

## Least Squares

```
> with(LinearAlgebra):  
with(plots):  
Warning, the name changecoords has been redefined
```

First we generate some data (4 points) by using a polynomial of degree three:

```
> f:=t->2-t^2+1/2 *t^3;
```

$$f := t \rightarrow 2 - t^2 + \frac{1}{2}t^3$$

```
> ff:=t->[t, f(t)];
```

$$ff := t \rightarrow [t, f(t)]$$

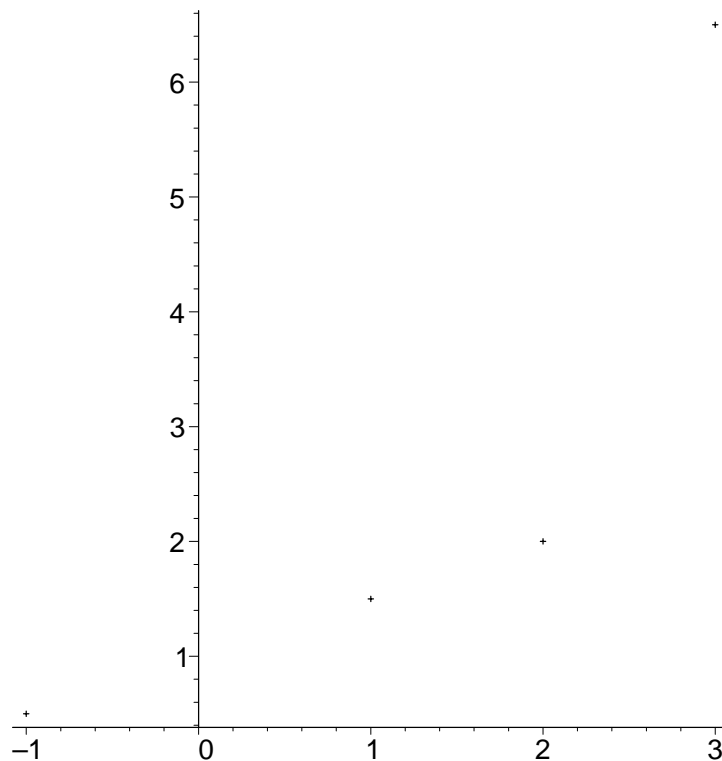
```
> x_pts:=[-1, 1, 2, 3];
```

$$x\_pts := [-1, 1, 2, 3]$$

```
> points:=map(ff, x_pts);
```

$$points := \left[ \left[ -1, \frac{1}{2} \right], \left[ 1, \frac{3}{2} \right], [2, 2], \left[ 3, \frac{13}{2} \right] \right]$$

```
> graph_points:=listplot(points, style=POINT):  
display(graph_points);
```



Next we propose a degree 3 model and try to adjust the data points:

```
> N:=3;
```

$$N := 3$$

This is the model that we propose

```
> f_model := (t, n) -> sum(' a [j] * t ^ j ', ' j ' = 0 .. n );
```

$$f\_model := (t, n) \rightarrow \sum_{j=0}^n 'a_j t^j'$$

And these equations are the condition that the model should pass through the data points

```
> eqs := [ seq ( f_model ( x_pts [ i ], N ) = points [ i ] [ 2 ], i = 1 .. nops ( x_pts ) ) ];
```

```
eqs :=
```

$$\left[ a_0 - a_1 + a_2 - a_3 = \frac{1}{2}, a_0 + a_1 + a_2 + a_3 = \frac{3}{2}, a_0 + 2a_1 + 4a_2 + 8a_3 = 2, a_0 + 3a_1 + 9a_2 + 27a_3 = \frac{13}{2} \right]$$

Rewrite the equations in terms of matrices

```
> (A, b) := GenerateMatrix ( eqs, [ seq ( a [ j ], j = 0 .. N ) ] );
```

$$A, b := \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \\ 2 \\ \frac{13}{2} \end{bmatrix}$$

