TOOLS IN MODELING: BASICS

Overview

- Equations of Motion
- Ordinary Differential Equations
- Programming Environments
    - FORTRAN
    - MAPLE
    - MATLAB
- Programming Issues
- Example
- Solution of Ordinary Differential Equations
    - Analytical Solutions
    - Numerical Solutions
    - Stiff Systems

“As I wrote many years ago at the very beginning of the debate about computers, a computer is just a glorified pencil. Einstein once said "my pencil is cleverer than I". What he meant could perhaps be put thus: armed with a pencil, we can be more than twice as clever as we are without. Armed with a computer (a typical World 3 object), we can perhaps be more than a hundred times as clever as we are without; and with improving computers there need not be an upper limit to this.”

Karl Popper, The Self and Its Brain, p. 208

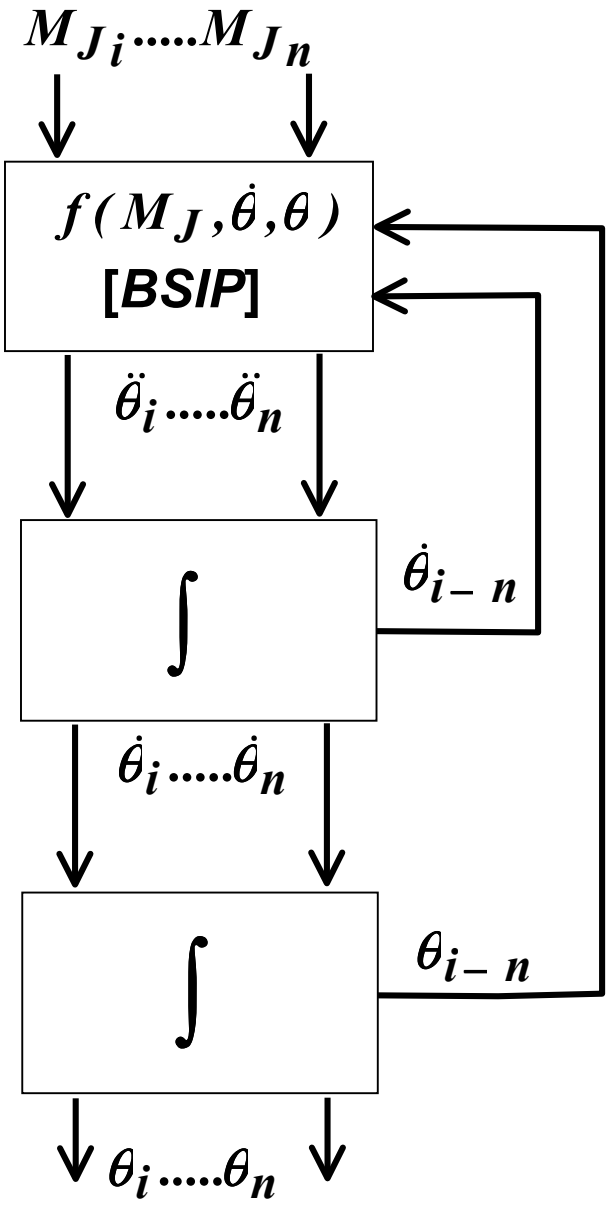
EQUATIONS OF MOTION

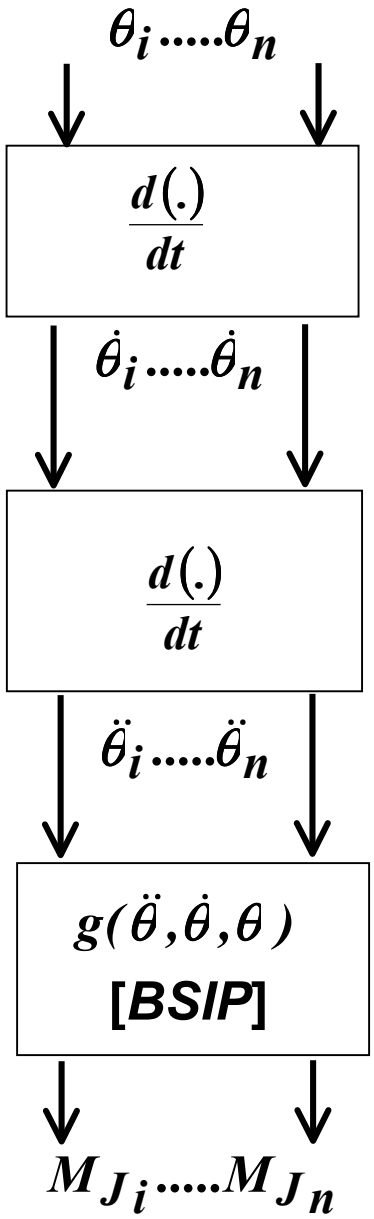
Equations of Motion – set of mathematical equations which describe the forces and movements of a body.

