CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4F13	Title of report	Coursework 1: Gaussian Processes

Date submitted: 04/11/2022	Assessment for this module is [✓ 100%] / ☐ 25% coursework of which this assignment forms 33 %

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5554D	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)		A*	A	B	C	D
Indicative grade guideline		>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:		*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

4F13 Probabilistic Machine Learning
Coursework 1: Gaussian Processes

CCN: 5554D
Words: 996

November 4, 2022

1 Task A

A Gaussian Process (GP) with squared exponential (SE) covariance function with isotropic distance measure, in (1), was used to model the data in `cw1a.mat`, which consisted of 1D inputs. This kernel is stationary and isotropic, as it is a function only of $|x_1 - x_2|$, and therefore invariant to all rigid motions in the input space [1].

$$k(x_1, x_2) = \sigma_s^2 \exp\left(\frac{-(x_1 - x_2)^2}{2l^2}\right) \tag{1}$$

σ_s^2 - signal variance, l - length scale

The GP model, with zero mean, assumed a Gaussian likelihood function, $P(y|\mu) = \mathcal{N}(y|\mu, \sigma_n^2)$, where σ_n^2 is the noise variance. The model was initialised as in Table 1 and optimised by minimising the negative log marginal likelihood (NLML).

Hyperparameter	log(l)	l	log(σ_s)	σ_s	log(σ_n)	σ_n
Initial	-1	0.3679	0	1	0	1
Optimised	-2.0540	0.1282	-0.1087	0.8970	-2.1385	0.1178
Negative Log Marginal Likelihood	11.899					

Table 1: Initial and final hyperparameters of GP model, where all logs are natural.

The GP's predictions are given in Figure 1. The model is confident in areas with many training data-points, shown by the narrow error bars (± 2 standard deviations from mean). Beyond -3.2 and $+2.8$, the predictive mean is flat at 0, and the error margin is very wide. Optimisation decreases the length-scale from 0.37 to 0.13 - this value allows the model to closely fit the variations in the output, y. The noise variance reduces to 0.12, which results in tight error bounds in areas with high densities of data, whilst the signal variance remains fairly large at 0.90, which results in the 95% bounds being large outside of the training range. This model is a good fit of the data, and does not overfit in outside of the training range.

Listing 1: Code to optimise hyperparameter by minimising the negative log marginal likelihood.

```
cov = [-1, 0]; % initial covariance: 1) log length-scale, 2) log signal std-dev
lik = 0; % initial likelihood - log noise st dev
hyp = struct('mean', [], 'cov', cov, 'lik', lik); % hyperparameter struct
[nlZ, ~] = gp(hyp, @infGaussLik, mean_func, cov_func, lik_func, x, y); % initial NLML
% optimised hyperparams by minimising negative log likelihood
hyp_opt = minimize(hyp, @gp, -100, @infGaussLik, mean_func, cov_func, lik_func, x, y);
[nlZ2, ~] = gp(hyp_opt, @infGaussLik, mean_func, cov_func, lik_func, x, y); % optimised NLML
```

