

Interpolation and Extrapolation

Mth 351 Aug 7 2001 Maple 6

Bent E. Petersen

Filename: 351u2001_assignment_05_interpolation_and_extrapolation.mws

Assignment solutions are due Aug 14, 2001.

Assignment: This Maple worksheet uses Egyptian cereal imports from 1970 through 1989 to illustrate interpolation and extrapolation. The worksheet contains a few problems which make up Summer 2001 Mth 351 Assignment 5. You can use Maple to do the assignment (recommended) or, if you are reading the PDF version of this document, you can extract the data and do the calculations laboriously with a calculator. Or you can write a little program in C or whatever. Above all, think, be creative, and write some insightful comments.

Acknowledgment: The Egyptian cereal imports data below are from table 3.5 in Sandra Lach Arlinghaus, ed., *Practical Handbook of Curve Fitting*, CRC Press, Boca Raton 1994. The data in that table are from the World Bank. The World Bank publishes an immense amount of world data - see for example, <http://www.worldbank.org/data/> .

Figure 3.6 in the Arlinghaus text show a linear and an exponential fit to the Egyptian data. While the authors of the text have much to say about the art of curve fitting and prediction the only one I can agree 100 % with is, "Any of an infinite number of curves may be used to fit data; the art comes in making reasonable selections." Yes, indeed!

The crazier curve fits below are mine (and Maple's) and I take responsibility for them. I particularly like the prediction that Egypt in 1992 will (so 'did' from our perspective) export more cereal than the world has produced throughout all of history!

```
[ > restart;
[ > with(stats): with(fit):
[ > with(plots):
Warning, the name changecoords has been redefined
```

Here's some "real world" data, cereal imports, in thousands of tons, by Egypt from 1970 through 1989. First the years

```
[ > Tdata := [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89];
[ Tdata := [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89]
```

Now here is the quantity of cereal imported in the corresponding years

```
> Ydata := [1305, 2447, 1773, 1872, 3877, 4214, 4347, 4935, 5852, 5400, 6027, 7199, 6799, 8114, 8616, 8903, 8407, 9348, 8500, 8542];
```

```
Ydata := [1305, 2447, 1773, 1872, 3877, 4214, 4347, 4935, 5852, 5400, 6027, 7199, 6799, 8114, 8616, 8903, 8407, 9348, 8500, 8542]
```

In Maple it is simple to find the least squares linear fit (this uses the stat library, a standard component of Maple, loaded above)

```
> eqn1 := leastsquare[{t, y}, y = m*t + b, {m, b}]([Tdata, Ydata]);
```

$$eqn1 := y = \frac{16407}{38} t - \frac{5415251}{190}$$

Because we loaded the plots library we can save up plots and combine and display them later. (This library provides other tools as well.) Note eqn1 is a label for the equation of the line - what we want to plot is the expression in t that forms the right-hand side of the equation. We use the rhs() function to do so

```
> img1 := plot(rhs(eqn1), t = 70..92, color = red);
```

We will want to display the original data points so we first built a list of points from the original data

```
> points := [seq([Tdata[k], Ydata[k]], k = 1..20)];
```

