Application 10.5
Engineering Functions

Periodic piecewise linear functions occur so frequently as input functions in engineering applications that they are sometimes called **engineering functions.** Computations with such functions are readily handled by computer algebra systems. In the paragraphs that follow we first show how to define typical engineering functions — such as sawtooth, triangular-wave, and square-wave functions — using *Maple*, *Mathematica*, and MATLAB, and then illustrate the solution of a mass-spring-dashpot problem

$$mx'' + cx' + kx = F(t), \qquad x(0) = x'(0) = 0 \tag{1}$$

when the external force function $F(t)$ is an engineering function. Our main object is to illustrate steady periodic responses to periodic inputs, so it is convenient to use numerical rather than Laplace transform methods. For your own investigation you can solve similarly the initial value problem in (1) with various mass-spring-dashpot parameters — for instance, selected digits of you student ID number — and with input engineering functions having various amplitudes and periods.

Using *Maple*

Typical engineering functions are defined by

```
SawTooth := t -> 2*t - 2*floor(t) - 1:

TriangularWave :=
     t -> abs(2*SawTooth((2*t-1)/4)) - 1:

SquareWave := t -> signum(TriangularWave(t)):
```

A plot of each of these functions verifies that it has period 2 and that its name is aptly chosen. For instance, the result of the command

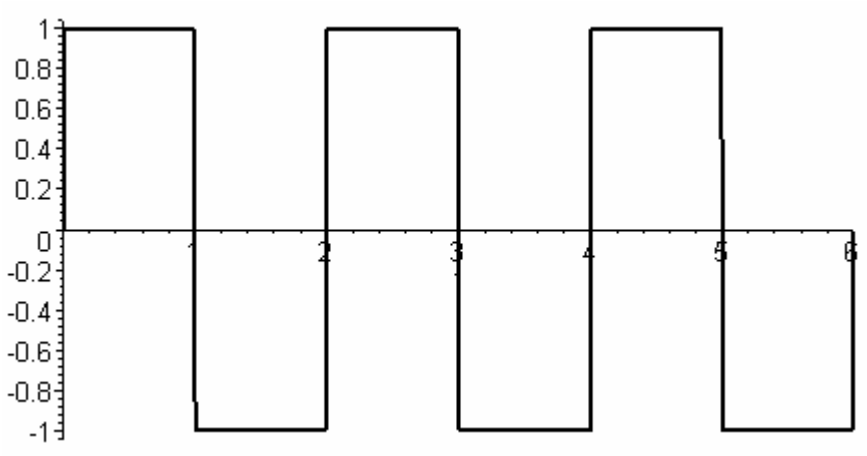
```
plot(SquareWave(t), t = 0..6);
```

shows (at the top of the next page) three square-wave cycles of length 2.

If $f(t)$ is one of the three period 2 engineering functions defined above, then the function $f\left(2t/p\right)$ will have period p. To illustrate this, try

```
plot(TriangularWave(2*t/p), t = 0..3*p);
```

with various values of p. The resulting graph (on the top of the next page) should show three triangular-wave cycles of length p.



Now let's consider the mass-spring-dashpot equation in (1) with selected parameter values and an input forcing function having period p and amplitude F_0, for instance,

```
m := 4:     c := 8:     k := 5:
p := 1:
F0 := 4:
input := t -> F0*SquareWave(2*t/p);
```

You can plot this **input** function to verify that it has period 1:

```
plot(input(t), t = 0..5);
```

Then our differential equation is

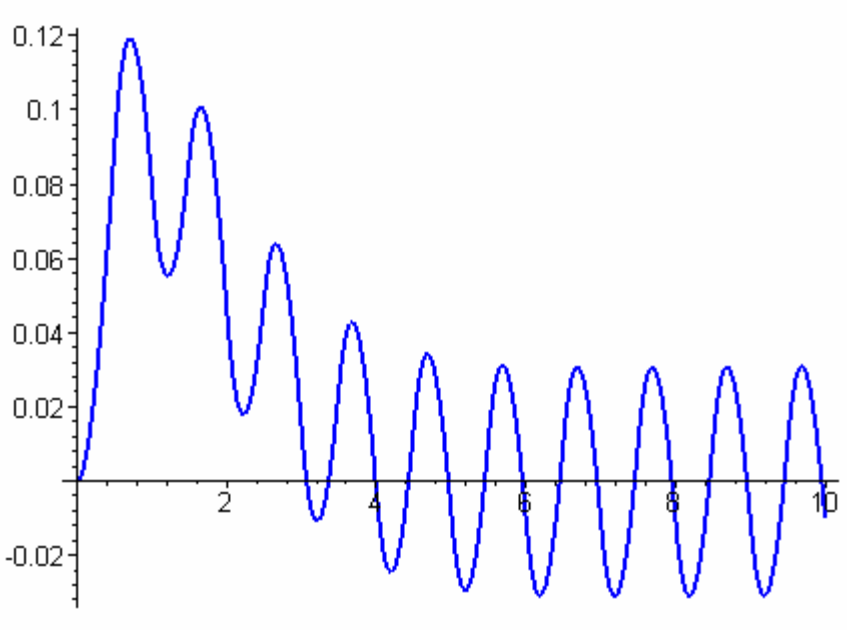
```
de :=
m*diff(x(t),t$2) + c*diff(x(t),t) + k*x(t) = input(t):
```