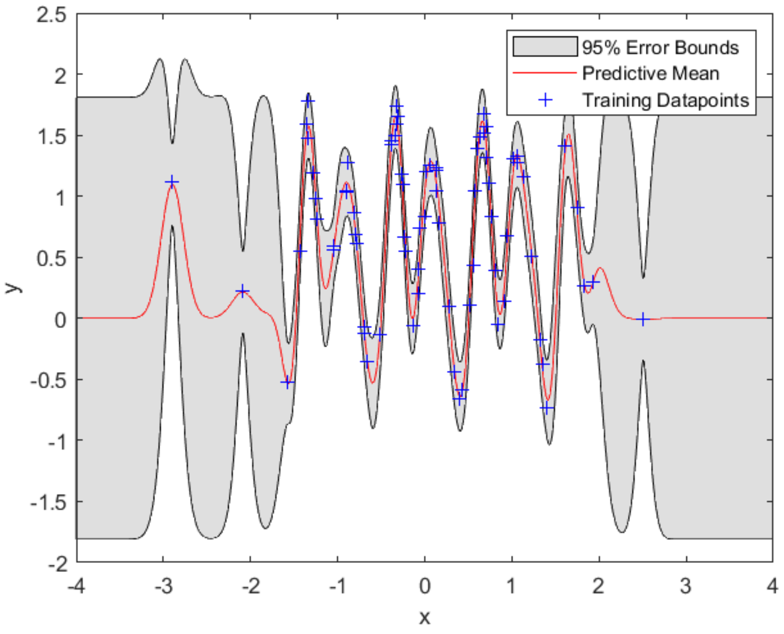

Figure 1: Model predictions for input range $[-4, 4]$, with 95% error bounds, and the training data overlaid.

2 Task B

When the hyperparameters are initialised differently, a different local optimum can be found, which is sub-optimal compared to the previous, as shown in Figure 2. The hyperparamters of this GP are given in Table 2. The length-scale, l, is 10.5, which is much larger than that of the GP in (a); this prevents the tracking of sharp changes in y with respect to x. The optimised signal and noise variances are fairly large at 0.73 and 0.66, respectively, which results in large error bounds for the entire input space. This GP is clearly underfitting, which is further backed up by a large NLML of 78, compared to 12 previously.

Hyperparameter	log(l)	l	$\log(\sigma_s)$	σ_s	$\log(\sigma_n)$	σ_n
Initial	-1	0.3679	-10	4.540e-5	0	1
Optimised	2.3477	10.4615	-0.3126	0.7315	-0.4107	0.6632
Negative Log Marginal Likelihood	78.2269					

Table 2: Initial and final hyperparameters of sub-optimal GP model, where all logs are natural.

To further investigate this, hyperparameter searches were conducted. Figure 3 shows contour plots of the NLML (before optimisation) as a function of the initial hyperparameters. In each plot, two hyperparameters were varied, whilst the third was held at 0. The contour plots show that the initial setting of the hyperparameters in task (a) (i.e. $[\log(l), \log(\sigma_s), \log(\sigma_n)] = [-1, 0, 0]$) had a low NLML, and therefore was a good choice. The GP model from task (a) was superior - it fit the data well, without overfitting outside of the training range, and had equal model complexity.

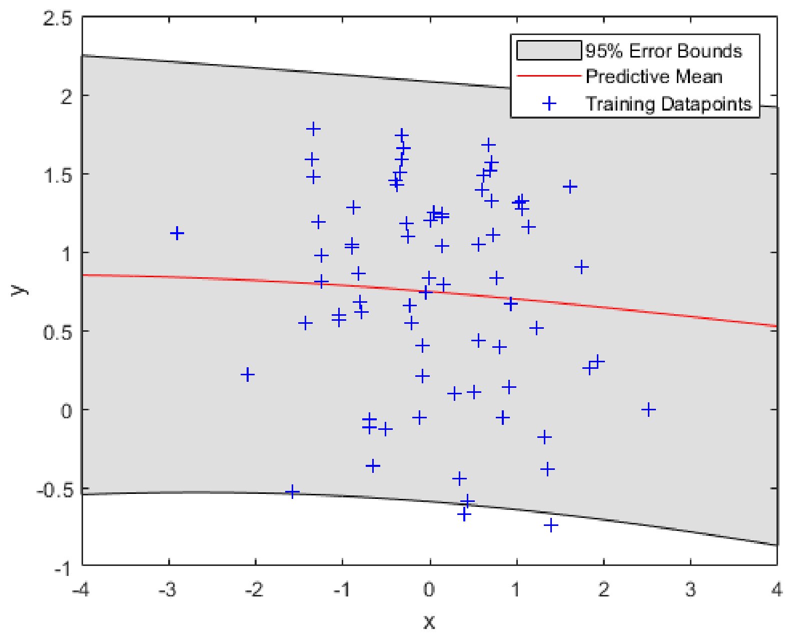

Figure 2: Predictions for input range $[-4, 4]$ by differently initialised model, with 95% error bounds, and the training data overlaid.