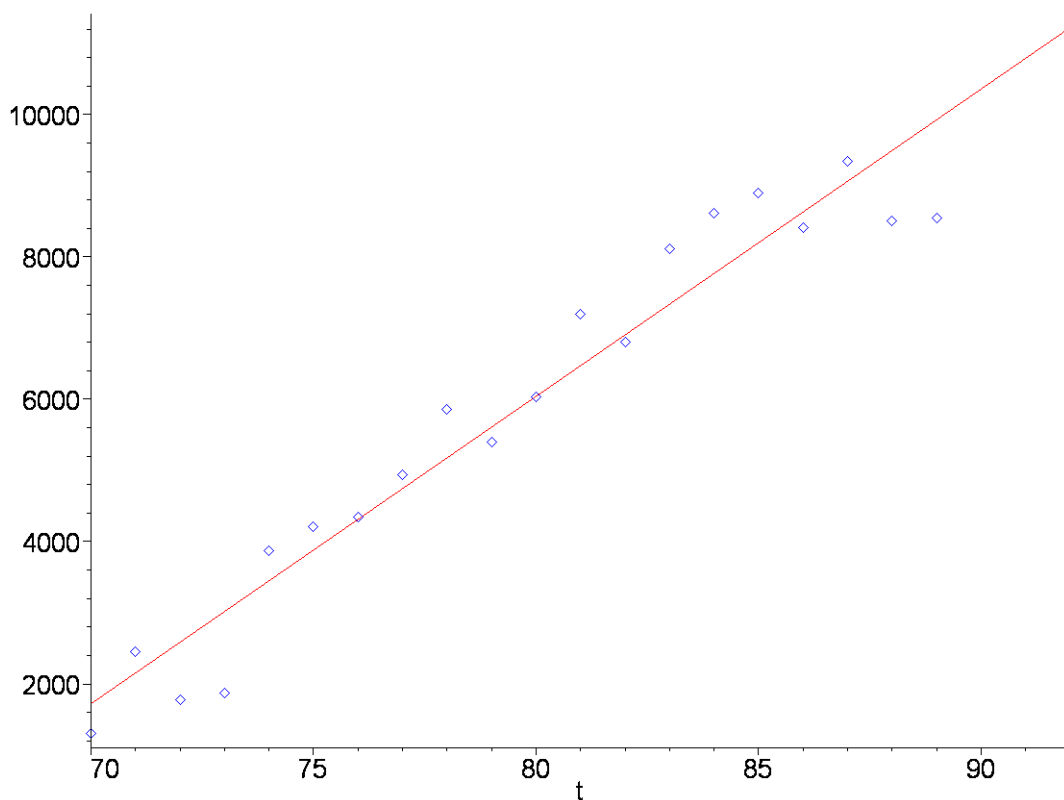
```
points := [[70, 1305], [71, 2447], [72, 1773], [73, 1872], [74, 3877], [75, 4214], [76, 4347], [77, 4935], [78, 5852], [79, 5400], [80, 6027], [81, 7199], [82, 6799], [83, 8114], [84, 8616], [85, 8903], [86, 8407], [87, 9348], [88, 8500], [89, 8542]]
```

Now we plot the points (and save the plot for later)

```
> pntplot := pointplot(points, color = blue, symbolsize = 20, symbol = diamond);
```

Let's display our two plots to see how well our least squares line fits the data

```
> display({img1, pntplot});
```



Not too shabby!

We can use our fitted line to "predict" the imports for 1992. This kind of extrapolation is open to criticism of course!

```
> subs (t=92, rhs (eqn1)) ;
```

```
      2131969
      -----
         190
```

That is excessively precise! Let's round it to an integer

```
> round (subs (t=92, rhs (eqn1))) ;
```

```
      11221
```

You can also just evaluate the fraction as a floating point number with 4 or 5 digits of precision

```
> evalf (subs (t=92, rhs (eqn1)), 5) ;
```

```
      11221.
```

Actually given the data and the wholly unjustified extrapolation we should probably only use 4 digits

```
> evalf (subs (t=92, rhs (eqn1)), 4) ;
```

11220.

Undoubtedly you are very unhappy about the terminating decimal point (which indicates a floating point number here). You can get rid of it by rounding!

```
> round(evalf(subs(t=92, rhs(eq1)), 4));
```

11220

Now it is clear what to do if you want only 3 digits of precision and an integer result

```
> round(evalf(subs(t=92, rhs(eq1)), 3));
```

11200

Note instead of assigning `rhs(eq1)` to a variable (that is, a label) I keep using it over and over above. You may think this approach is wasteful because Maple has to evaluate the expression each time. This is not the case. Maple remembers the values of expressions it has already evaluated, and simply reuses them. Thus you can free your mind to play with the mathematics without having to worry about grubby lower order stuff like efficiency.