Find the least squares solution to the problem:

```
> sol := MatrixInverse ( Transpose ( A ) . A ) . Transpose ( A ) . b;
```

$$sol := \begin{bmatrix} 2 \\ 0 \\ -1 \\ \frac{1}{2} \end{bmatrix}$$

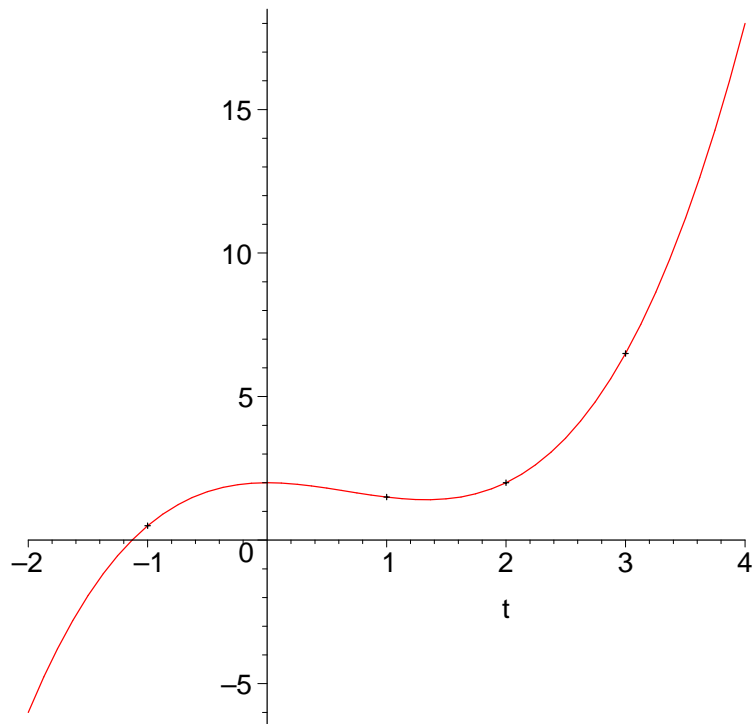
Finally, produce a graph of the least squares solution together with the data points:

```
> f_sol := t -> sum ( ' sol [ j ] * t ^ ( j - 1 ) ', ' j ' = 1 .. Dimension ( sol ) );
```

```
> curve [ N ] := plot ( f_sol ( t ), t = -2 .. 4 );
```

```
display ( curve [ N ], graph_points );
```

$$f\_sol := t \rightarrow \sum_{j=1}^{\text{LinearAlgebra:Dimension(sol)}} 'sol_j t^{(j-1)},$$



```
> Err:=VectorNorm(b-A.sol,2);
```

