 -1 0]` `v=[2 3 4 pi]`	Define u and v to be the lists shown.
`t= [.1 .2 .3 ... 5]`	Set t to the list of 50 numbers implied by the expression.
`y=v(3)`	sets y to the third value of v (in this case 4).
`A=[1 2 3 6.9` `5 0 1 12 ]`	Set A to the array shown.
`z= A(2,3)`	Set z to the element of A in the second row and third column.

`w=[3`
`4`
`2`
`5]`
	Define w to be a column vector.

`w = [3 4 2 5]'`	Same as above. ' means transpose.
`u+v`	Vector addition. In this case the result is $[3\ 3\ 3\ \pi]$.
`u*v`	Element by element multiplication, in this case $[2\ 0\ -4\ 0]$.
`sum(w)`	Add the elements of w, in this case 14.
`cos(w)`	Make a new list, each element of which is the cosine of the corresponding element of $[w]$.
`mag(u)`	The square root of the sum of the squares of the elements in $[u]$, in this case 1.41421...
`u dot v`	The vector dot product of component lists $[u]$ and $[v]$, (we could also write `sum(u*v)`.
`C cross D`	The vector cross product of $\vec{\boldsymbol{C}}$ and $\vec{\boldsymbol{D}}$, assuming the three element component lists for $[C]$ and $[D]$ have been defined.
`A matmult w`	Use the rules of matrix multiplication to multiply $[A]$ and $[w]$.
`eqset = {3x + 2y = 6` `6x + 7y = 8}`	Define 'eqset' to stand for the set of 2 equations in braces.
`solve eqset` `for x and y`	Solve the equations in 'eqset' for x and y.
`solve Ax=b for x`	Solve the matrix equation $[A][x] = [b]$ for the list of numbers x. This assumes A and b have already been defined.

`for i = 1 to N`
`such and such`
`end`
	Execute the commands 'such and such' N times, the first time with $i = 1$, the second with $i = 2$, etc

`plot y` *vs* `x`	Assuming x and y are two lists of numbers of the same length, plot the y values *vs* the x values.

`solve ODEs`
`with ICs`
`until t=5`
	Assuming a set of ODEs and ICs have been defined, use numerical integration to solve them and evaluate the result at $t = 5$.

With an informality consistent with what is written above, other commands will be introduced as needed.