Application Demonstration

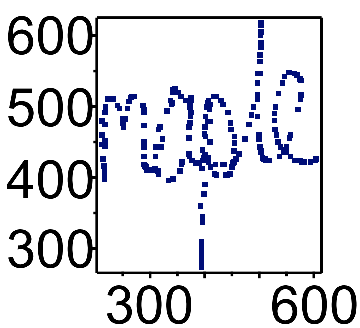
www.maplesoft.com

Generating Parametric Curves from 2-D Data using Discrete Fourier Transforms

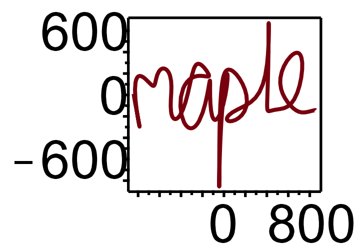
Introduction

This application will generate parametric equations for a set of 2-D points. When plotted, the parametric equations resemble the shape of the points.

Turn X-Y points like this...	...into two parametric equations...	...which look like this when plotted



$$
\begin{aligned}
X(t) ={}& -620.\ \sin(-1.6+0.084\,t)\\
&- 4.1\ \sin(-4.3+0.17\,t)\\
&- 130.\ \sin(-1.6+0.25\,t)\\
&- 39.\ \sin(-4.7+0.34\,t)\\
&- 18.\ \sin(-1.6+0.43\,t)\\
&- 33.\ \sin(1.5+0.50\,t)\\
&- 15.\ \sin(-4.7+0.78\,t)\\
&- 14.\ \sin(-1.5+0.59\,t)\\
&- 9.6\ \sin(-1.9+0.68\,t)\\
&- 27.\ \sin(-1.7+0.84\,t)\ \ldots\\
Y(t) ={}& -89.\ \sin(1.5+1.7\,t)\\
&- 59.\ \sin(-1.7+1.4\,t)\\
&- 49.\ \sin(-1.8+1.3\,t)\\
&- 17.\ \sin(1.3+0.84\,t)\\
&- 28.\ \sin(-1.5+0.78\,t)\\
&- 1.2\ \sin(-1.4+0.59\,t)\\
&- 3.7\ \sin(-4.0+0.50\,t)\\
&- 15.\ \sin(1.4+0.43\,t)\\
&- 21.\ \sin(-4.4+0.25\,t)\\
&- 84.\ \sin(-1.6+0.17\,t)\\
&- 26.\ \sin(-1.4+0.084\,t)\\
&- 16.\ \sin(1.5+3.0\,t)\\
&- 8.5\ \sin(-4.7+2.8\,t)\\
&- 15.\ \sin(-1.2+2.7\,t)\ \ldots
\end{aligned}
$$



This application has an interactive plot that lets you draw a curve. Maple will generate discrete points on this curve as it is drawn. Maple will then

- compute the discrete Fourier transforms (DFT) of the X and Y coordinates.
- generate two parametric equations that consist of a sum of sines. The frequency and amplitude of each sine term are extracted from the DFT.
- assign the equations to two variables.

The equations can be now be plotted or manipulated.

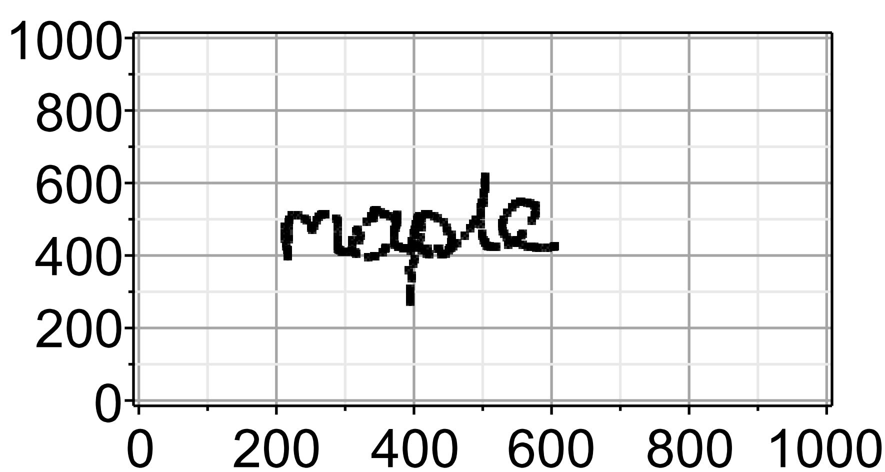
The core calculations are exposed at the bottom of the worksheet.

INSTRUCTIONS
1. Click the reset button (this removes the old points from the plot)
2. Click and drag a curve on the large plot (you could draw any curve, including "handwritten" words, as long as it is one continuous curve). Maple will generate points on this curve as it is drawn.
3. Release the left-mouse button when you're finished with the curve.. Maple will now generate the parametric equations, plot them in a preview, and assign the equations to two variables.

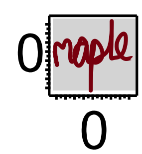
You can remove every nth point from the input data using the data filter slider (a value of 2 seems to generate a good result).