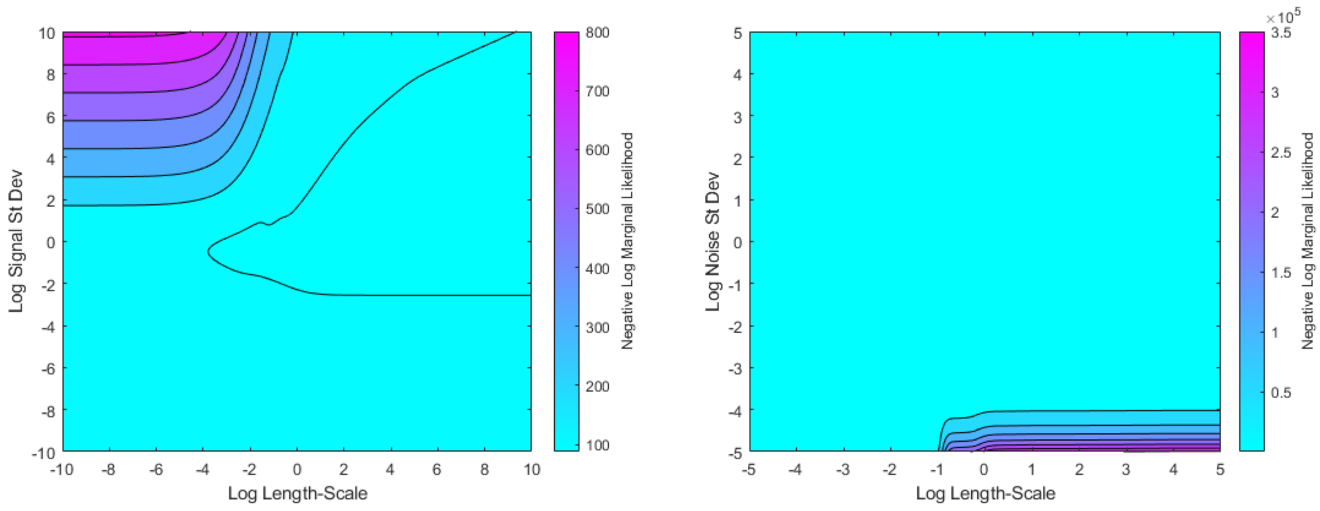

Figure 3: Hyperparameter search

Listing 2: Code to conduct hyperparameter search and plot 2D contour map.

```
N = 20; % number of hyperparams to test
cov_vals = linspace(-5, 5, N); % range of values for covariance hyperparams
grid = zeros(N, N);
for i = 1:N
    for j = 1:N
        hyp = struct('mean', [], 'cov', [cov_vals(i), 0], 'lik', cov_vals(j)); % hyperparameter struct
        % obtain negative log margianl likelihood (nlZ)
        [nlZ, ~] = gp(hyp, @infGaussLik, mean_func, cov_func, lik_func, x, y);
        grid(j,i) = nlZ;
    end
end
contourf(cov_vals, cov_vals, grid)
```

3 Task C

A GP with a periodic covariance function, given by (2), was used to model the same data. Various initial settings of the hyperparameters were tested, with the predictions of six shown in Figure 4. The optimised parameters and NLMLs of each are provided in Table 3. The lowest NLML is -35, which corresponds to the top centre plot in Figure 4; visually, this plot looks to fit the data the best out of the six. A negative NLML corresponds to a marginal likelihood greater than one - much greater than that of the GP with SE covariance. However, on further inspection of the plot, we can see that the model is overly confident in areas of no training data. The predicted periodic pattern spans the entirety of the input space, and the error bounds do not widen outside of the training range.