Inverse Dynamics – starting from the motion of the body determines the forces and moments causing the motion.

Direct Dynamics – starting from the forces and moments acting on a body determines the motion arising from these forces and moments. (Also called forward dynamics.)

EQUATIONS OF MOTION





ORDINARY DIFFERENTIAL EQUATIONS

Many problems end up in the form of an ordinary differential equation (ODE). A typical format for an ODE is

$$\frac{dx(t)}{dt} = -\alpha x(t)$$

Any problem involving an ODE is not completely specified by the equation, you require knowledge of boundary conditions. This knowledge could be in the form of additional functions or numerical values.

Initial Value Problem – when inputs and outputs are known at some specified point.

Two-point Boundary Value Problem – when inputs and outputs to the ODE are known at two points.

PROGRAMMING ENVIRONMENTS
FORTRAN

FORTRAN – this is an acronym for ***FOR***mula ***TRAN***slation. It is a high level programming language designed for scientific and mathematical purposes. It is written in a combination of algebraic formulae and English statements.

Syntax is not dissimilar between programming environments (e.g. Pascal, C, etc.), although all have their own peculiarities.

The one to use depends as much on availability as the application and local expertise.

PROGRAMMING ENVIRONMENTS
FORTRAN

$$R = \frac{V^2}{g}.\cos\theta.\left[\sin\theta + \left(\sin^2\theta + \frac{2.g.h}{V^2}\right)^{1/2}\right]$$

In FORTRAN this equation becomes

```
C  Routine to find distance of a projectile
      V2 = V * V
      CT = COS(theta)
      ST = SIN(theta)
C
      R= ( V2 / g ) * CT *
     +     (ST + (ST**2 + (2. * g * h)/ (V2) )**0.5)
C
      WRITE(*,*) 'Distance traveled by projectile was'
      WRITE(*,100) R
100 FORMAT( F8.3, 'meters')
```

PROGRAMMING ENVIRONMENTS
MAPLE

MAPLE – is a general purpose symbolic computation program. (Others include MATHMATICA, DERIVE)

It can perform a number of mathematical functions, for example

>expr:=x^4/y;

$$expr := \frac{x^4}{y}$$

>diff(expr,x);

$$4\frac{x^3}{y}$$

>diff(expr,x,x);

$$12\frac{x^2}{y}$$

PROGRAMMING ENVIRONMENTS
MAPLE

```
>a:=(x^2-4)/(x+2);
```

$$a := \frac{x^2 - 4}{x + 2}$$

```
>simplify(a):
```

$$x - 2$$

It will also factor, expand, numerically differentiate or integrate, output computer program code, etc.

PROGRAMMING ENVIRONMENTS
MATLAB

MATLAB – this is an acronym for ***MAT***rix ***LAB***oratory. As with FORTRAN it is a high level programming language designed for scientific and mathematical purposes. It is written in a combination of algebraic formulae and English statements.

Example MATLAB

```
>a=[ 2 2 4; 1 2 3; 1 3 6];
>b=[ 6 7 8; 3 5 7; 8 9 9];
>c=a*b;
>a,b,c
```

```
a =     2     2     4
        1     2     3
        1     3     6
b =     6     7     8
        3     5     7
        8     9     9
c =    50    60    66
       36    44    49
       63    76    83
```

PROGRAMMING ENVIRONMENTS
MATLAB

It does much more than handle matrices. It has two principle advantages over FORTRAN, one is that it comes with many built in routines (matrix inversion, solution of ODEs, etc.), and that sizes of data arrays do not have to declared.

Format for solving an ODE.

[time, x] = ode45('odefunc',[0 2], [11])

Where

ode45 – is a ODE solver which used a combination of 4th and 5th order Runge-Kutta formulas

