Building Functions from Sine Waves

Approximation of functions by polynomials is the basis of Taylor series, Newton-Raphson iteration, Simpson's Rule, etc.

It can also be useful to approximate functions by sums of sine waves, of varying frequency and amplitude. These *Fourier series* or *frequency domain* representations are used to:

- Analyze speech signals — eg, when trying to recognize spoken language by computer.
- Process audio and video signals to try to remove noise.
- Detect subtle periodicities in data, such as records of world climate.
- Create randomly-generated visual scenes — eg, realistic-looking mountains for the background in a computer animation.

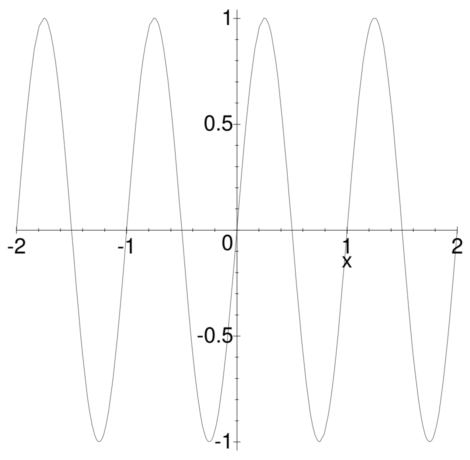
Frequency, Amplitude, and Phase

A sine wave can be written as follows:

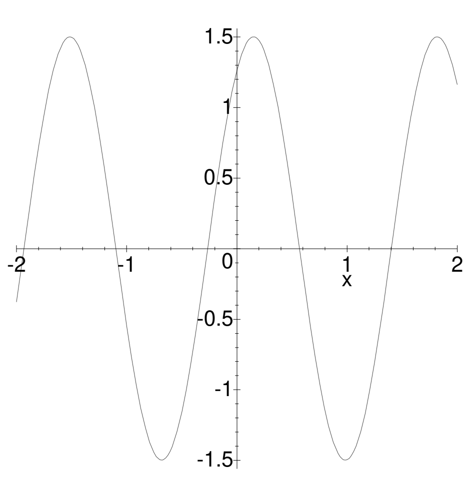
$$y(x) \;=\; A\,\mathsf{sin}(2\pi f x + \phi)$$

Here, A is the *amplitude* of the sine wave, f is its *frequency* (in Hertz (cycles/second) if x is time), and ϕ is the *phase* of the sine wave.

```
> plot(sin(2*Pi*x),x=-2..2);
```




```
> plot(1.5*sin(2*Pi*0.6*x+1),x=-2..2);
```



Functions from Random Sine Waves

Here is a Maple program that creates a 1D function by adding together many sine waves with random phases:

```
waves1 := proc (x, ampl, maxf, n)
  local i, f, result, phase, scale;
  scale := 1/sqrt(n);
  result := 0;
  for i from 1 to n do
    f := maxf*i/n;
    phase := stats[random,uniform[0,2*Pi]]();
    result := result
            + evalf(scale*ampl(f)*sin(2*Pi*f*x+phase));
  od;
  result;
end:
```

The `n` sine waves have frequencies equally spaced from just above zero up to `maxf`.

The amplitudes of the sine waves can vary with frequency, according to the `ampl` function.

The phases of the sine waves are randomly selected, uniformly, from all possible phases.

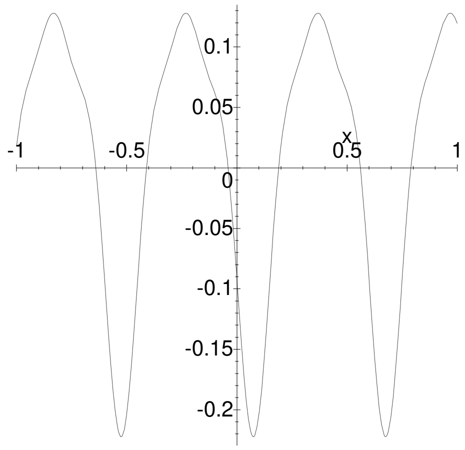
Sums of Many Sine Waves

When we add together a large number of sine waves, it is no longer obvious that the final function was built from sine waves:

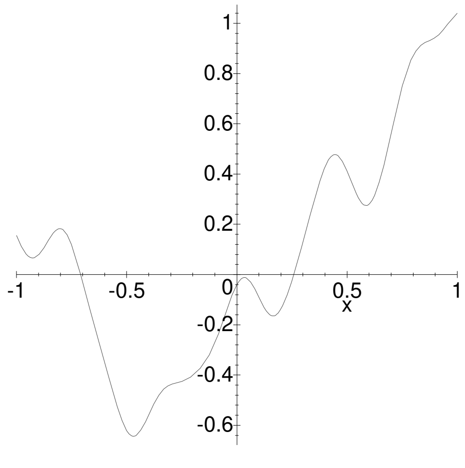
```
> w := waves1(x,f->1/(1+f^2),5,3);
```

$$w := .1528280125\sin(10.47197551\,x + 3.943294813) + .04767112316\sin(20.94395103\,x + 3.084630808) + .02220577959\sin(31.41592654\,x + 2.403573914)$$

```
> plot(w,x=-1..1);
```



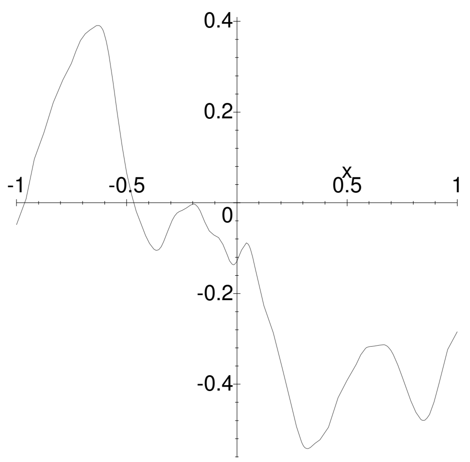

```
> w := waves1(x,f->1/(1+f^2),5,300):
> plot(w,x=-1..1);
```



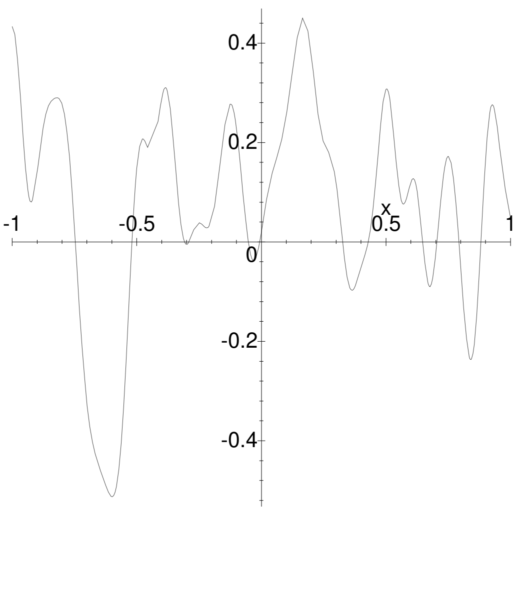
Functions with Different Spectra

The way a random function looks depends on the way the amplitude varies with the frequency — what is called the *spectrum*.

```
> plot(waves1(x,f->1/(1+f^2),10,1000), x=-1..1);
```



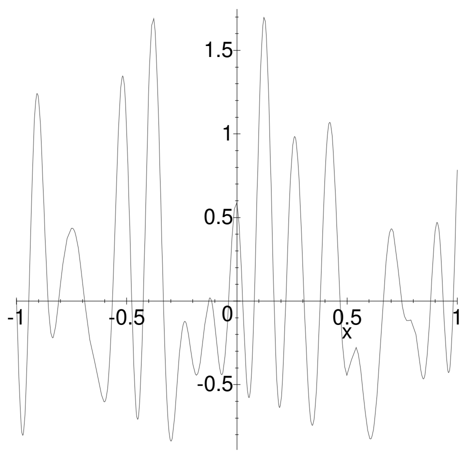

```
> plot(waves1(x,f->1/(1+f^2)^0.5,10,1000), x=-1..1);
```



White Noise

When the amplitude is constant over frequency (a flat spectrum), we get what is called *white noise*:

```
> plot(waves1(x,f->1,10,1000), x=-1..1);
```



For true white noise, there is no upper limit on the frequency, but then the function becomes impossible to plot.

