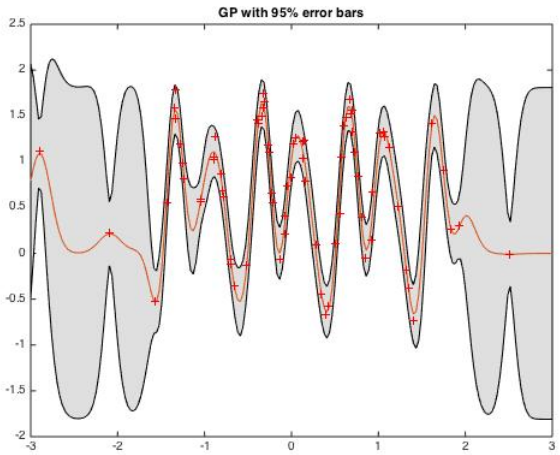
Question A)

Below I have displayed the function plotted with the 95% error bars and the key commands for the task in the code listing at the end of this document.



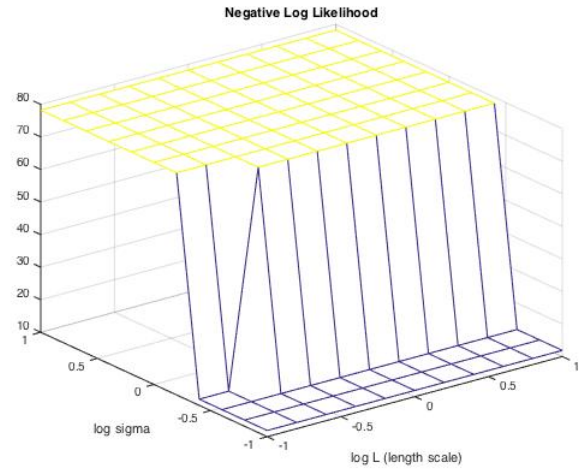
Comment on the predictive error bars and hyper-parameters:

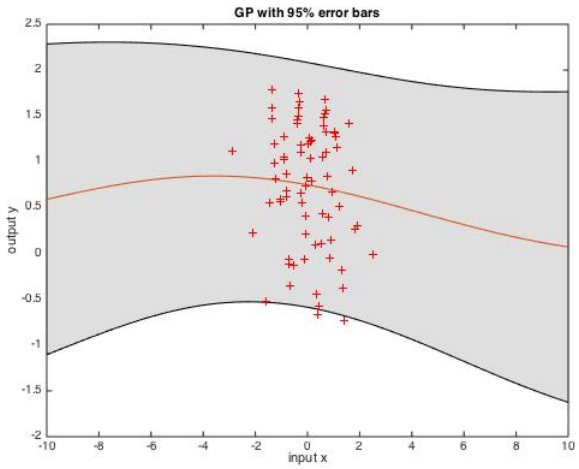
After minimising the negative log marginal likelihood, the length scale l was set to 0.1282 and the signal standard deviation σ was set to 0.8970. (NB. These are the parameters of the kernel function: $k(x^p, x^q) = \sigma^2 exp(-(x^p - x^q)^T(x^p - x^q)/2l^2)$.) The noise standard deviation was 0.1178.

I interpret these parameters as follows: the small length scale suggests that the data be explained by a function that is very wiggly with little correlation between distant points. This makes sense for if we look at the graph above, points that are very close together tend to be somewhat correlated, but points that are further apart (a distance of 1 away) tend to barely correlated at all.

The standard deviation of the noise is quite small at 0.1 , whereas the standard deviation of the signal is about 10 times higher. There seems to be little noise for points that are close together in x value are similarly close in y value. However, we would expect the signal standard deviation to be higher (at about 1) for as we can see most of the y values fall within a range of 1.8 and -1.8 or roughly which covers the 95% confidence interval.