The (non-logarithmic) optimised hyperparameters of model (b) are $[l, p, \sigma_s, \sigma_n] = [1.045, 0.999, 1.237, 0.109]$.

$$k(x_1, x_2) = \sigma_s^2 \exp\left(\frac{-2\sin^2(\pi(x_1 - x_2)/p)}{l^2}\right) \tag{2}$$

σ_s^2 - signal variance, l - length scale, p - period

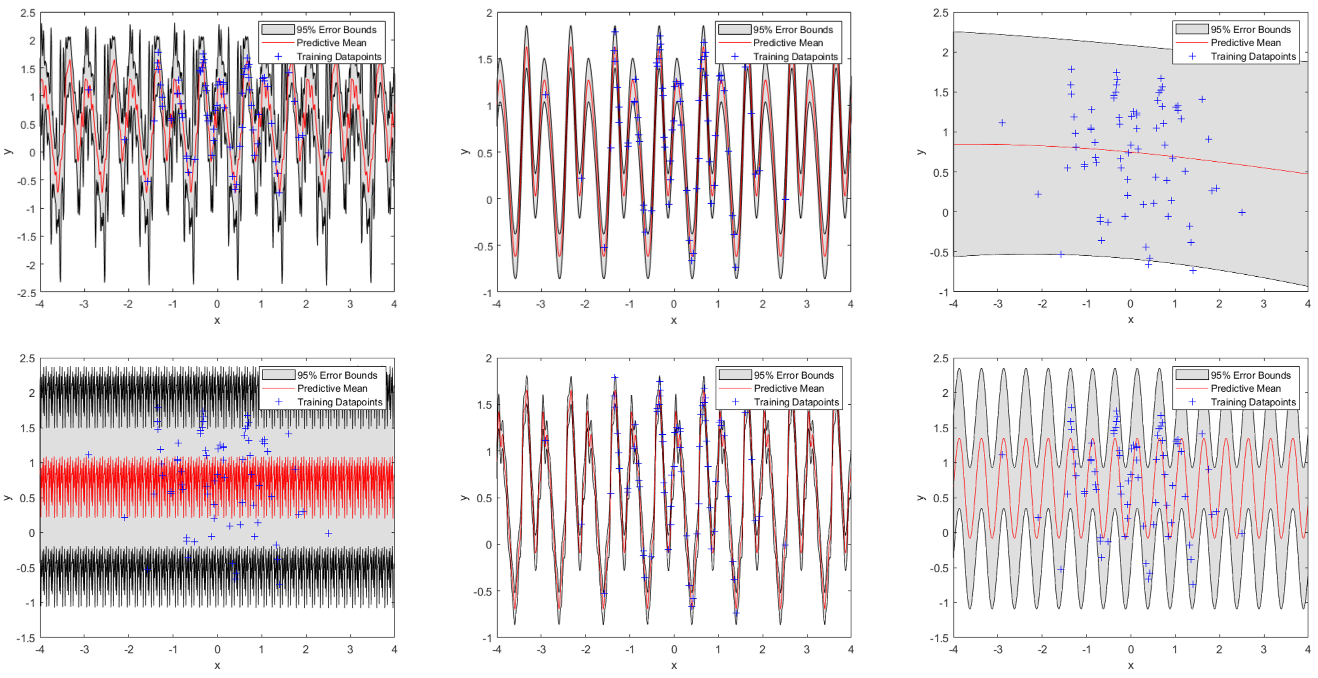

Figure 4: Predictions for input range $[-4, 4]$ for GPs with periodic covariance functions, for 6 different settings of the hyperparameters. From top left to bottom right, $[\log(l), \log(p), \log(\sigma_s), \log(\sigma_n)]$ = (a) $[5, 0, 0, 0]$, (b) $[0, 0, 0, 0]$, (c) $[0, 5, 0, 0]$, (d) $[0, -5, 0, 0]$, (e) $[0, 0, 0, -9.4]$, (f) $[0, 0, -10.4, 0]$.

Model	(a)	(b)	(c)	(d)	(e)	(f)
Optimised $\log(l)$	-2.0260	0.0437	-1.6001	0.0398	-1.1366	1.3457
Optimised $\log(p)$	-0.0011	-0.0012	5.5836	-5.0019	-0.0012	-0.6905
Optimised $\log(\sigma_s)$	0.4166	0.2124	-0.3441	-0.0446	-0.1378	0.0967
Optimised $\log(\sigma_n)$	-1.7115	-2.2124	-0.4110	-0.4897	-2.4649	-0.7072
Negative Log Marginal Likelihood	33.2047	-35.2508	78.2207	77.9421	-19.9287	44.8051

Table 3: Model (log) parameters and negative log marginal likelihoods of six GPs with periodic covariance functions, but different initial settings of the hyperparameters.