Application

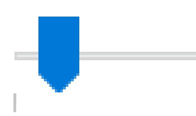
Click and drag a curve on the plot below



The parametric equations are plotted below



Data filter



3

Reset

218 points

Equations

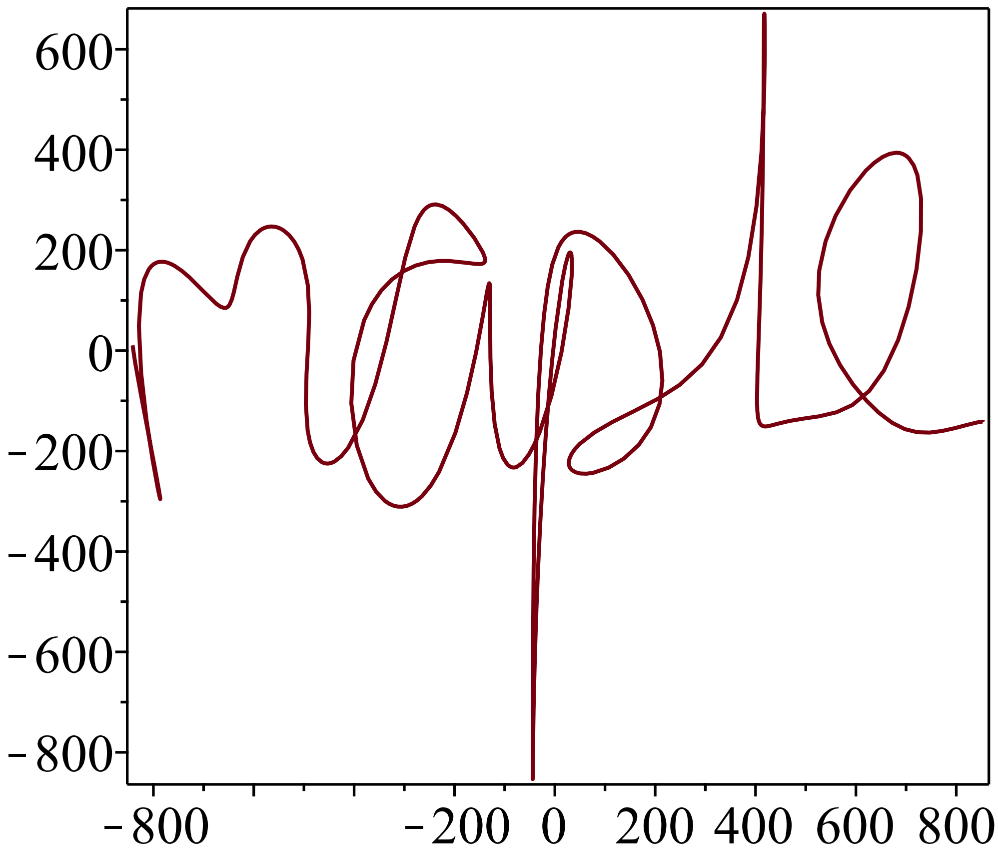
The parametric equations are assigned to Actions:-x and Actions:-y

```
> evalf(Actions:-x, 2);
  evalf(Actions:-y, 2)
```

$$-33.\sin(1.5+0.50t)-39.\sin(-4.7+0.34t)-18.\sin(-1.6+0.43t)-130.\sin(-1.6+0.25t)-4.1\sin(-4.3+0.17t)-620.\sin(-1.6+0.084t)-6.2\sin(-1.1+3.0t)-4.2\sin(1.2+2.9t)-11.\sin(-1.8+2.8t)-3.2\sin(1.5+2.7t)-3.6\sin(1.4+2.5t)-3.9\sin(-2.2+2.6t)-5.0\sin(0.74+2.4t)-4.3\sin(-1.6+2.4t)-4.6\sin(-4.4+2.2t)-7.0\sin(-1.8+2.3t)-3.7\sin(-0.90+2.1t)-8.8\sin(0.96+2.0t)-13.\sin(-1.5+1.9t)-26.\sin(1.4+2.0t)-16.\sin(-1.9+1.8t)-14.\sin(1.5+1.7t)-5.8\sin(-1.5+1.6t)-22.\sin(1.4+1.5t)-57.\sin(-1.7+1.4t)-37.\sin(-1.8+1.3t)-19.\sin(-1.7+1.1t)-41.\sin(-1.7+1.2t)-24.\sin(1.4+1.0t)-20.\sin(-1.7+0.93t)-27.\sin(-1.7+0.84t)-9.6\sin(-1.9+0.68t)-15.\sin(-4.7+0.78t)-14.\sin(-1.5+0.59t)$$