'odefunc' – is user defined ODE

[0 2] - is the time over which the integration should take place

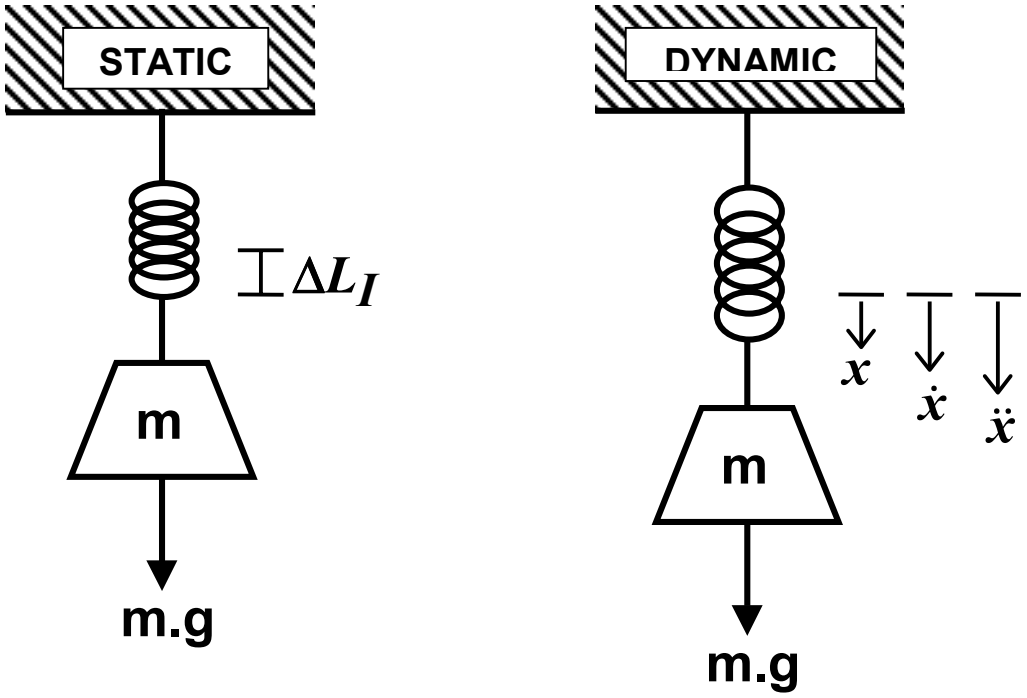
[11] – is the initial value.

PROGRAMMING ISSUES

- Portability
- Machine Precision
- Code interpretation
- Time
- Routine Availability

[Do not ignore other packages for example Excel or Minitab.]

EXAMPLE



A mass suspended from a mass-less spring, the equation for a linear spring is

$$F = \Delta L . k$$

Where

F - force applied to spring

ΔL - change in length of spring

k - stiffness of spring

1) Static Equilibrium

$$F = m . g = \Delta L . k$$

EXAMPLE

2) IF IN MOTION

$$m.\ddot{x} = \sum F = mg - k(\Delta L + x)$$

As $m.g = \Delta L.k$, then the equation becomes

$$m.\ddot{x} = - k.x \qquad \textbf{[An ODE]}$$

3) SOME RE-ARRANGEMENT

$$m.\ddot{x} = -kx \Rightarrow \ddot{x} = \frac{- kx}{m}$$

$$\ddot{x} - \frac{- kx}{m} = 0 \Rightarrow \ddot{x} + \frac{kx}{m} = 0$$

EXAMPLE

4) SUBSTITUTION

$$\omega_n^2 = \frac{k}{m}$$

$$\ddot{x} + \omega_n^2 . x = 0$$

5) ANALYTICALLY INTEGRATE

Initial conditions $x(0), \dot{x}(0)$

$$x = \frac{\dot{x}(0)}{\omega_n} \sin \omega_n . t + x(0) \cos \omega_n . t$$

So we have a function which describes the motion of a mass bouncing on a spring.

