Linear Regression

Say we have a set of functions and wish to build a curve from some linear combination of these functions in such a way
that we pass through a collection of given data point. Finding such a curve amounts to solving a linear system, say $A\,x = b$.
In general, our collection of functions may not be able to accomodate our data points and so we settle for the "best" we can
do. By best we mean a curve that minimizes the sum of squares of errors in outputs. In other words, we settle for a least
squares solution and solve $A^*A\,x = A^*b$ (the normal equations) instead.

The code below tries to fit a curve belonging to the span of myBasis through the points in myData.
If no such curve fits, the least squares approximation is used.

The code also produces a plot of myData, the regression curve, and line segments showing how far off the approximation is.
In addition the "coefficient of determination" (i.e. R^2) is computed. This value is rigged so that $R^2 = 1$ means a perfect fit!

Keep it real: myData is assumed to be lists of pairs of real numbers. No complex stuff here please.

```
> restart;
  with(LinearAlgebra):
  with(plots):
  with(plottools):

  myRegress := proc(myData,myBasis)
     local A,i,j,b,X,k,f,ptPlot,fPlot,meanObs,varObs,err;

     # create a matrix with a row per data point and column per basis
  function.
     A := Matrix(nops(myData),nops(myBasis)):

     # plug the x coordinate of the i-th data point into the j-th basis
  function
     # to form the (i,j)-entry of the matrix A.
     for i from 1 to nops(myData) do
        for j from 1 to nops(myBasis) do
           A[i,j] := subs(x=myData[i][1],myBasis[j]);
        end do;
     end do:

     print('A' = A);

     # the target outputs are the the y coordinates of our data points.
     b := Matrix(nops(myData),1):

     for i from 1 to nops(myData) do
        b[i,1] := myData[i][2];
     end do:
```

```
   print('b' = b);

   # We cannot use...
   # X := evalf((Transpose(A).A)^(-1).Transpose(A).b);
   # ...in certain underdetermined cases.

   # solve the normal equations (& plug 0 into any free variables)
   X := evalf(<subs({seq(s[k]=0,k=1..nops(myBasis))},LinearSolve
(<Transpose(A).A|Transpose(A).b>,free='s'))>):

   print('X' = X);

   # build the best fit function
   f := sum('X[k,1]'*myBasis[k],k=1..nops(myBasis));

   print('f' = f);

   # data points plotted in blue
   ptPlot := pointplot(myData,color=blue):

   # best fit curve plotted in black
   fPlot := plot(f,x=min(seq(myData[i][1],i=1..nops(myData)))..max(seq
(myData[i][1],i=1..nops(myData))),color=black):

   # the mean and variance of the y-values of data points
   meanObs := sum(myData[k][2],k=1..nops(myData))/nops(myData):
   varObs := sum((myData[k][2]-meanObs)^2,k=1..nops(myData)):

   # the sum of squares of errors
   err := sum((myData[k][2]-subs(x=myData[k][1],f))^2,k=1..nops
(myData)):

   print(R^2 = 1-err/varObs);

   # plot data points, best fit curve, and lines connecting them
together...
   return display({ptPlot,fPlot,seq(line(myData[k],[myData[k][1],subs
(x=myData[k][1],f)],color=red,thickness=3),k=1..nops(myData))});
end proc:
```

Example: Let's do a best fit quadratic.

```
> myData := [[-2,3],[0,4],[1,2],[3,13]];
myBasis := [1,x,x^2];

myRegress(myData,myBasis);
```