Question B)





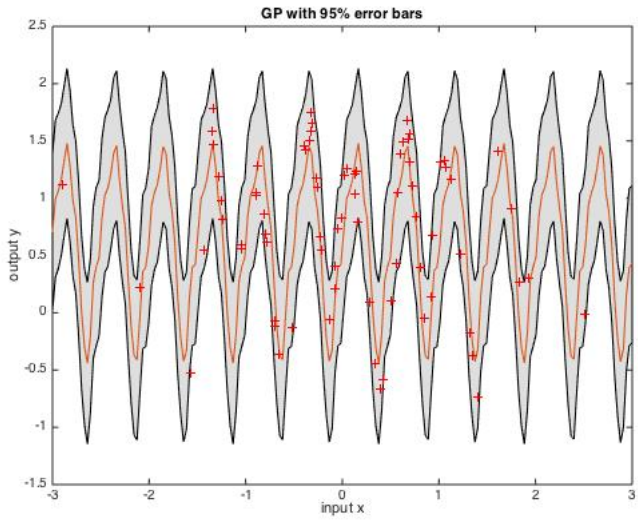
(2)

Try a range of values. Show the fit. Explain what is going on. Which fit is best, and why?

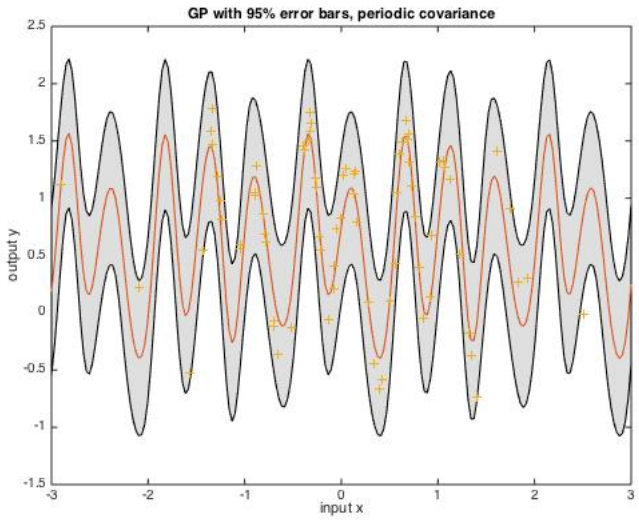
At the initialisation stage, I varied the standard deviation of the signal noise from e^{-1} to e^1, and similarly for the length scales. (I also separately varied the other noise but found little effect on the optimum). What I found is that depending on the parameters, we could reach one of two local minima after optimisation. The second minima has a length scale of 8.3488 and a signal standard deviation of 0.6989. The noise standard deviation is 0.6632. Because the length scale is very large, this means the model is trying to explain the spread of points through noise rather than through covariance with neighbouring points.

The first model has a higher log likelihood (-11 rather than -78) and this should give us evidence to believe that it is a better model. To my eye there is a good degree of covariance in closely located data points. There are quite a few seemingly almost vertical lines of points. This cannot be purely accidental, as the new model would suggest. The old model is better.

Question C)



(1)



(2)

Comment on the behaviour of the error-bars, compared to your fit from (a). Do you think the data generating mechanism was really periodic? Why, why not?

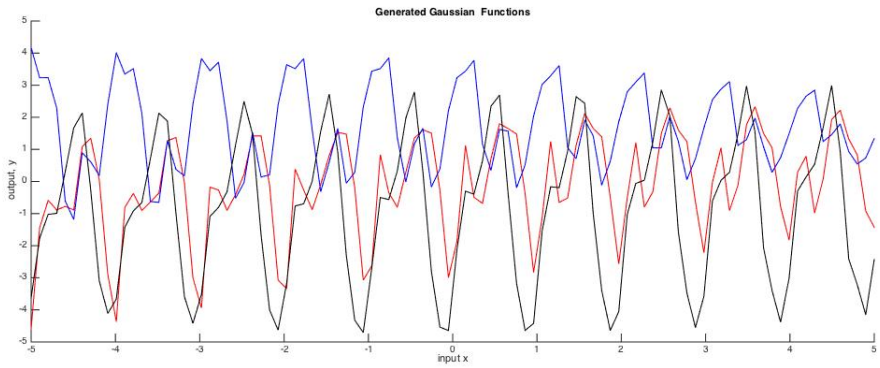
First of all, I would like to note that depending on where we set the initial parameters we can converge to different solutions when optimising for the hyper-parameters, some of which are wiggly some of which have no wiggle-room. For instance, a very wiggly model is shown to the left, and a less wiggly one on the right. To find a good curve, I sampled from the space of parameters and found the one with highest likelihood (after minimisation).