Next, let's suppose that the mass is initially at rest in its equilibrium position and solve numerically the resulting initial value problem.

```
response :=
dsolve( {diffEq, x(0)=0, D(x)(0)=0}, x(t),
         type=numeric);
```

When we plot this solution, we see that after an initial transient dies out, the response function $x(t)$ settles down (as expected?) to a periodic oscillation with the same period as the input.

```
with(plots):
odeplot(response, [t,x(t)], 0..10, numpoints=300);
```



Using *Mathematica*

Typical engineering functions are defined by

```
SawTooth[t_] := 2t - 2 Floor[t] - 1

TriangularWave[t_] := Abs[2 SawTooth[(2t-1)/4]] - 1

SquareWave[t_] := Sign[TriangularWave[t]]
```

A plot of each of these functions verifies that it has period 2 and that its name is aptly chosen. For instance, the result of the command

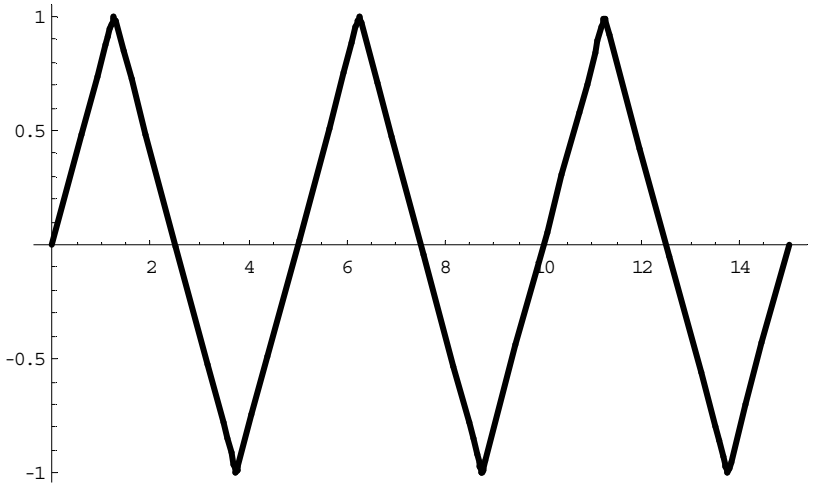
```
Plot[SquareWave[t], {t,0,6}];
```

shows three square-wave cycles of length 2. (Try it.)

If $f(t)$ is one of the three period 2 engineering functions defined above, then the function $f(2t/p)$ will have period p. To illustrate this, try

```
p = 5;
Plot[TriangularWave[2 t/p], {t, 0, 3p}];
```

with various values of p. The resulting graph should show three triangular-wave cycles of length p. For instance, with $p = 5$ we get the period 5 triangular-wave graph shown at the top of the next page.



Now let's consider the mass-spring-dashpot equation in (1) with selected parameter values and an input forcing function having period p and amplitude F_0, for instance,

```
m = 3;    c = 35;     k = 10;
p = 2;
F0 = 15;
input = F0 TriangularWave[2t/p];
```

You can plot this **input** function to verify that it has period 2:

```
Plot[ input, {t, 0, 15} ];
```

Then our differential equation is

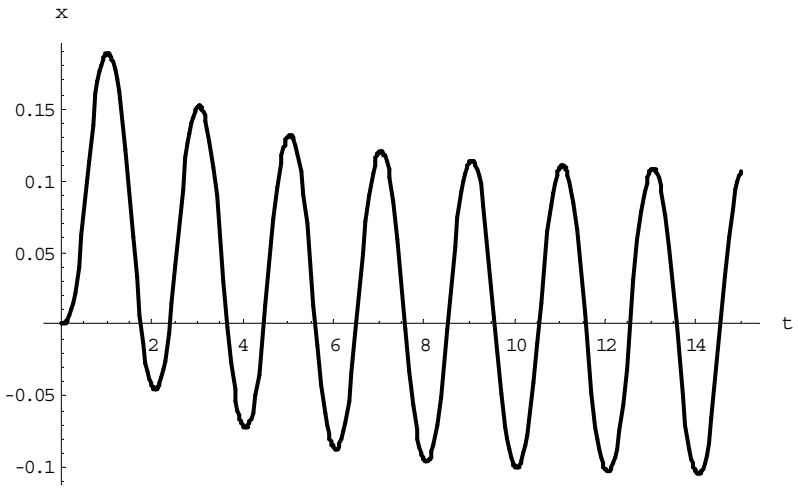
```
de =   m x''[t] + c x'[t] + k x[t] == input
```