$$myData := [[-2, 3], [0, 4], [1, 2], [3, 13]]$$

$$myBasis := [1, x, x^2]$$

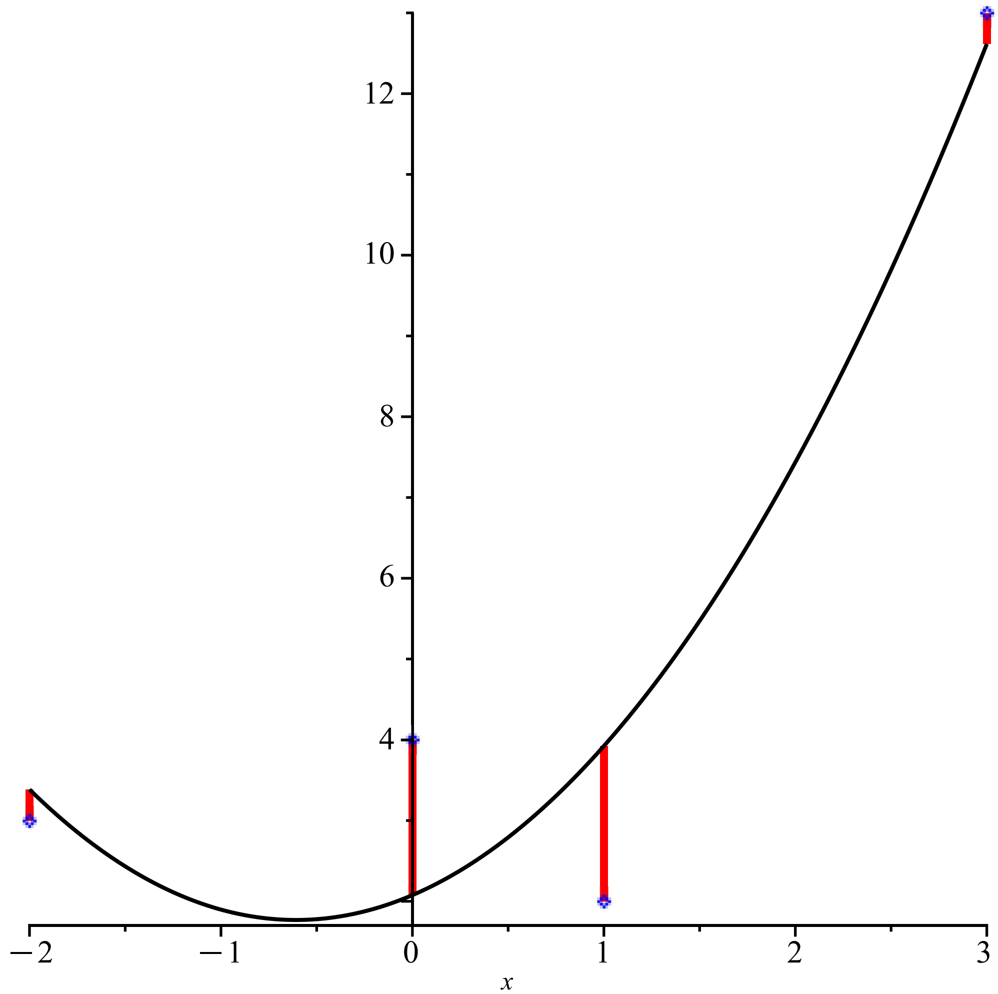
$$A = \begin{bmatrix} 1 & -2 & 4 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 3 & 9 \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \\ 4 \\ 2 \\ 13 \end{bmatrix}$$

$$X = \begin{bmatrix} 2.076923077 \\ 1.012820513 \\ 0.8333333333 \end{bmatrix}$$

$$f = 2.076923077 + 1.012820513\, x + 0.8333333333\, x^2$$

$$R^2 = 0.9000999001$$



Example: Let's fit to a function of the form $f(x) = a \sin(x) + b \cos(x) + c \sin(2x) + d \cos(2x)$

```
> myData := [[-2,3],[0,4],[1,2],[3,13]];
  myBasis := [sin(x),cos(x),sin(2*x),cos(2*x)];

  myRegress(myData,myBasis);
```