$$-89.\sin(1.5+1.7t)-59.\sin(-1.7+1.4t)-49.\sin(-1.8+1.3t)-15.\sin(1.4+0.43t)-130.\sin(1.5+0.34t)-21.\sin(-4.4+0.25t)-26.\sin(-1.4+0.084t)-84.\sin(-1.6+0.17t)-16.\sin(1.5+3.0t)-22.\sin(-1.9+2.9t)-15.\sin(-1.2+2.7t)-8.5\sin(-4.7+2.8t)-9.6\sin(-1.7+2.6t)-12.\sin(-2.0+2.5t)-7.6\sin(0.59+2.4t)-81.\sin(1.4+2.3t)-5.1\sin(-3.6+2.4t)-48.\sin(-1.7+2.2t)-8.5\sin(-1.2+2.1t)-23.\sin(-1.7+2.0t)-26.\sin(-4.7+2.0t)-61.\sin(-1.7+1.9t)-29.\sin(1.5+1.8t)-49.\sin(-1.6+1.6t)-64.\sin(-1.7+1.5t)-24.\sin(1.3+1.5t)-110.\sin(-1.6+1.2t)-23.\sin(1.3+1.1t)-230.\sin(1.5+1.0t)-17.\sin(1.3+0.84t)-50.\sin(1.5+0.93t)-28.\sin(-1.5+0.78t)-40.\sin(-4.6+0.68t)-3.7\sin(-4.0+0.50t)-1.2\sin(-1.4+0.59t) \quad \textbf{(3.1)}$$

```
> plot([Actions:-x, Actions:-y, t = 0..Actions:-t_max], scaling =
  constrained, axes = boxed)
```



Fun stuff you can do with the equations

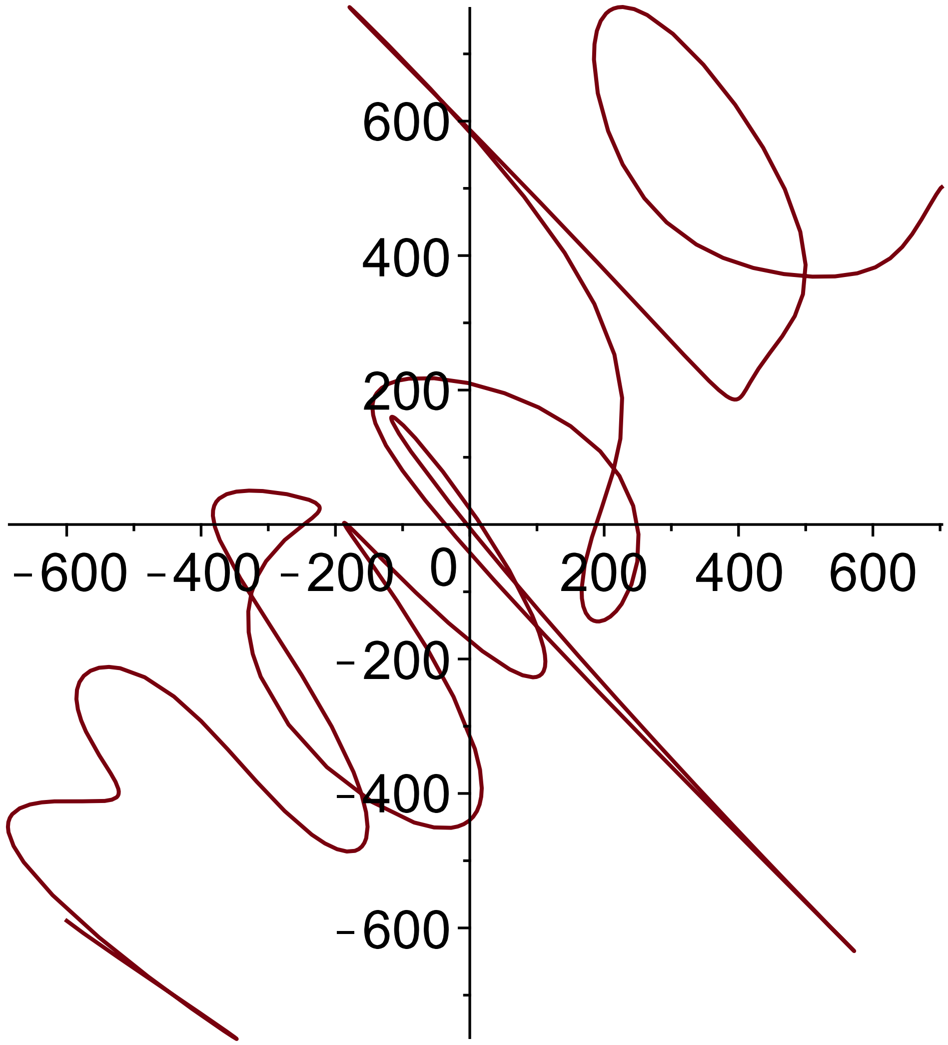
Rotation

```
> rot := Matrix([[cos(theta), -sin(theta)], [sin(theta), cos
  (theta)]]) . < x, y>
```

$$rot := \begin{bmatrix} \cos(\theta)\, x - \sin(\theta)\, y \\ \sin(\theta)\, x + \cos(\theta)\, y \end{bmatrix} \quad (4.1.1)$$

```
> rot_ang := Pi / 4:
  eq_vec := eval(subs({x = Actions:-x, y = Actions:-y}, rot),
  theta = rot_ang):

  plot([eq_vec[1], eq_vec[2], t = 0..Actions:-t_max], scaling =
  constrained, axesfont = [Arial])
```