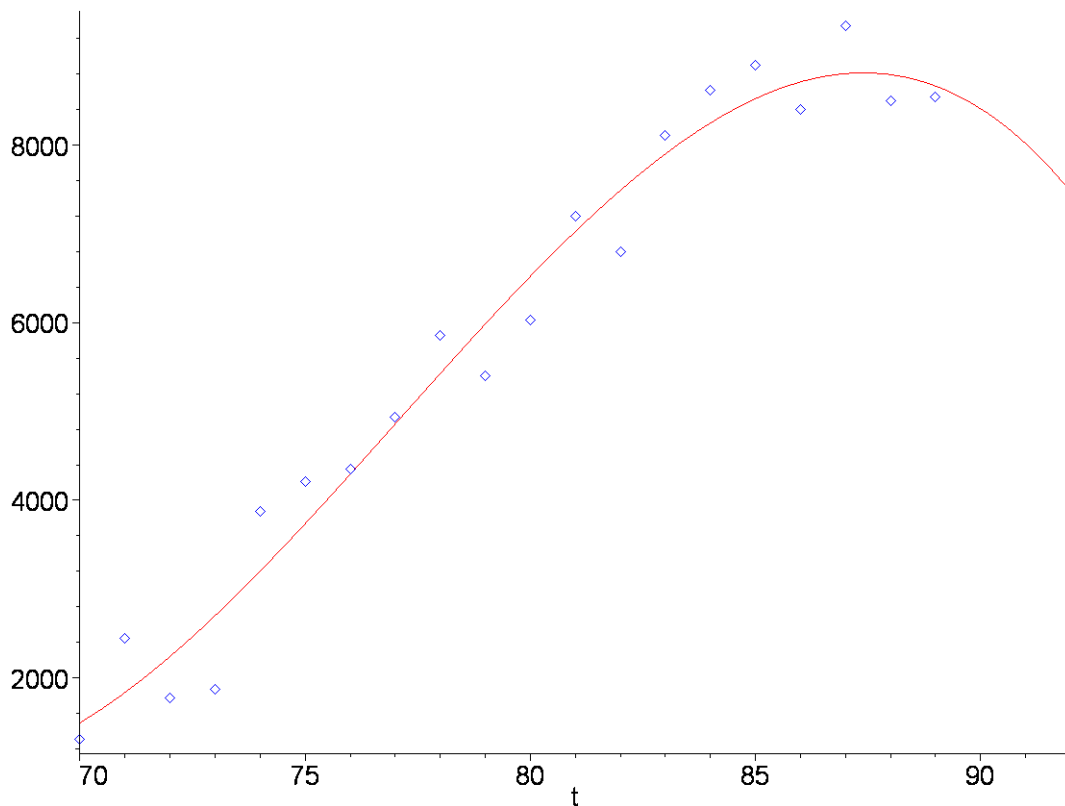
Consider now a least squares cubic polynomial fit to our data

```
> eqn3:=leastsquare[{t,y},y=a*t^3+b*t^2+c*t+d,{a,b,c,d]}([Tdata,Ydata]);
```

$$eqn3 := y = -\frac{2605885}{1470942}t^3 + \frac{160110}{391}t^2 - \frac{45571003763}{1470942}t + \frac{41106922627}{53295}$$

```
> img3:=plot(rhs(eqn3),t=70..92,color=red):
```

```
> display({img3,pntplot});
```



Nice! Let's use our fitted cubic to "predict" the imports for 1992. This extrapolation is even more open to criticism than the linear one above!

```
> round(evalf(subs(t=92,rhs(eq3)),3));
          7480
```