$$myData := [[-2, 3], [0, 4], [1, 2], [3, 13]]$$

$$myBasis := [\sin(x), \cos(x), \sin(2x), \cos(2x)]$$

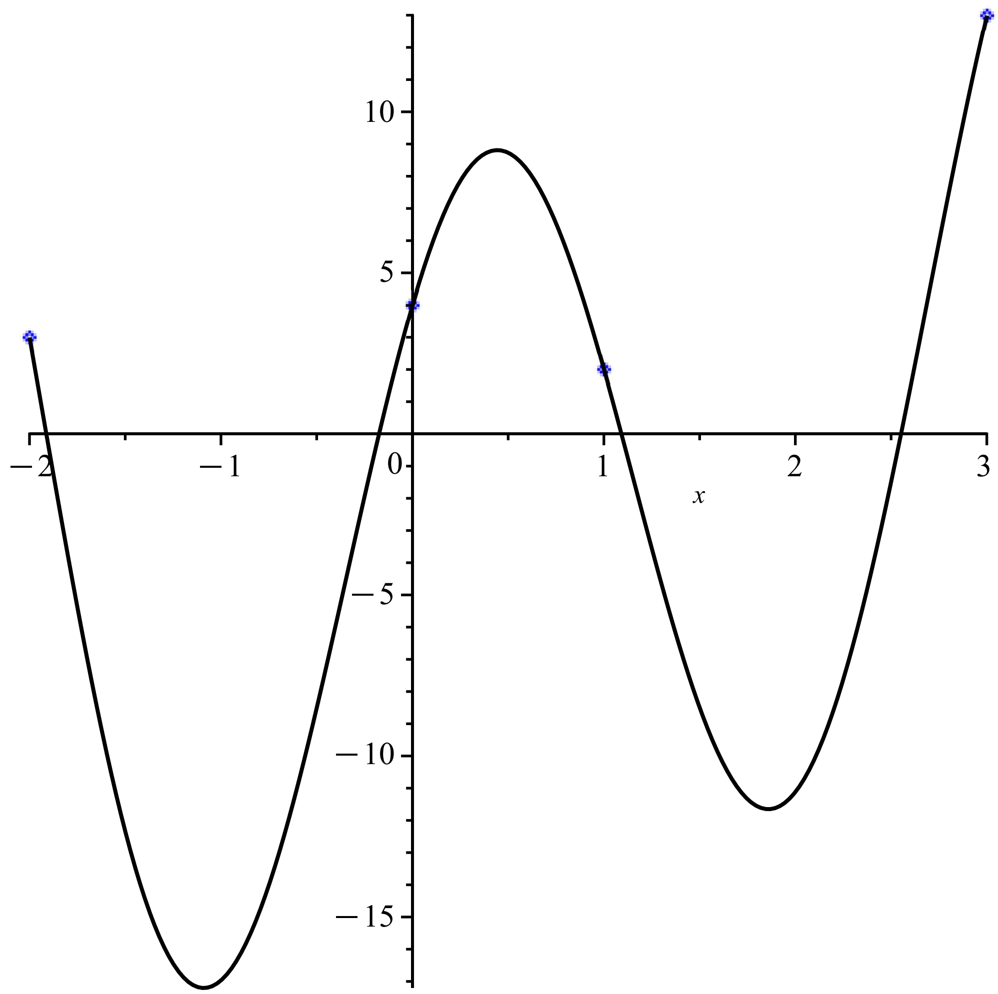
$$A = \begin{bmatrix} \sin(-2) & \cos(-2) & \sin(-4) & \cos(-4) \\ \sin(0) & \cos(0) & \sin(0) & \cos(0) \\ \sin(1) & \cos(1) & \sin(2) & \cos(2) \\ \sin(3) & \cos(3) & \sin(6) & \cos(6) \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \\ 4 \\ 2 \\ 13 \end{bmatrix}$$

$$X = \begin{bmatrix} 0.5182438048 \\ -6.084957906 \\ 9.951037722 \\ 10.08495791 \end{bmatrix}$$

$$f = 0.5182438048 \sin(x) - 6.084957906 \cos(x) + 9.951037722 \sin(2x) + 10.08495791 \cos(2x)$$

$$R^2 = 1.$$



Example: Let's fit to a function of the form $f(x) = a \sin(x) + b \cos(x) + c\, e^x$

```
> myData := [[-2,3],[0,4],[1,2],[3,13]];
  myBasis := [sin(x),cos(x),exp(x)];

  myRegress(myData,myBasis);
```

