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;

\[
f(t) = 2 - t^2 + \frac{1}{2} t^3
\]

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

\[
ff(t) = [t, f(t)]
\]

> x_pts:=-1,1,2,3;

\[
x_{\text{pts}} := [-1, 1, 2, 3]
\]

> points:=map(ff,x_pts);

\[
\begin{bmatrix}
-1, \frac{1}{2} \\
1, \frac{3}{2} \\
2, 2 \\
3, \frac{13}{2}
\end{bmatrix}
\]

> 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

\[ f \_ \text{model} := (t, n) \rightarrow \sum_{j=0}^{n} a[j] \cdot t^j \]

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

\[ \text{eqs} := [\text{seq}(f \_ \text{model}(x \_ \text{pts}[i], N) = \text{points}[i][2], i=1..\text{nops(x \_ \text{pts}))}] \]

Rewrite the equations in terms of matrices

\[ (A, b) := \text{GenerateMatrix(eqs, [seq(a[j], j=0..N)])} \]

Find the least squares solution to the problem:

\[ \text{sol} := \text{MatrixInverse(Transpose(A) . A) . Transpose(A) . b} \]

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

\[ f \_ \text{sol} := t \rightarrow \sum_{j=1}^{\text{Dimension(sol)}} 's[j][j-1]' . t^{j-1} \]
Next, we put all of the above together in a function:

```plaintext
> 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)]);
   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

```plaintext
> do_approx(3,curve);
```

Approximation Error: 0
Then we model with a degree 2 polynomial

\[
> \text{do\_approx}(2,\text{curve});
\]

Approximation Error: 1.144155
And finally a degree 1 polynomial

```haskell
> do_approx(1, curve);
Approximation Error: 2.529822
```
Now we generate a lot of data points based on the function $f$ shown below

```maple
> 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:

```plaintext
> do_approx(1, cur);
Approximation Error: 8.832658
```
Finally we approximate with a polynomial of degree 10:

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