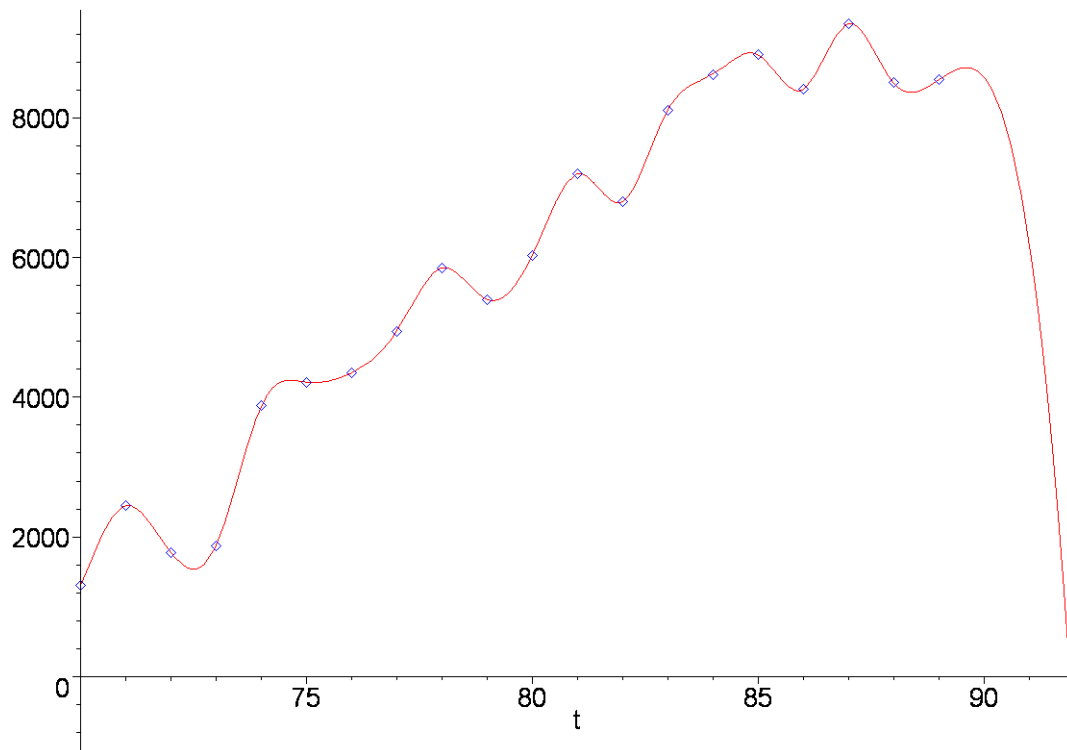
If you were in charge of formulating policy for Egypt based on predicted cereal imports, you'd have a headache now. You'd certainly realize that either a solid realistic underlying model is needed, or you need to rely on experience and judgment in conjunction with mathematics.

Let's look now at a natural cubic spline fit

```
> spl:=spline(Tdata,Ydata,t,cubic):
```

I suppressed the output here because it is rather long. It is a piecewise cubic expression in the specified variable t.

```
> imgspl:=plot(spl,t=70..92,color=red):
> display({imgspl,pntplot});
```



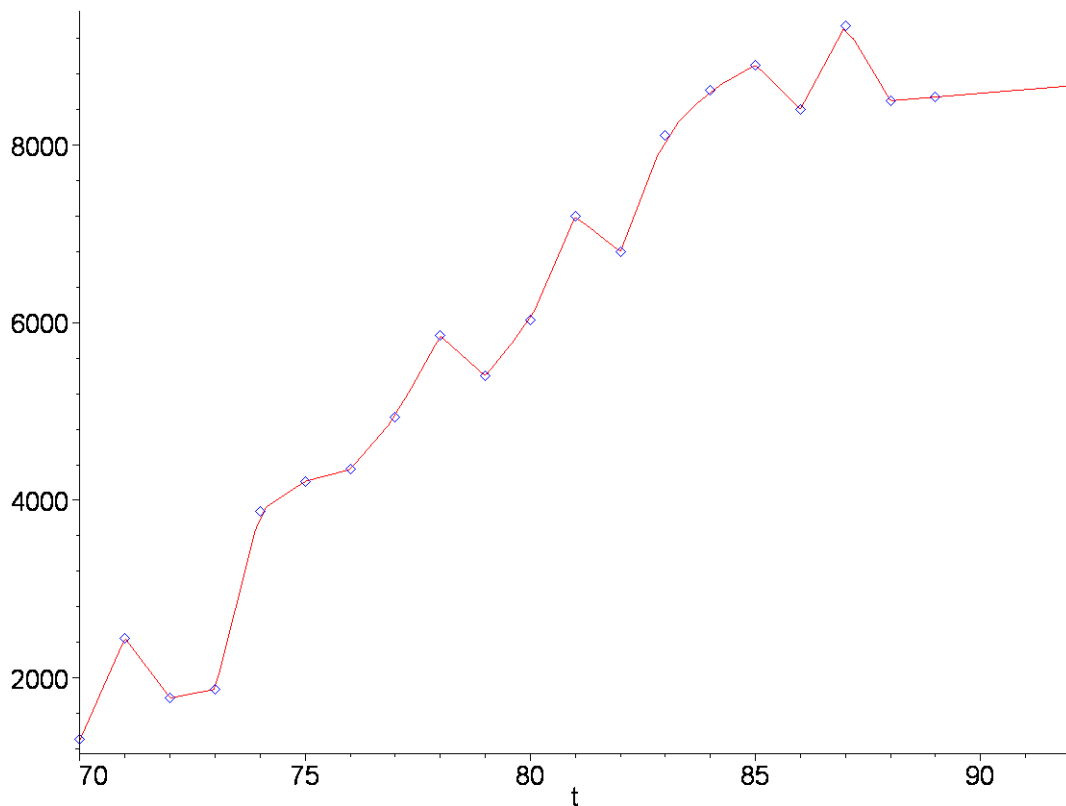
You can see how the spline passes through all the data points without excessive curvature. Again we can make a prediction for 1992

```
> round(evalf(subs(t=92,spl),3));
-904
```

Wow! We have a solid (?) prediction that Egypt will export cereal in 1992. From the Egyptian point of view this prediction is much nice than the ones above.