Next, let's suppose that the mass is initially at rest in its equilibrium position and solve numerically the resulting initial value problem.

```
response =
NDSolve[{de, x[0]==0, x'[0]==0}, x, {t,0,15}]
```

When we plot this solution, we see that after an initial transient dies out, the response function $x(t)$ settles down (as expected?) to a periodic oscillation with the same period $p = 2$ as the input.

```
Plot[x[t] /. response, {t,0,10}];
```



Using MATLAB

Typical engineering functions are defined by

```
function   x = sawtooth(t)
x = 2*t - 2*floor(t) - 1;

function   x = triangle(t)
x = abs(2*sawtooth((2*t-1)/4)) - 1;

function   x = square(t)
x = sign(triangle(t));
```

A plot of each of these functions verifies that it has period 2 and that its name is aptly chosen. For instance, the result of the commands

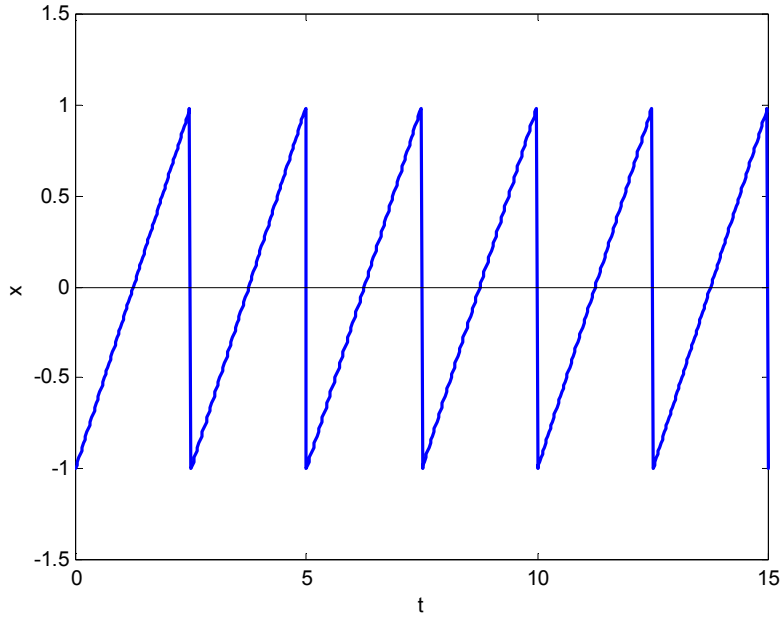
```
t = 0 : 0.01 : 6;
x = square(t);
plot(t,x)
```

shows three square-wave cycles of length 2. (Try it.)

If $f(t)$ is one of the three period 2 engineering functions defined above, then the function $f\left(2t/p\right)$ will have period p. To illustrate this, try

```
p = 5;
t = 0 : p/200 : 3*p;
plot(t, sawtooth(2*t/p))
```

with various values of p. The resulting graph should show three sawtooth-wave cycles of length p. For instance, with $p = 5$ we get the period 5 sawtooth-wave graph shown at the top of the next page.



Now let's consider the mass-spring-dashpot equation in (1) with selected parameter values and an input forcing function having period p and amplitude F_0, for instance,

```
function  F = force(t)
p = 5;    % period
F0 = 400;   % amplitude
F = F0*sawtooth(2*t/p);
```

You can plot this input force function to verify that it has period 5:

```
t = 0 : 0.03 : 15;
plot(t, force(t))
```

Then our differential equation is defined (as a system of two first-order equations) by the function

```
function   xp = de(t,x)
m = 3;   c = 100;   k = 20;
xp = x;
y = x(2);
x = x(1);
xp(1) = y;
xp(2) = (-k*x - c*y + force(t))/m;
```

Next, let's suppose that the mass is initially at rest in its equilibrium position and solve numerically the resulting initial value problem.

```
[t,x] = ode45('de', 0:0.05:30, [0;0]);
response = x(:,1);
```

When we plot this solution, we see that after an initial transient dies out, the response function $x(t)$ settles down (as expected?) to a periodic oscillation with the same period as the input.

```
plot(t,response);
axis([0 30 -2.5 2.5])
hold on
plot([0 30],[0 0],'k')
```