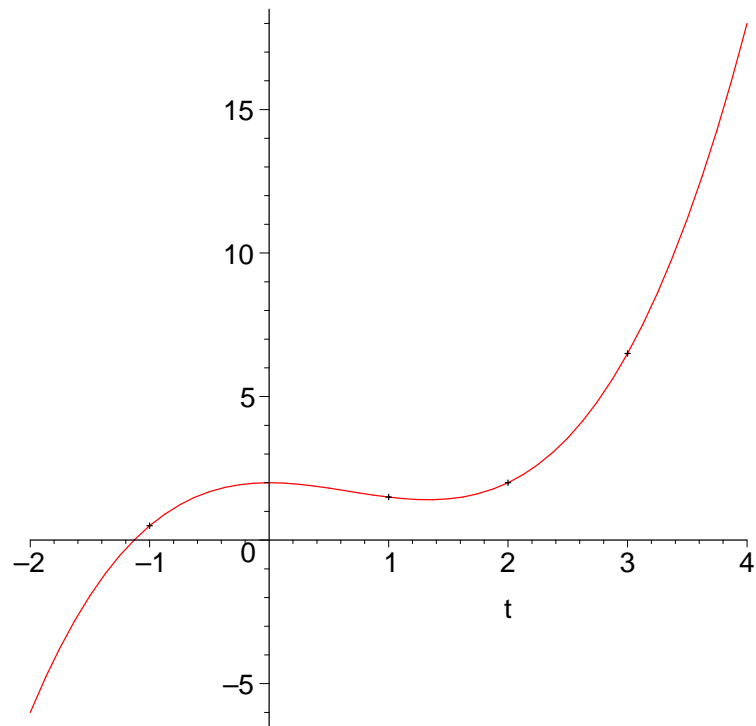
```
Err:=0
```

Next, we put all of the above together in a function:

```
> do_approx:= proc(N,curve)
  local f_model, eqs, A, b, sol, f_sol, err;
  description "approximate some data with a polynomial of degree
  N";
  f_model:=(t,n)->sum('a[j]*t^j','j'=0..n);
  eqs:=[seq(f_model(x_pts[i],N)=points[i][2],i=1..nops(x_pts))];
  (A,b):=GenerateMatrix(eqs,[seq(a[j],j=0..N)]);
  sol:=MatrixInverse(Transpose(A).A).Transpose(A).b;
  err:=VectorNorm(b-A.sol,2);
  printf("Approximation Error: %g\n",err);
  f_sol:=t->sum('sol[j]*t^(j-1)','j'=1..Dimension(sol));
  curve[N]:=plot(f_sol(t),t=-2..4);
  display(curve[N],graph_points);
end proc;
```

First we apply the procedure to the same situation as before, a polynomial of degree 3

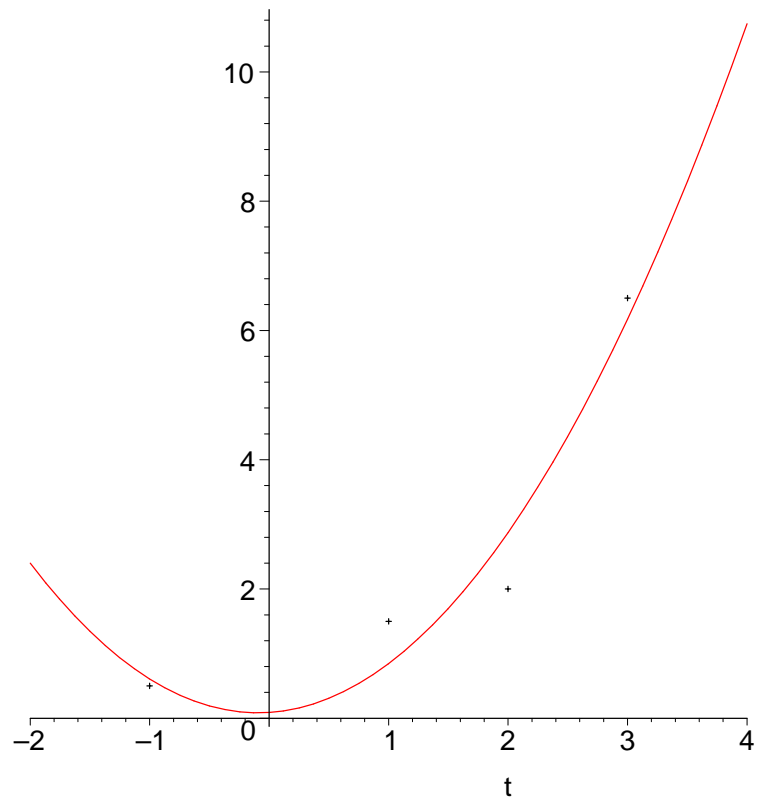
```
> do_approx(3,curve);
Approximation Error: 0
```