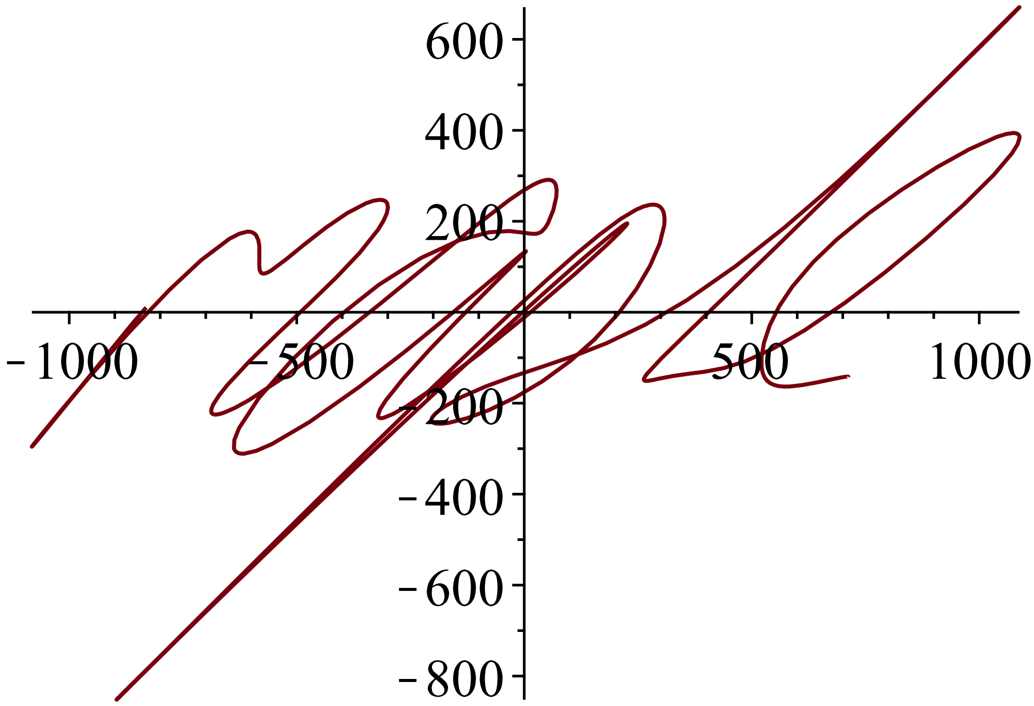
Shear in the x direction

```
> shear_x := Matrix([[1, lambda], [0, 1]]). < x, y>
```

$$shear\_x := \begin{bmatrix} \lambda y + x \\ y \end{bmatrix} \quad (4.2.1)$$

```
> shear_value := 1:
  eq_vec := eval(subs({x = Actions:-x, y = Actions:-y}, 
  shear_x), lambda = shear_value):

  plot([eq_vec[1], eq_vec[2], t = 0..Actions:-t_max], scaling = 
  constrained)
```



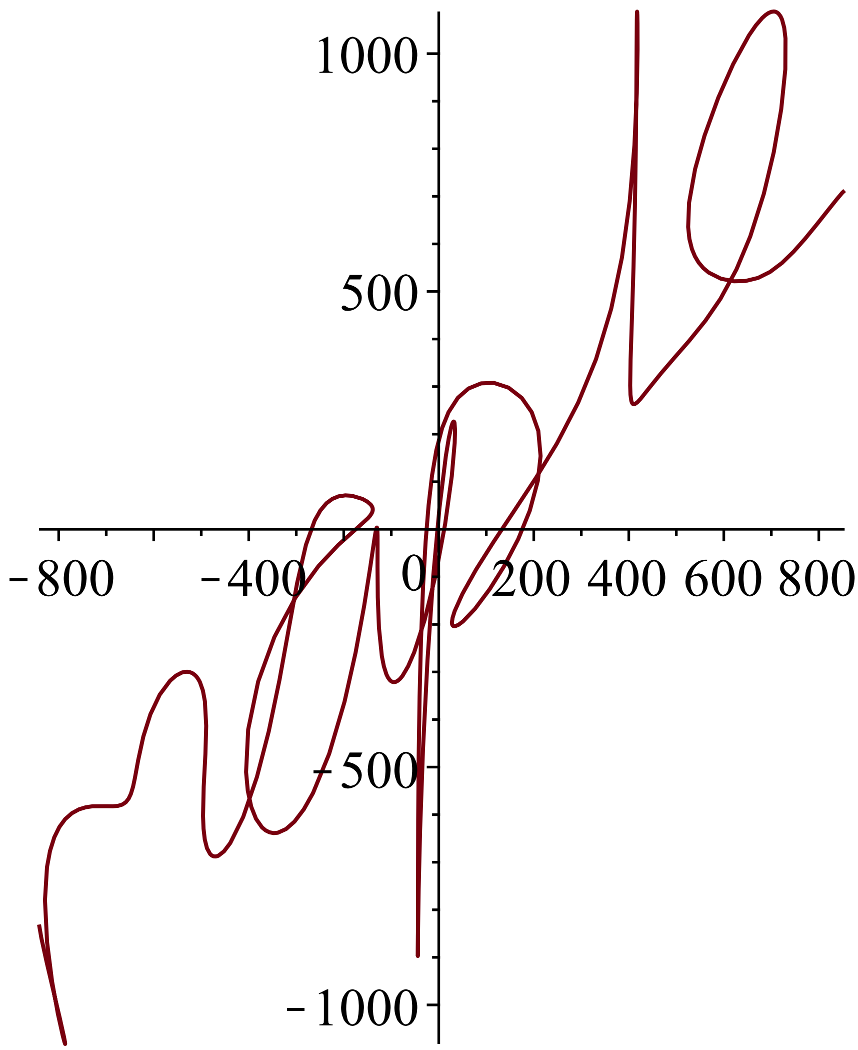
Shear in the y direction

```
> shear_y := Matrix([[1, 0], [lambda, 1]]). <x, y>
```