$$x = A . \sin \omega_n . t + B . \cos \omega_n . t$$

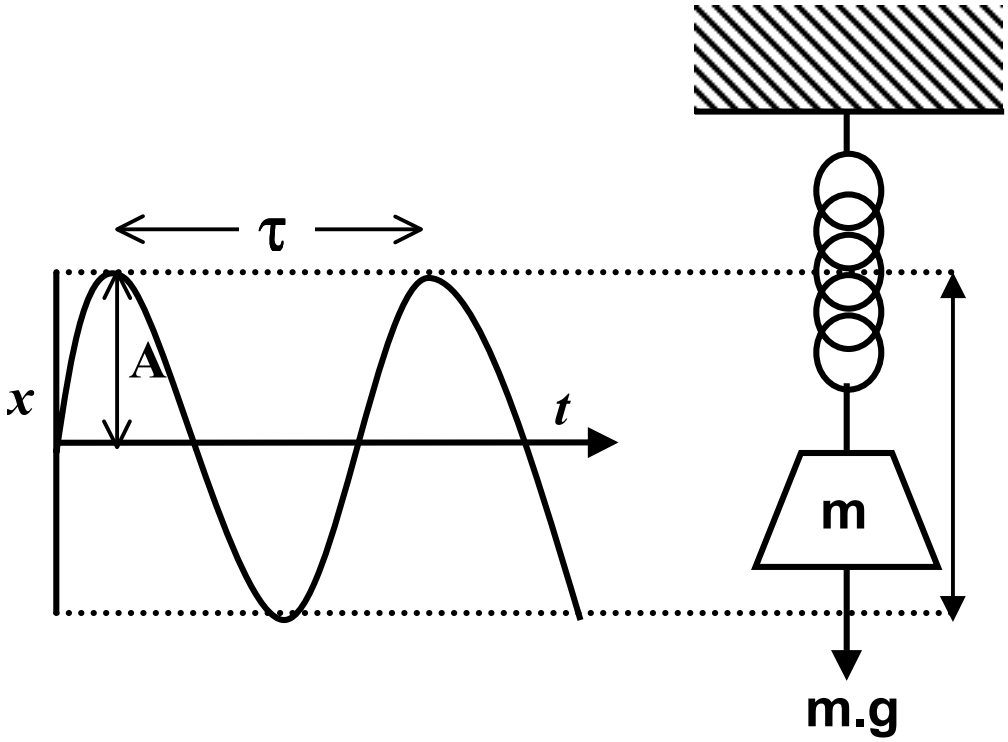
EXAMPLE

Given suitable initial conditions the equation can be reduced to

$$x = A.\sin \omega_n .t$$

Note

$$\omega_n = \frac{2.\pi}{\tau}$$



SOLUTION OF ODES
ANALYTICAL SOLUTIONS

Given an ODE it is sometimes possible to analytically solve the equation, the alternative is a numerical solution in theory they should both give the same results. In practice this is not the case due to rounding errors. Whichever solution is used, both require initial values before they can effect a solution.

Analytical

- By hand
- Using a package such as MAPLE

Numerical

- Writing a routine
- Using a routine

SOLUTION OF ODES
ANALYTICAL SOLUTIONS

Take the ordinary differential equation of earlier

$$\frac{dx}{dt} = -\alpha x \Rightarrow \frac{dx}{dt} + \alpha x = 0$$

In MAPLE this can be solved analytically

```
>eq:=diff(x(t),t)+alpha*x(t)=0;
```

$$eq := \left(\frac{\partial}{\partial t} x(t) \right) + \alpha x(t) = 0$$

```
>sol:=dsolve({eq,x(0)=x0},x(t));
```

$$sol := x(t) = e^{(-\alpha t)} x0$$

MAPLE is a good way of checking if your ODE is analytically differentiable.

SOLUTION OF ODES
ANALYTICAL SOLUTIONS

Take the ordinary differential equation for the spring-mass system

$$\ddot{x} + \omega_n^2 . x = 0$$

```
>eq:=diff(x(t),t,t)+omega)^2*x(2)=0;
>sol:=dsolve({eq,x(0)=x0, D(x)(0)=V0},x(t));
```

Yields

$$x = x0.\cos(\omega_o . t) + \frac{v0.\sin(\omega_o . t)}{\omega_o}$$

which is equivalent to the earlier equation

$$x = \frac{\dot{x}(0)}{\omega_n} \sin \omega_n . t + x(0) \cos \omega_n . t$$

SOLUTION OF ODES
NUMERICAL SOLUTIONS

Numerical solutions consider small time steps in the function and approximate to the function using a Taylor series approximation (or something similar). After each small step the new predicted values become the initial values for the next step.

Eulers method – first crude approximation, not recommend.

Runge-Kutta – not computationally very efficient but robust. Varying orders of Runge-Kutta, fourth is the most popular. Fourth order requires four function evaluations per time step. Generally higher order gives better accuracy.

Step-Size Control – the step size is adjusted to allow for estimated local error to achieve a required accuracy. (Smaller step sizes require more time).

SOLUTION OF ODES
STIFF EQUATIONS

Stiffness – this occurs when different processes within the system behave under different time scales. Therefore the solution changes on a different time scale which is short compared to the size of the integration step.

Example: imagine the heel during running, during the flight phase its path is easy to predict and the time scale is relatively long. When impact occurs there are a number of very rapid changes under a different time scale. The equations describing the motion of the heel are stiff.

Solution

- Ignore stiff problems.
- Separate function up if know when time scale changes.
- Use a stiff ODE solver, normally based around a good step-size controller.

REVIEW QUESTIONS

1) What is the difference between inverse and direct dynamics?

2) What is an ordinary differential equation (ODE)?

3) What is a stiff ODE?

4) How can you solve an ODE?

5) Provide a critique of a published paper which uses a spring-mass system to describe an aspect of human movement.