Then we model with a degree 2 polynomial

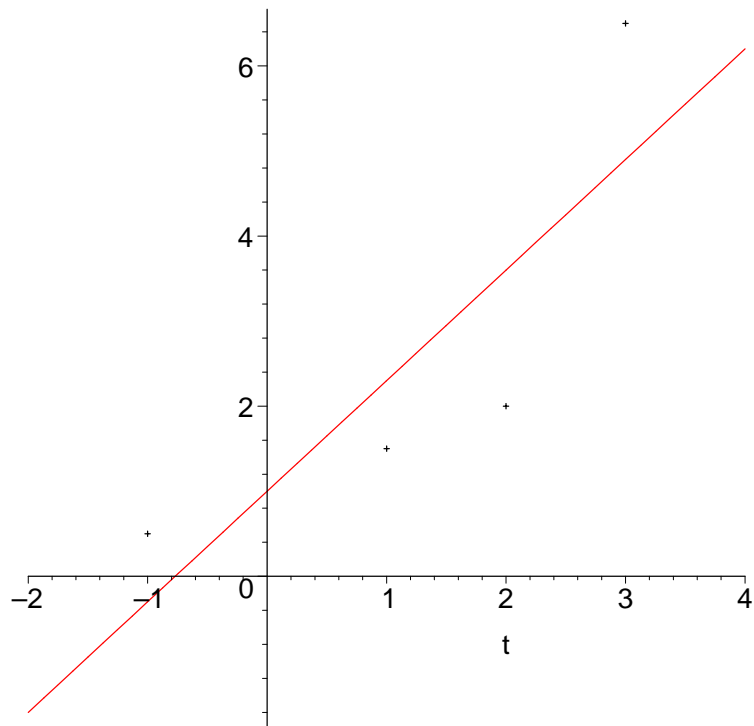
```
> do_approx(2, curve);
```

```
Approximation Error: 1.144155
```



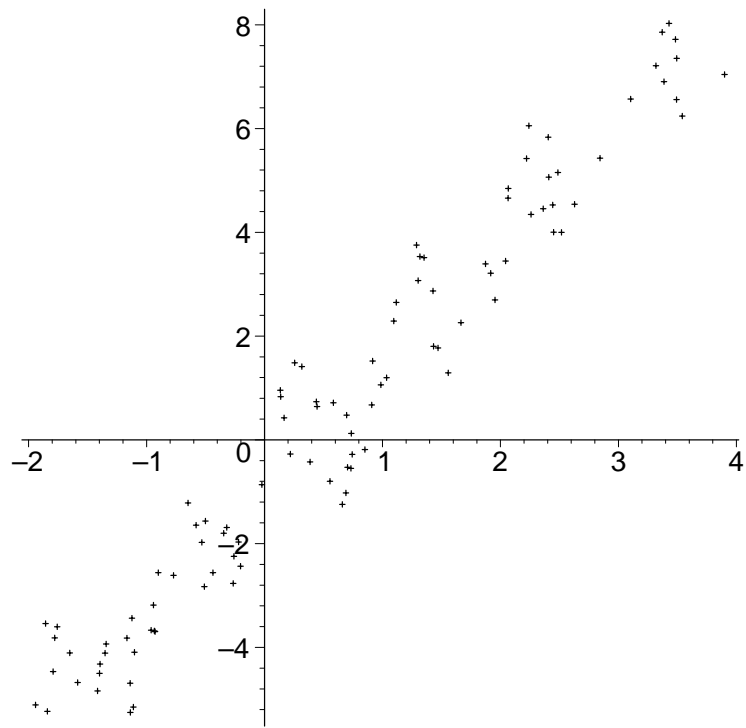
And finally a degree 1 polynomial

```
> do_approx(1, curve);  
Approximation Error: 2.529822
```