$$shear\_y := \begin{bmatrix} x \\ \lambda x + y \end{bmatrix} \quad \textbf{(4.3.1)}$$

```
> shear_value := 1:
  eq_vec := eval(subs({x = Actions:-x, y = Actions:-y},
  shear_y), lambda = shear_value):

  plot([eq_vec[1], eq_vec[2], t = 0..Actions:-t_max], scaling =
  constrained)
```



Interactive app to control rotation and shear

```
> rot_shear := Matrix([[cos(theta), -sin(theta)],[sin(theta),
  cos(theta)]]) . Matrix([[1, lambda],[0, 1]]) . < x, y>
```

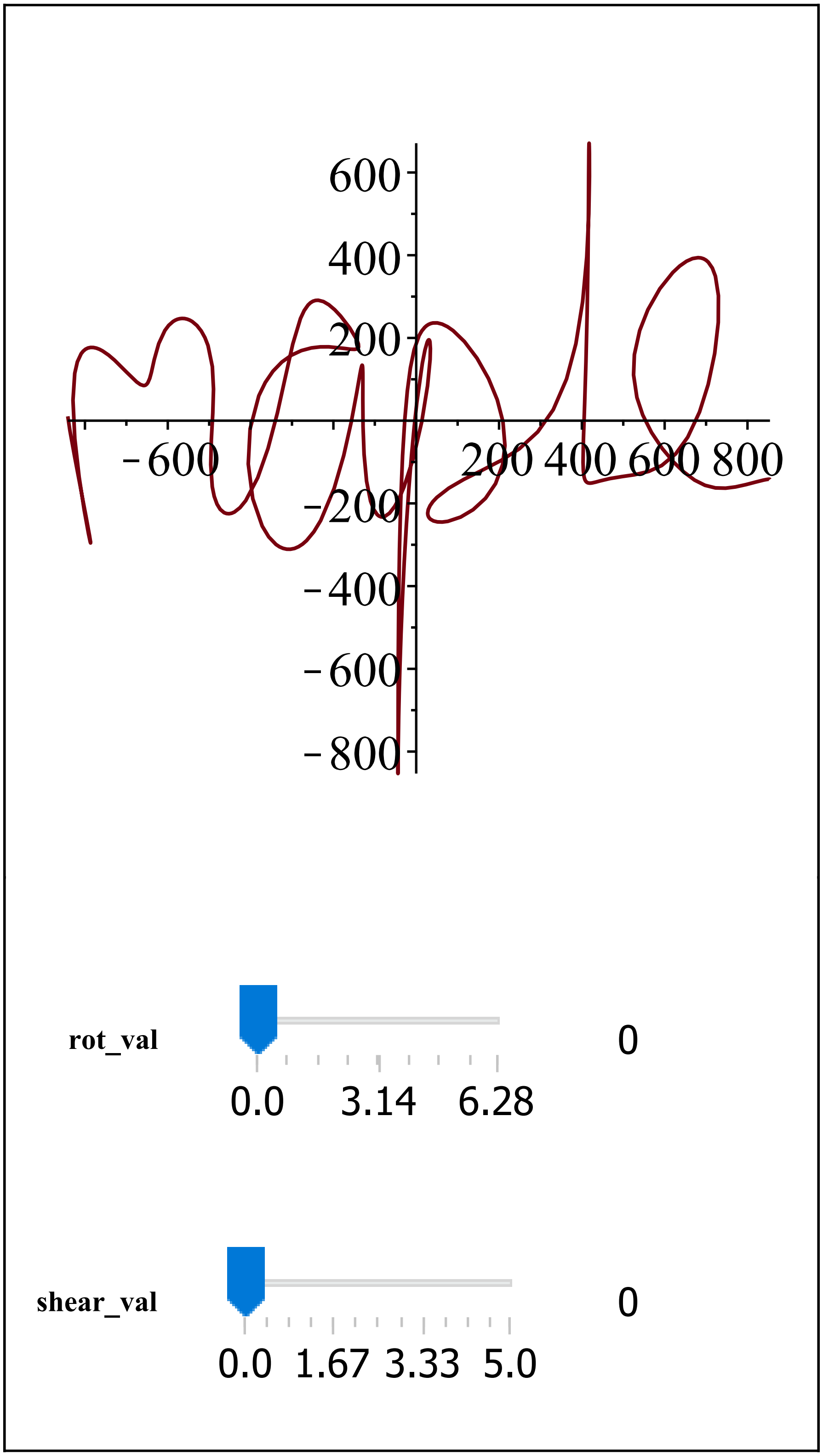
$$rot\_shear := \begin{bmatrix} \cos(\theta)\, x + (\cos(\theta)\, \lambda - \sin(\theta))\, y \\ \sin(\theta)\, x + (\sin(\theta)\, \lambda + \cos(\theta))\, y \end{bmatrix} \quad (4.4.1)$$

```
> f := proc(rot_ang, shear_value)
     local eq_vec:

     eq_vec := eval(subs({x = Actions:-x, y = Actions:-y},
  rot_shear), [lambda =shear_value, theta = rot_ang]):

     return plot([eq_vec[1], eq_vec[2], t = 0..Actions:-t_max],
  scaling = constrained)

  end proc:
> Explore(f(rot_val, shear_val), parameters=[rot_val = 0..2.*
  Pi, shear_val = 0..5.0])
```