White noise is "white" because it contains equal amounts of every frequency, just as white light contains equal amounts of all colours.

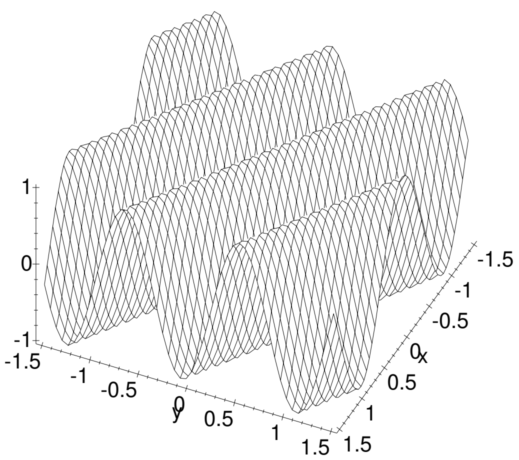
Sine Waves in Two Dimensions

A sine wave can also be defined in two dimensions, with an equation of the form:

$$z(x, y) = A \sin(2\pi f[\cos(\theta)x + \sin(\theta)y] + \phi)$$

Here, θ gives the direction in which the sine wave waves. The function is constant in the perpendicular direction.

```
> plot3d(sin(2*Pi*(cos(1)*x+sin(1)*y)),
        x=-1.5..1.5,y=-1.5..1.5,grid=[50,50]);
```



Building Random 2D Functions

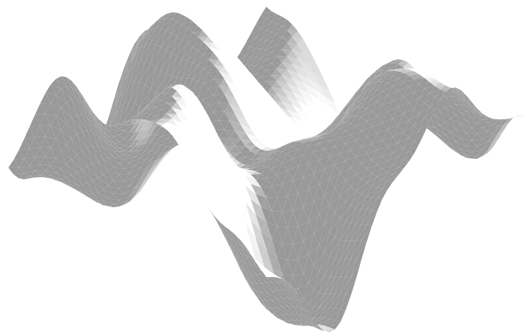
Here is a Maple program that creates a 2D function by adding together many sine waves with random phases, and random wave directions:

```
waves2 := proc (x, y, ampl, maxf, n)
  local i, f, k, result, phase, angle, scale;
  scale := 1/sqrt(n);
  result := 0;
  for i from 1 to n do
    f := maxf*i/n;
    phase := stats[random,uniform[0,2*Pi]]();
    angle := stats[random,uniform[0,2*Pi]]();
    k := cos(angle)*x + sin(angle)*y;
    result := result
            + evalf(scale*ampl(f)*sin(2*Pi*f*k+phase));
  od;
  result;
end:
```

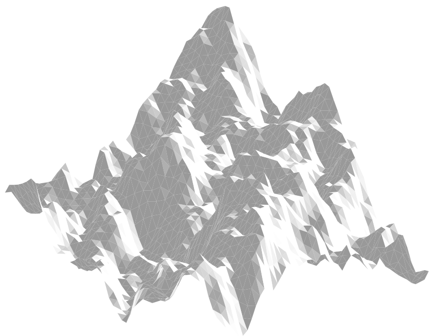
One use of this function is to build artificial mountains, for use in computer graphics.

Armchair Mountain Building

```
> plot3d(waves2(x,y,f->exp(-f^2),10,500),
         x=-1.5..1.5,y=-1.5..1.5,grid=[50,50]);
```



```
> plot3d(waves2(x,y,f->1/(1+f^2),10,500),
         x=-1.5..1.5,y=-1.5..1.5,grid=[50,50]);
```



Where Do We Go From Here?

A variety of interesting functions can be obtained by adding together sine waves. But can *all* functions be obtained this way?

Not quite, but a wide class can be represented in this way. This has important consequences:

- We can use sums of sine waves as approximations, in the same way that we have used polynomials.
- We can gain insight into a function by looking at its representation in terms of sine waves.
- We can determine the result of applying a *linear* operation to a function by seeing how that operation affects each of the component sine waves.

The last point above is crucial when analysing the effect of *filters* on functions — such as the audio filters that are used in stereo systems.