Now, if we look at the error bars for the model on the left, they don't seem to change in width. For a periodic function, we might expect the error to be greater at the crest with amplitude is highest. This is indeed the case for the function on the right, which makes us believe it is a more plausible model. In our model (a) we see that there are variable error bar widths and these are greater at the peaks of the curve (just like the model on the right, which I think is a better model).

I think the data is periodic because it is possible to identify successive peaks and troughs in the data set. These seem to be spaced at a regular interval. However, there may also be aperiodic components, for instance it looks as though there is a slight downward trend. Also, the peaks are clearly not of a constant amplitudes. The log marginal likelihood of the better model in the right is 35, which is greater than the log likelihood of the models in (a). This supports the hypothesis that the mechanism is periodic.

Question D)

Below is a plot of the sampled gaussian functions for different points sampled from a random distribution.



Why add the off diagonal? Explain their behaviour.

We added the small off diagonal for numerical stability. For ill-conditioned matrices, the cholesky decomposition can be unstable. Adding the off-diagonal ensures that the matrix is well-conditioned.

To generate other gaussian functions I created a random unit ball of points in 100-D space, then stretched this ball according to the cholesky decomposition of the covariance matrix. I then plotted the outputs against the function's 100 input points.

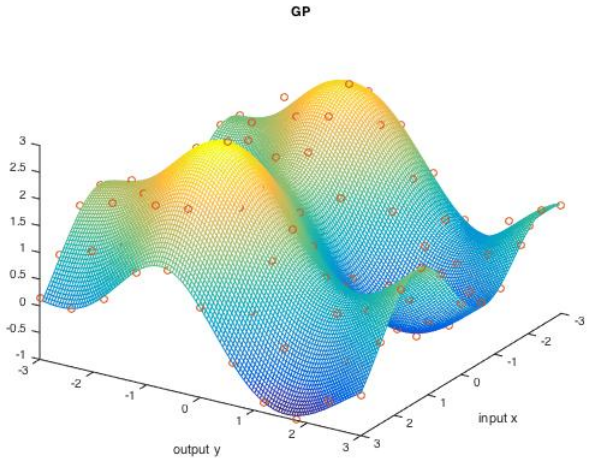
The covariance function is a product of a periodic covariance kernel and a squared exponential covariance kernel. This means we should expect our points to broadly speaking resemble a periodic function, without being exactly periodic.

If we chart successive peaks, we can actually see that they are often rising or falling from left to right. The squared exponential kernel is responsible for this general trend whereas the covariance kernel is responsible for the successive peaks in the data. We initialised the period parameter to 1 (and indeed we see roughly a period of 1 in the diagram). The length scale of the squared exponential was set to e^2, which is quite long. This would explain the slow moving general trend of the diagram.

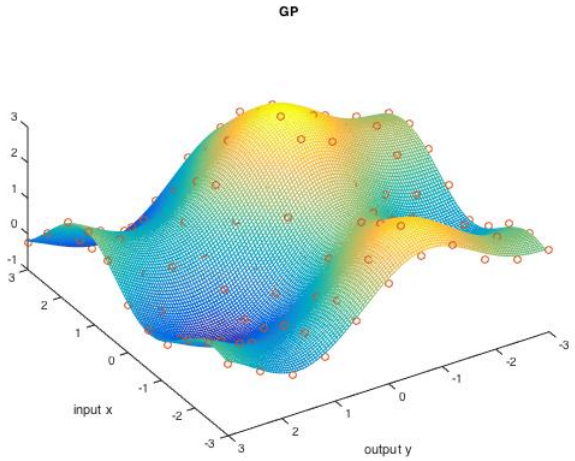
The difference between the curves can be explained through random variations in the data we generated.