To investigate whether the data-generating mechanism was strictly periodic, the residuals between model (b)'s predictions and the training data were found and plotted as a histogram, shown in Figure 5. This suggests that the noise is zero-mean Gaussian, which would suggest that the data is in fact periodic.

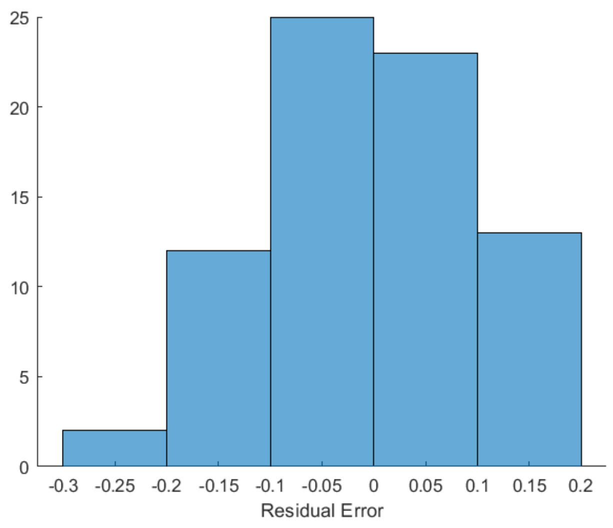

Figure 5: Histogram of residual errors between periodic predictive mean and the ground truth data.

Listing 3: Code to plot residual errors as a histogram

```
% make predictions on training input data, x
[mu, s2] = gp(hyp_opt, @infGaussLik, mean_func, cov_func, lik_func, x, y, x);
% plot histogram
histogram(y-mu)
```

4 Task D

Random noise-free functions were generated from a GP with a covariance function equal to the product of a periodic and SE kernel, with the hyperparameters set to $[\log(l_p), \log(p), \log(\sigma_{s,p}), \log(l_{se}), \log(\sigma_{s,se}), \log(\sigma_n)] = [-0.5, 0, 0, 2, 0, 0]$. The data is generated by making use of Cholesky Decomposition, which requires the matrix to be positive definite. To satisfy this requirement, a small diagonal matrix is added to the covariance matrix - without it, finite precision arithmetic may result in negative eigenvalues.

Figure 6 shows four randomly generated functions from the GP, as well as the separate components from the two covariance functions. The randomly generated functions have periodic components, as well as gradual changes (e.g. growth and sloping).

On further inspection of the data from `cw1a.mat` (in Figures 1 and 2), there is a clear periodic component but also a slight downwards slope from left to right. This suggest that the data may have been sampled from one of these random functions, generated by a GP with a composite product covariance i.e. the data-generating mechanism is not strictly periodic.