Theory

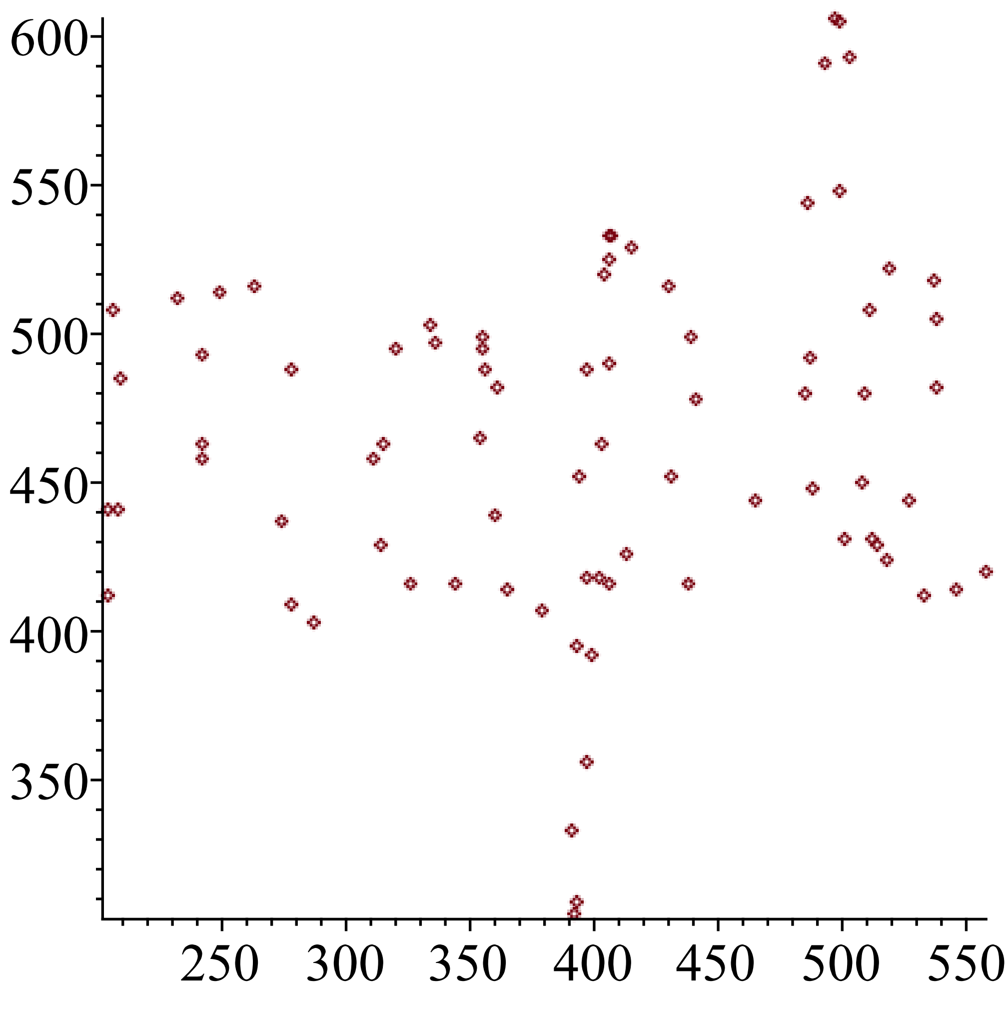
The code for the application is in the startup code region editor (Edit > Startup Code). The core calculations are exposed here.