Question E)

(NB: typo in labeling, "output y" should read "input y".)



(1)



(2)

Comment on the fit. How much noise is there in the data?

Above I show two different orientations of the data points, fitted with gaussian functions. I think the gaussian functions make a good fit for the data. We can see from the proximity of the curve to the data-points, pictured in red, that there is little noise.

The model predicts a noise standard deviation of 0.1026. This is small. Another way to measure the noise would be to measure the sum of squared difference between the data points and the function. I have done this and found that the sum of squared errors is 0.8346 (small).

Question F)

Is this a better model, why, why not? What is the relative probability of the two models, e) and f)?

The difference between the two models is that the second has an isotropic length scale whereas the first has different length scales for different inputs. We would expect the non-isotropic model in E to do better if the inputs are more naturally represented along different length scales. To my eye, it looks as though the data varies more along the y axes than along the x axes. So I would say the current model is a worse model.

For the non-isotropic model, I obtained a log marginal likelihood of 19.2187. For the isotropic, I obtained a log marginal likelihood of 18.0691. The relative probability is given by $\frac{\text{likelihood1}}{\text{likelihood2}} = \exp\{\text{loglikelihood1-loglikelihood2}\} = \exp\{19.2187 - 18.0691\} = 3.1569$. The first model is around 3 times as likely given the data.

Question G)

Why is symmetry breaking necessary? Fit. Explain the model. Is the predicted function significantly different from the one obtained in e)? Is the probability of the model very different? Explain how to reconcile your last two answers.

We have to break symmetry with the hyper-parameters because otherwise when we perform gradient ascent to optimise the hyper parameters it will move the hyper-parameters from the first kernel in exactly the same direction as the hyper-parameters for the second kernel. Thus we would do no better than by just having one kernel. The model's parameters include length scales in [x,y] of [1.4268, 886.9967] for the first kernel and [1.0722, 689.9363] for the second. Now if we compare this to our result from (E), we obtained [1.5116, 1.2859] as the length scales.

On the surface it seems as we have similar kernels since each kernel in our model from G nullifies a single feature by giving it a huge length scale . What this amounts to is that one kernel effectively only measures the covariance from points in x (since the y length scale is too large) and the other from points in y for the same reason. When we look at the parameters, we in fact find that they are very close to the length scales for the kernel from E. The sd of the noise is also very similar. Howeveer their form is different. Because of the huge length scales a simplification of their form would be this:

$c * exp(-x1^2 - x2^2)$ for the first model and $c * exp(-x1^2) + exp(-x2^2)$ for the second.

The likelihood of the new model is much lower at -66.3859. In contrast the model from E had a likelihood of 19.218. I believe this is because kernel function from the first model measures closeness in 2d space before exponentiating, whereas the second kernel only measures closeness in each dimension before exponentiating. We would think of a points (0,0) and (1000,1) as being far away, but the second kernel of the second model does not because the points are close in one feature! So it is a worse kernel.

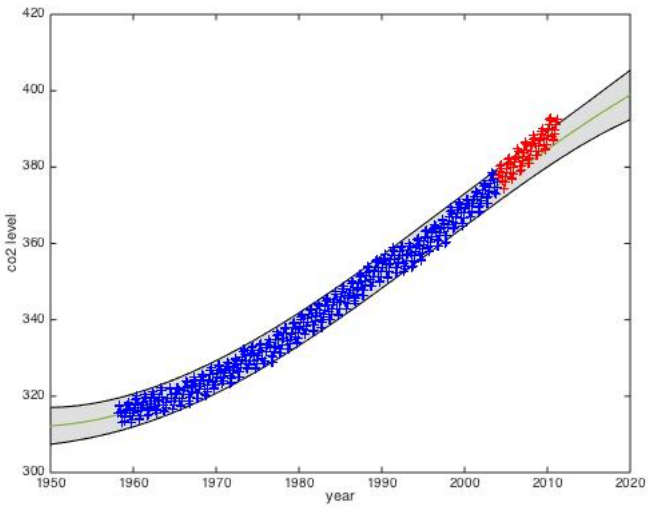
Question H)