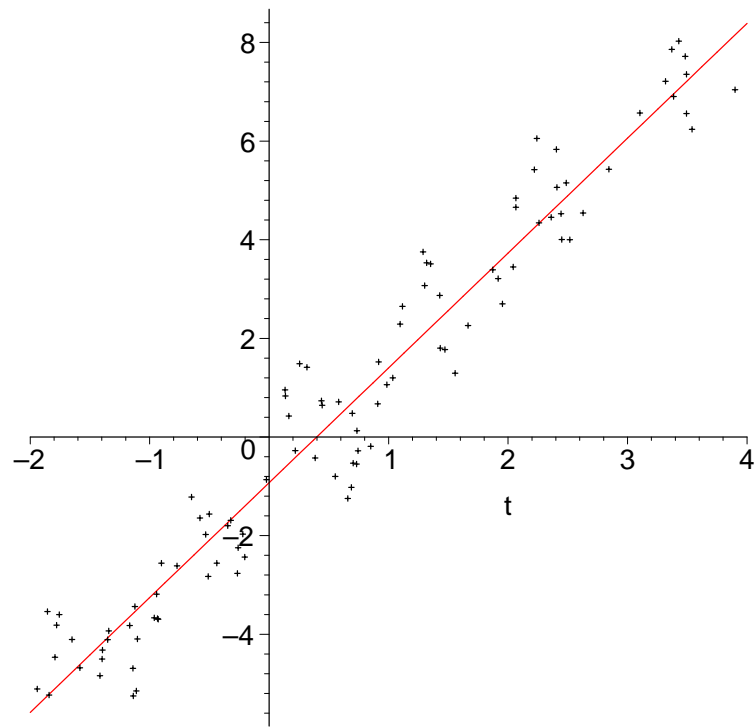
[ Now we generate a lot of data points based on the function  $f$  shown below

```
[ > pert:=rand(-100..100)/100:  
[ > f:=t->2.3*t-1+sin(2*Pi*t)+pert():  
[ > ff:=t->[t,evalf(f(t))]:  
[ > x_pts:=convert(RandomVector(100,generator=-2..4.0),list):  
[ > points:=map(ff,x_pts):  
[ > graph_points:=listplot(points,style=POINT):  
  display(graph_points);
```



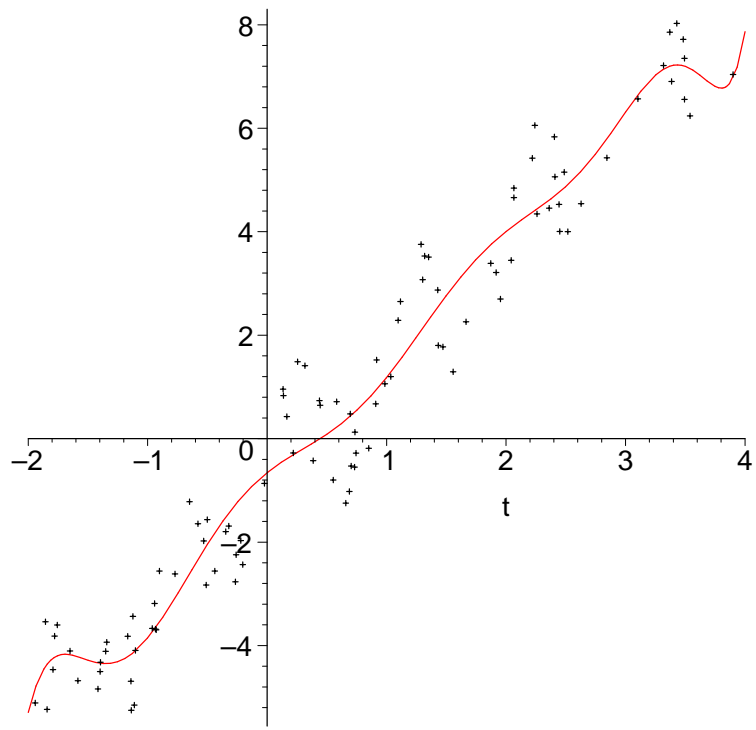
Next we approximate with a linear polynomial:

```
> do_approx(1, cur);  
Approximation Error: 8.832658
```



Finally we approximate with a polynomial of degree 10:

```
> do_approx(10, cur);  
Approximation Error: 8.087864
```



```
[ >  
[ >
```