Consider these points

```
> X := Array([558, 533, 512, 509, 519, 538, 527, 501, 485, 493,
  499, 499, 465, 406, 413, 441, 430, 406, 397, 393, 392, 397, 403,
  406, 397, 365, 356, 354, 326, 311, 336, 355, 315, 278, 278, 249,
  242, 232, 204, 208, 209, 204, 206, 242, 242, 263, 274, 287, 334,
  361, 320, 314, 344, 355, 360, 379, 406, 407, 399, 393, 391, 394,
  404, 415, 439, 431, 402, 438, 487, 503, 497, 486, 488, 514, 538,
  537, 511, 508, 518, 546]):

  Y := Array([420, 412, 431, 480, 522, 505, 444, 431, 480, 591,
  605, 548, 444, 416, 426, 478, 516, 533, 488, 395, 305, 356, 463,
  525, 418, 414, 488, 465, 416, 458, 497, 495, 463, 409, 488, 514,
  458, 512, 441, 441, 485, 412, 508, 493, 463, 516, 437, 403, 503,
  482, 495, 429, 416, 499, 439, 407, 490, 533, 392, 309, 333, 452,
  520, 529, 499, 452, 418, 416, 492, 593, 606, 544, 448, 429, 482,
  518, 508, 450, 424, 414]):

  plot(X, Y, style = point)
```



DFT of X and Y points

```
> dft_X  := Vector(SignalProcessing:-FFT(X)):
  dft_Y  := Vector(SignalProcessing:-FFT(Y)):
```

Phase and amplitude of sines

```
> n   := numelems(X):

  P_X  := Re(argument~(dft_X[2..ceil(n / 2)])):
  A_X  := abs(dft_X[2..ceil(n / 2)]):

  P_Y  := Re(argument~(dft_Y[2..ceil(n / 2)])):
  A_Y  := abs(dft_Y[2..ceil(n / 2)]):
```

Parametric curves

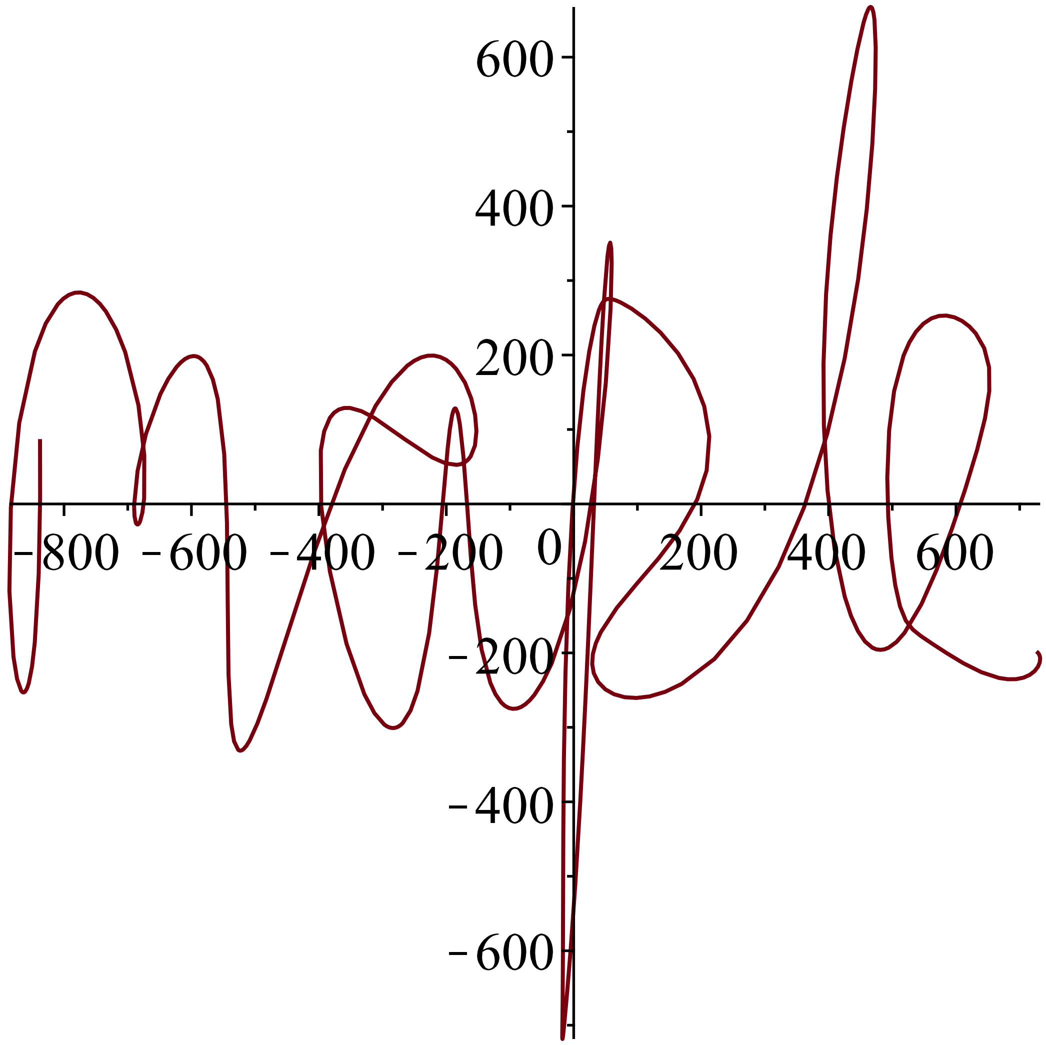
```
> curve_X := evalf[3](add(seq(A_X[i] * sin(Pi / 2 - 2 * Pi / n * i
  * t + P_X[i]), i = 1..ceil(n / 2) - 1)));
  curve_Y := evalf[3](add(seq(A_Y[i] * sin(Pi / 2 - 2 * Pi / n * i
  * t + P_Y[i]), i = 1..ceil(n / 2) - 1)));
```

