Harald Nordgren
900404-1936
ada09hno@student.lth.se

FMN011 – Project 1

I have used least squares to analyze unemployment in France over the years, using test data provided to us which describes the number of unemployed workers monthly from 1983 to 2014. Given the vast amount of data points, any polynomial model of reasonable degree – as in less than degree 377 – will form an over-determined system where the most productive approach is going to be an approximate fit. The least squares method aims to do this by minimizing the error per data point.

The `xls` file was imported to Matlab via the handy `xlsread` function, after which I split the data into corresponding x and y vectors:

```
DATA = xlsread('frenchUnemployment.xlsx');
data_size = size(DATA);
interval = 1:data_size(1);

data_x = DATA(interval,1);
data_y = DATA(interval,2);
```

To fit the data to different models I used `cftool` – Matlab's curve fitting tool – which works both for regular polynomial models and trigonometric polynomials, up to degree 9. At first I struggled with the `polyfit` and `polyval` functions – because the input data was considered ill-conditioned – which I got around by using `cftool` and its data normalization functionality.

I generated code with the fitting tool, for four different models. Two regular polynomial models of degrees 4 and 9, and two trigonometric polynomials models both of degree 8 with a quarterly and yearly period. The code for the regular polynomial of degree 4 looks like this: (`'poly4'` is changed to `'poly9'` for the other regular polynomial model)

```
fitresult = fit(data_x, data_y, fittype('poly4'), 'Normalize', 'on');
plot(fitresult, data_x, data_y);
```

I drew from the data sample, and from an intuitive sense of the problem, that the periodicity of the trigonometric model should be tied to the yearly cycle. Thus, I tried to fix this value in `cftool` while allowing the other variables to float freely. For the quarterly model I used

```
opts = fitoptions('Method', 'NonlinearLeastSquares');

opts.Lower = [-Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf
-Inf -Inf -Inf -Inf 1.5707];

opts.Upper = [Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
```

```
Inf 1.5709];

opts.Normalize = 'on';
fitresult = fit(data_x, data_y, fittype('fourier8'), opts);
```

where the infinite values of the `Lower` and `Upper` vectors mean that no value is actually imposed on the model. And the numerical value at the end of both vector aim to get close to `2*pi/4 = 1.5708`, without actually having exactly the same value in both parameters which Matlab won't allow. The yearly model has the same code except that I aim to get close to the value `2*pi/12 = 0.5236` instead. Both models were plotted like the regular polynomials, and all four models then had a legend and axis labels added. (See last page)

I found that for the regular polynomials, a higher degree generally gave a better fit, although they spiral out of control very quickly outside of the dataset interval meaning that they will be less useful for interpolating any future values.

It's apparent from the graphs that the regular polynomial of degree 4 doesn't really model the data in a satisfying way. The one of degree 9 does a much better job, although it's still not close to the precision of the trigonometric models. The trigonometric polynomials both with yearly and quarterly period capture the general trends of the data in a good way, where outliers have much less of an impact, and the quarterly model even seems to be pretty stable at its end values. Unemployment in France seems be definitely affected by the seasons.

This project made me more comfortable with using Matlab, and especially its help pages. I'm starting to understand Matlab more as a programming language, and I'm discovering the power of writing scripts.

	RMSE	Execution time
Polynomial degree 4	247	0.5 s
Polynomial degree 9	165	0.6 s
Trigonometric polynomial degree 8,quarterly period	134	0.6 s
Trigonometric polynomial degree 8, yearly period	135	0.6 s