Perhaps by now you are not prepared to accept anything but a piecewise linear relationship - one that makes no assumptions about the data. Let's try it.

```
> lin:=spline(Tdata,Ydata,t,linear):
> imglin:=plot(lin,t=70..92,color=red):
> display({imglin,pntplot});
```



What about prediction?

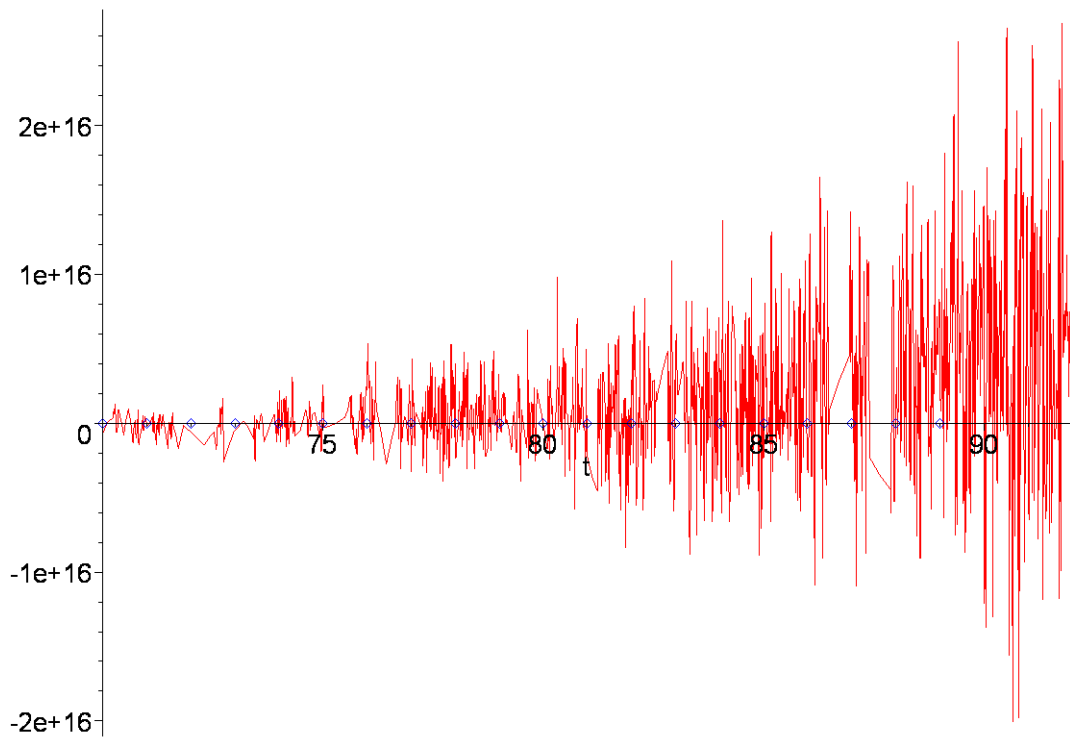
```
> round(evalf(subs(t=92, lin), 3));
      8670
```

Pretty reasonable perhaps. Of course one can argue that this prediction makes use only of the last two data points and no use whatsoever of the earlier history. That may, or may not, be reasonable. We're back to judgment! There's no way around it - simple calculation will not replace thought.

Problem 1: Construct the least squares polynomial of degree 2. Make the 1992 prediction again. Note if you do this by hand, you will need to solve 3 linear equations in 3 unknowns. Do you think you may be skeptical in the future about predictions based on trends (rather than models).