$$curve\_X := -53.8\sin(-1.83+1.18t) - 27.1\sin(-1.70+1.02t) - 24.8\sin(1.28+0.942t) - 10.1\sin(1.19+0.864t) - 12.8\sin(1.41+0.706t) - 30.5\sin(-1.70+0.785t) - 6.13\sin(-1.81+0.628t) - 33.7\sin(-1.71+0.550t) - 3.65\sin(0.77+0.392t) - 31.5\sin(1.51+0.471t) - 55.9\sin(1.52+0.314t) - 120.\sin(-1.62+0.236t) - 596.\sin(-1.59+0.0785t) - 51.2\sin(1.56+0.157t) - 1.94\sin(-2.63+3.06t) - 4.39\sin(1.35+2.98t) - 3.86\sin(0.19+2.83t) - 3.67\sin(-2.57+2.90t) - 2.18\sin(-2.35+2.75t) - 1.74\sin(0.60+2.67t) - 6.48\sin(-2.09+2.59t) - 4.14\sin(-1.55+2.43t) - 3.92\sin(-3.50+2.51t) - 2.84\sin(-0.583+2.36t) - 6.46\sin(1.20+2.28t) - 2.10\sin(-4.52+2.12t) - 8.09\sin(-1.76+2.20t) - 23.0\sin(-2.02+2.04t) - 11.1\sin(1.20+1.96t) - 0.102\sin(-1.23+1.81t) - 5.37\sin(1.05+1.88t) - 14.1\sin(1.12+1.73t) - 20.3\sin(-2.01+1.65t) - 16.4\sin(-1.94+1.57t) - 4.18\sin(-1.95+1.41t) - 27.8\sin(1.31+1.49t) - 11.6\sin(1.15+1.33t) - 22.2\sin(-1.90+1.26t) - 28.1\sin(-1.92+1.10t)$$

$$curve\_Y := -104.\sin(-1.98+1.41t) - 29.7\sin(-1.95+1.33t) - 13.0\sin(-1.58+1.26t) - 76.6\sin(-1.91+1.10t) - 54.3\sin(-1.91+1.18t) - 14.4\sin(1.15+1.02t) - 229.\sin(1.32+0.942t) - 41.8\sin(-1.67+0.785t) - 90.8\sin(1.47+0.864t) - 63.4\sin(-1.74+0.706t) - 28.8\sin(1.35+0.628t) - 4.99\sin(-0.948+0.471t) - 14.9\sin(-4.62+0.550t) - 34.5\sin(1.48+0.392t) - 90.1\sin(1.50+0.236t) - 55.1\sin(-1.67+0.157t) - 9.29\sin(0.75+3.06t) - 16.7\sin(-1.48+2.98t) - 7.94\sin(-3.43+2.83t) - 9.47\sin(-0.03+2.90t) - 15.4\sin(-1.10+2.75t) - 7.00\sin(-2.07+2.67t) - 16.4\sin(-1.75+2.51t) - 17.5\sin(1.36+2.59t) - 33.7\sin(1.24+2.43t) - 47.7\sin(-2.15+2.36t) - 50.6\sin(0.98+2.28t) - 9.02\sin(0.20+2.12t) - 35.0\sin(-2.21+2.20t) - 14.5\sin(1.02+2.04t) - 5.39\sin(-1.78+1.96t) - 20.0\sin(0.94+1.81t) - 20.8\sin(-2.20+1.88t) - 20.4\sin(1.38+1.73t) - 66.0\sin(-1.96+1.65t) - 53.7\sin(1.16+1.49t) - 50.8\sin(1.14+1.57t) - 127.\sin(1.52+0.314t) - 40.1\sin(-1.59+0.0785t) \tag{5.1}$$

Plot the parametric curves

```
> plot([curve_X, curve_Y, t = 0 .. n / 2])
```


600
400
200
-200
-400
-600
-800
-600
-400
-200
0
200
400
600