$$myData := [[-2, 3], [0, 4], [1, 2], [3, 13]]$$

$$myBasis := [\sin(x), \cos(x), e^x]$$

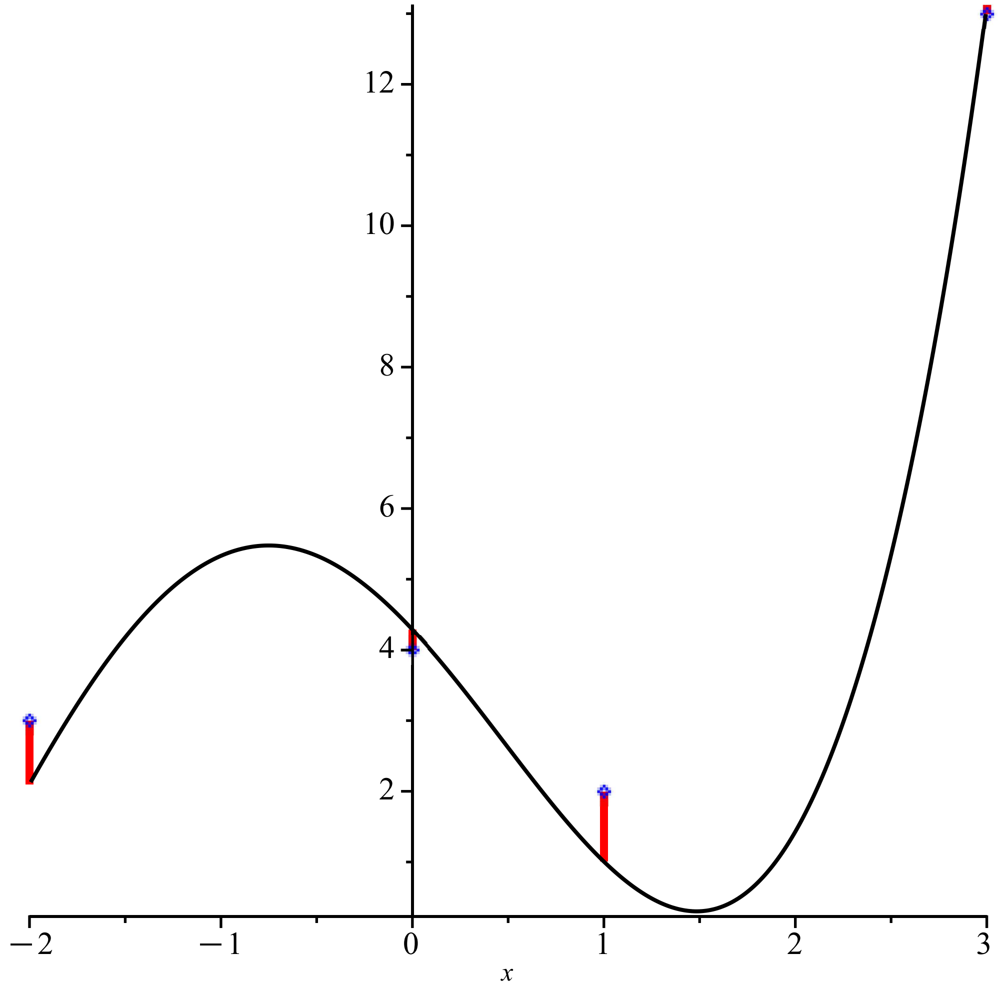
$$A = \begin{bmatrix} \sin(-2) & \cos(-2) & e^{-2} \\ \sin(0) & \cos(0) & e^0 \\ \sin(1) & \cos(1) & e \\ \sin(3) & \cos(3) & e^3 \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \\ 4 \\ 2 \\ 13 \end{bmatrix}$$

$$X = \begin{bmatrix} -3.756078144 \\ 3.438116984 \\ 0.8493708261 \end{bmatrix}$$

$$f = -3.756078144 \sin(x) + 3.438116984 \cos(x) + 0.8493708261\, e^x$$

$$R^2 = 0.9753539183$$



Example: Here's a best fit quadratic...but it is underdetermined (not "enough" data)

```
> myData := [[-2,3],[0,4]];
  myBasis := [1,x,x^2];

  myRegress(myData,myBasis);
```