Show and comment on the fit and the hypers, and the predictions for the test data

The fit for the training set seems to be quite good. It captures the long term trend well by choosing a very large length scale and smoothing out the data. However, it seems quite bad at modeling the yearly trends in co2 emissions, where there is quite a strong periodic component. These variations could be modeled by a second periodic kernel function on a much smaller length scale. Our model misses this out.

The predictions for the test data are pretty off. We can sense from the graph that co2 emissions are increasing at an accelerating rate, but the curve misses this and is more content to plot what seems more like a straight line up to 2010. The curve then tails off at around 2020, perhaps due to the large length scales and being unduly influenced by points far away (in the 1960s). This again seems wrong.

The hyper-parameters are: [l =105.3774, signal σ=372.4518, noise σ = 2.1385].



(1)

Question I)

Explain, and show your fit.

Ideally, I would like to have kernel functions that capture all of these facets of the data:

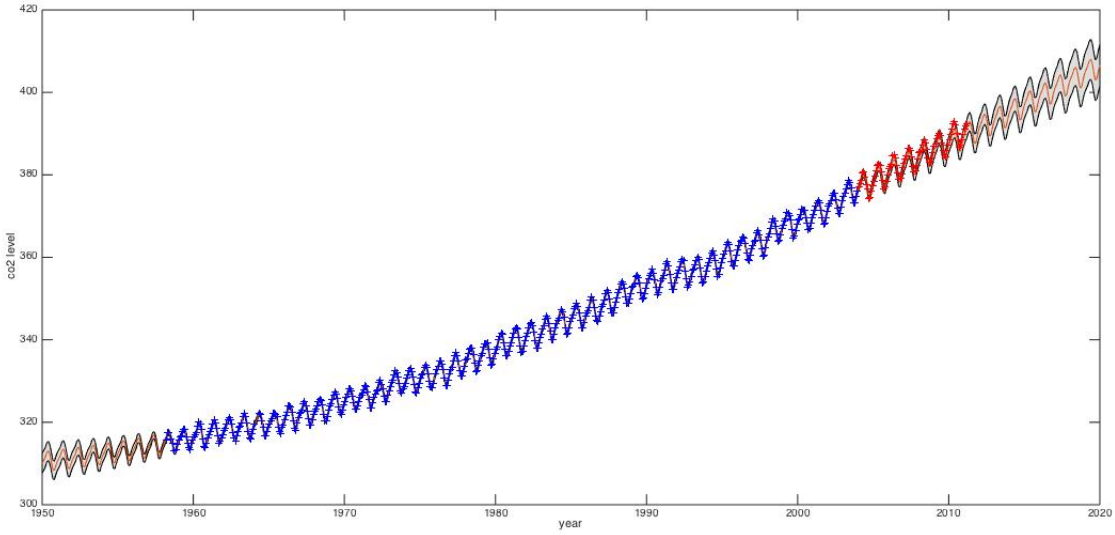
- The long term trend for the CO2 emissions; the annual periodic component; variations across different length scales; short term, month to month variations in CO2 levels, which tend to be highly correlated. To this end, I will want 4 Kernels as described below:

The covariance functions are defined as follows:

```
k1 = @covSEiso; % covariance contributions, long term trend
k2 = {@covProd, {@covPeriodic, @covSEisoU}}; % close to periodic component similar
k3 = @covRQiso; % fluctuations with different length-scales
k4 = @covSEiso; % very short term (month to month) correlations
covfunc = {@covSum, {k1, k2, k3, k4}};
```