As our final example let's try an interpolation polynomial

```
> poly:=interp(Tdata,Ydata,t):
> imgpoly:=plot(poly,t=70..92,color=red):
> display({imgpoly,pntplot});
```

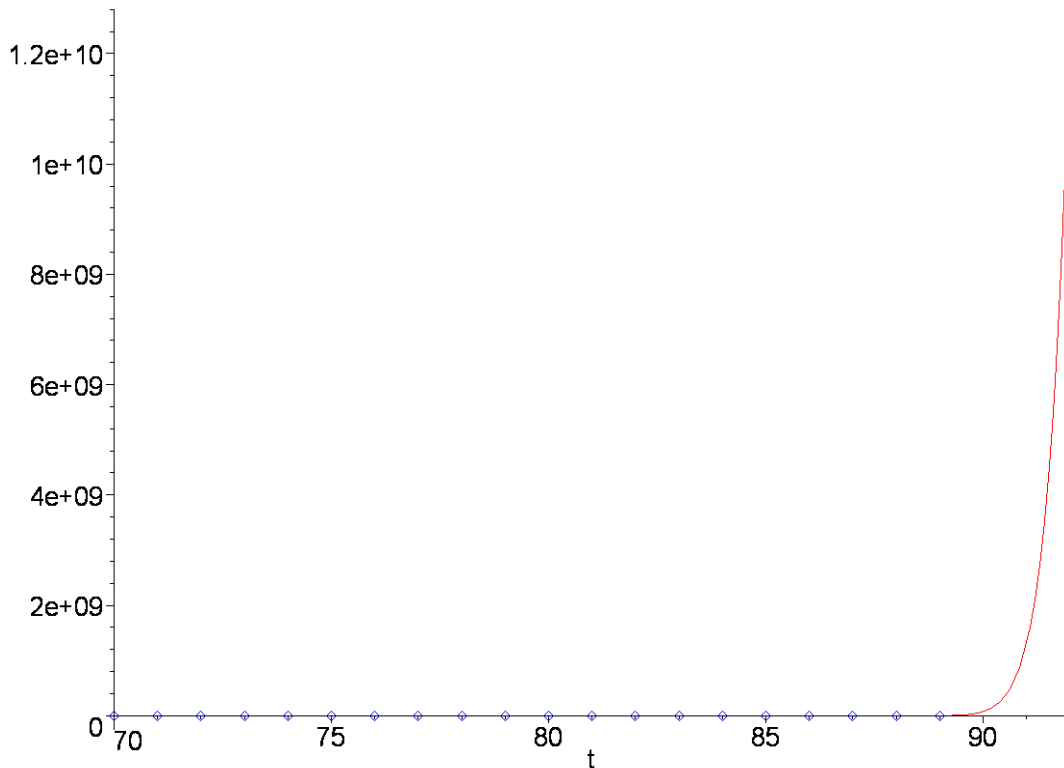


Wow!

Problem 2: What happened? A polynomial of degree 19 can not oscillate that much.

It may help you to know that Maple uses floating point to calculate the points to plot. The default precision is 10 decimal digits. Let's see what happens if we temporarily increase the precision to 30 decimal digits before computing the points to plot.

```
> Digits:=30:imgpoly:=plot(poly,t=70..92,color=red):Digits:=10:
> display({imgpoly,pntplot});
```



What about the predicted cereal import for 1992?

```
> round(evalf(subs(t=92,poly),3));
12600000000
```

Hmm. That is a lot of cereal.

Let's take a look at the interpolation polynomial, but limit the precision.

```
> poly2:=evalf(sort(poly),10);
poly2 := .1259435846 10-9 t19 - .1885446109 10-6 t18 + .0001336522180 t17 - .05962908647 t16
+ 18.77184531 t15 - 4430.484924 t14 + 812989.6853 t13 - .1186918793 109 t12
+ .1399059946 1011 t11 - .1343209285 1013 t10 + .1054711622 1015 t9 - .6773391769 1016 t8
+ .3542989816 1018 t7 - .1496275700 1020 t6 + .5027518408 1021 t5 - .1313356978 1023 t4
+ .2572198282 1024 t3 - .3554589428 1025 t2 + .3091650281 1026 t - .1273241211 1027
```

Here the `sort()` command arranges the terms in increasing order by degree. The `evalf(,8)` command converts the coefficients to floating point using 8 decimal digits of precision. Without it we would get an immense exact expression.

We have only changed the coefficients a little bit. What happens to our prediction for 1992?