$$myData := [[-2, 3], [0, 4]]$$

$$myBasis := [1, x, x^2]$$

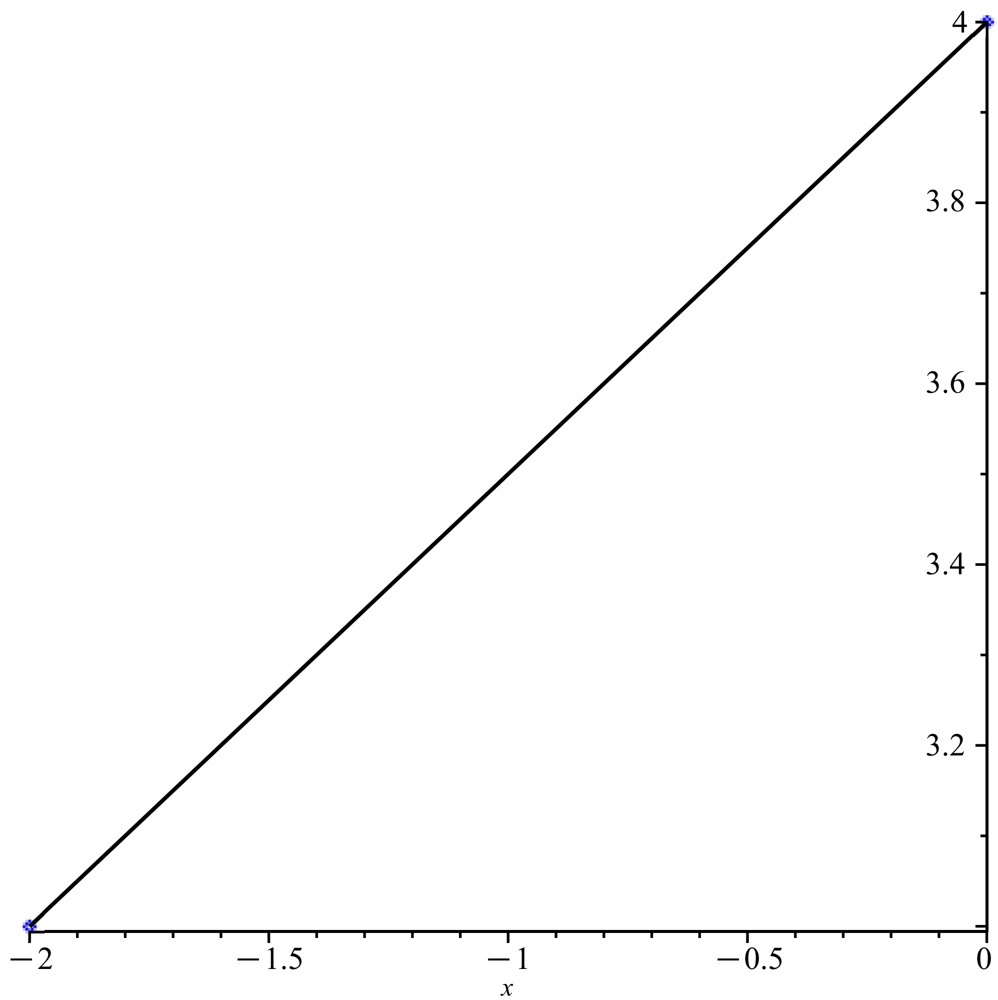
$$A = \begin{bmatrix} 1 & -2 & 4 \\ 1 & 0 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$X = \begin{bmatrix} 4. \\ 0.5000000000 \\ 0. \end{bmatrix}$$

$$f = 4. + 0.5000000000\, x$$

$$R^2 = 1.$$



Example: This produces an animation of successively higher order polynomials attempting to fit the data.

```
> myData := [[-2,6],[-1,5],[0,2],[1,3],[2,0],[3,2],[4,1]];
  myBasis := [1,x];

  for m from 0 to 6 do
     myBasis := [seq(x^k,k=0..m)];

     A := Matrix(nops(myData),nops(myBasis)):

     for i from 1 to nops(myData) do
        for j from 1 to nops(myBasis) do
           A[i,j] := subs(x=myData[i][1],myBasis[j]);
        end do;
     end do:

     'A' = A;

     b := Matrix(nops(myData),1):

     for i from 1 to nops(myData) do
        b[i,1] := myData[i][2];
     end do:

     'b' = b;
```

```
   # X := evalf((Transpose(A).A)^(-1).Transpose(A).b);
   X := evalf(<subs({seq(s[k]=0,k=1..nops(myBasis))},LinearSolve
(<Transpose(A).A|Transpose(A).b>,free='s'))>);

   f := sum('X[k,1]'*myBasis[k],k=1..nops(myBasis));

   ptPlot := pointplot(myData,color=blue):
   fPlot := plot(f,x=min(seq(myData[i][1],i=1..nops(myData)))..max(seq
(myData[i][1],i=1..nops(myData))),color=black):

   meanObs := sum(myData[k][2],k=1..nops(myData))/nops(myData):
   varObs := sum((myData[k][2]-meanObs)^2,k=1..nops(myData)):
   err := sum((myData[k][2]-subs(x=myData[k][1],f))^2,k=1..nops
(myData)):
   R^2 = 1-err/varObs;

   myTitle := cat("Degree = ",m," and R^2 = ",1-err/varObs);

   myPlot[m] := display({ptPlot,fPlot,seq(line(myData[k],[myData[k]
[1],subs(x=myData[k][1],f)],color=red,thickness=3),k=1..nops(myData))}
,title=myTitle);
end do:

display([seq(myPlot[m],m=0..6)],insequence=true);
```

$$myData := [[-2, 6], [-1, 5], [0, 2], [1, 3], [2, 0], [3, 2], [4, 1]]$$

$$myBasis := [1, x]$$

Degree = 6 and R^2 = 1.