- Having described the purpose of the different kernels, I can explain how I went about choosing the different parameters. Kernel k1 was intended to capture long term fluctuations in the data, so I initialised the length-scale to e^3 (roughly 20 years) and signal variance to e^4=50 since there are large scale fluctuations in co2 over a long length scale. For the second kernel, we want the period to be a year = e^0, and the length scale to be about 3 =e^1. The yearly fluctuations and monthly fluctuations are small so I set the signal variance to 1=e^0. For the 4th kernel, we wanted to capture short term fluctuations, i.e. month to month changes in the data, so we set the length scale to $e^{-2} = 0.13$ which is approximately one month. Also monthly fluctuations tend to be small so we gave a similar signal sd. For the rational quadrature kernel I tried different scales for alpha by trial and error and found 0.1 to be a good choice. In the end my initialisation parameters looked like this: hyp.cov = [3 4 0 0 0 4 0 0 -1 -2 -2];

I obtained a log marginal likelihood of -47.738 on the test data which was better than the -200 that obtained for the function in F. Below is a plot.



(1)similar

Code Snippets

```
%code Qa Main command
hyp.lik = 0; hyp.cov = [0  -0];
hyp = minimize(hyp, @gp, -100, @infExact, [], @covSEiso, @likGauss, x, y);
```

```
%code Qb Find the likelihood of the data for a range of initial hyper-parameters and store in Z
y = datasample(1:10,100)
Z=zeros(11,11);
for n=0:10
        for m=0:10
                N= -1+2*n/10 ; M=-1+2*m/10; hyp.cov=[N,0]; hyp.lik=M;
                hyp = minimize(hyp, @gp, -100, @infExact, [], covfunc, likfunc, x, y)
                Z(n+1,m+1) = gp(hyp, @infExact, [], covfunc, likfunc, x, y);
        end
end
```

```
%code Qc Find a well fitting periodic function to model the data
 covfunc=@covPeriodic;    likfunc = @likGauss;
% I used code similar to Qb to find very good local minima for the  params and then plotted
```

%Question D

```
% Code to draw a single Gaussian function.
% Parameter settings
hyp.lik = 0; hyp.cov = [-0.5 0 0 2 0]; covfunc={@covProd, {@covPeriodic, @covSEiso}};
small_Diagonal= 1e-6*eye(n); % diagnal unit for stability
 % number of data points
n=100;

% generate the covariance of the gaussian process  (mean=0)
% Think of the 100 points of f(x) as being 100 dimensions.
    x = linspace(-5,5,n)';
    xrand = gpml_randn(2, n, 1); % generate 100 dimensional unit gaussian noise
    K = feval(covfunc{:}, hyp.cov, x)+small_Diagonal;  % add diagonal unit to covariance for stability
    y = chol(K)'*xrand;  hold on;  %give the unit gaussian the correct covariance
    plot(x,y)  % plot the points
```

%Question E

```
% Code to create a mesh grid of points in interval [-3..3]^2 and vector of
% this grid
len=(-3:6/(n-1):3);
wid=(-3:6/(n-1):3);
[Z1,Z2]=meshgrid(len,wid);
vectored=[reshape(Z1,[],1), reshape(Z2,[],1)]

%Get the output values for this vector (according to the function) and plot
[m s2] = gp(hyp, @infExact, [], covfunc, likfunc, x, y, vectored);
mesh(Z1,Z2,reshape(m,n,n))
```

```
%code QF Initialisation parameters for the isotropic kernel
 likfunc = @likGauss; covfunc = @covSEiso; hyp.cov = [-1 0]; hyp.lik = 0;
% Otherwise Code runs exactly the same as E

%code QG Initialisation parameters for the isotropic kernels
 likfunc = @likGauss;   covfunc = {@covSum, {@covSEard, @covSEard}};   hyp.cov = 0.1*randn(6,1);
hyp.lik = 0;
% Otherwise Code runs exactly the same as E

%code QH Initialisation parameters for the isotropic kernels
likfunc = @likGauss; covfunc = @covSEard; hyp.cov = [-1,0]; hyp.lik = 0;
% Otherwise Code runs similarly to A
```