Point

The Point structure is used as a substructure usually to store the location of the mouse on the screen.

```
Structure Point
   Int16 v
   Int16 h
EndStructure
```

PointF

The PointF structure is the same as Point but with floating point fields.

```
Structure Point
   float v
   float h
EndStructure
```

poissonNoise

poissonNoise(*num*)

The poissonNoise function returns a pseudo-random value from the Poisson distribution whose probability distribution function is

$$f(x;\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!}, \qquad \begin{array}{c} \lambda > 0 \\ x = 0,1,2... \end{array}$$

with mean and variance equal to *num* (= λ).

The random number generator initializes using the system clock when Igor Pro starts. This almost guarantees that you will never repeat a sequence. For repeatable "random" numbers, use **SetRandomSeed**. The algorithm uses the Mersenne Twister random number generator.

See Also

The **SetRandomSeed** operation.

Noise Functions on page III-390.

Chapter III-12, **Statistics** for a function and operation overview.

poly

poly(*coefsWaveName, x1*)

The poly function returns the value of a polynomial function at x = *x1*.

coefsWaveName is a wave that contains the polynomial coefficients. The number of points in the wave determines the number of terms in the polynomial.

If you use poly in a real expression, x1 must be real and poly returns a real value. The wave containing the coefficients can be real or complex. Complex coefficients are interpreted as real(coef).

If you use poly in a complex expression, x1 must complex and poly returns a complex value. The wave containing the coefficients can be real or complex. Real coefficients are interpreted as cmplx(coef,0). Support for complex expressions was added in Igor Pro 9.00.

Examples

```
// Fill wave0 with 100 points containing the polynomial 1 + 2*x + 3*x^2 + 4*x^3
// evaluated over the range from x = -1 to x= 1
Make coefs = {1, 2, 3, 4}        // f(x) = 1 + 2*x + 3*x^2 + 4*x^3
Make/N=100/O wave0; SetScale/I x, -1, 1, wave0
wave0 = poly(coefs, x)
Display wave0
```

poly2D

poly2D(*coefsWaveName, x1, y1*)

The poly2D function returns the value of a 2D polynomial function at x = *x1*, y = *y1*.