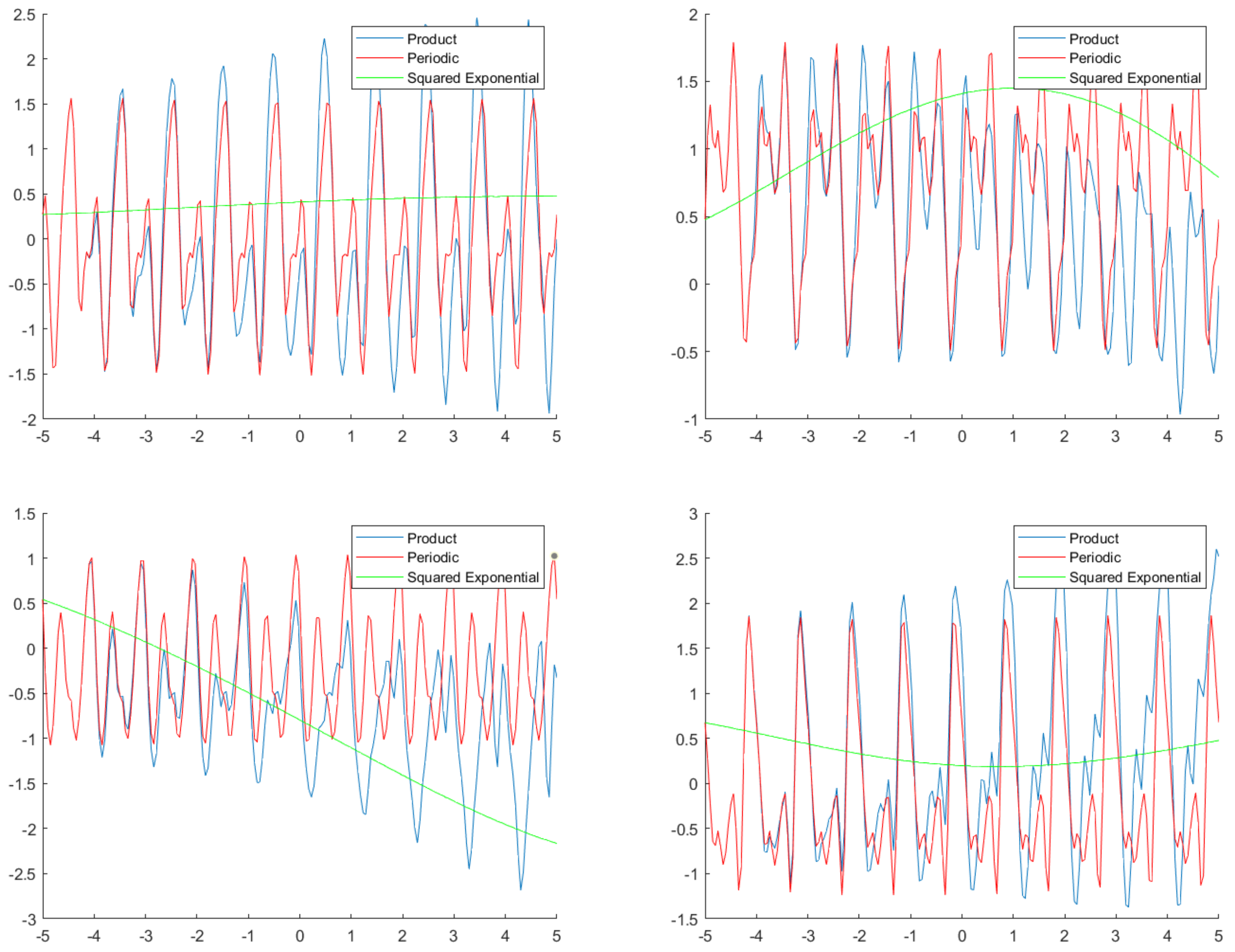

Figure 6: Four randomly generated functions from GPs with product covariance (periodic x SE), periodic covariance, and squared exponential covariance.

Listing 4: Code to generate random noise-free functions from a GP, and plot the functions.

```
mean_func = []; % empty - don't use mean function
cov_func = {@covProd, {@covPeriodic, @covSEiso}}; % squared exponential covariance function
lik_func = @likGauss; % gaussian likelihood func

%initial hyperparams
cov = [-0.5, 0, 0, 2, 0]; % initial covariance params
lik = 0; % initial likelihood - log noise st dev

x = randn(200, 1);
xs = linspace(-5, 5, 200)';
K = feval(cov_func{:}, cov, xs) + 1e-6*eye(200);
K_per = feval(@covPeriodic, [-0.5, 0, 0], xs) + 1e-6*eye(200);
K_iso = feval(@covSEiso, [2, 0], xs) + 1e-6*eye(200);

y = chol(K)' * x;
y_per = chol(K_per)' * x;
y_iso = chol(K_iso)' * x;

hold on;
plot(xs, y, 'DisplayName', "Product", 'LineWidth', 1)
plot(xs, y_per, 'r', 'DisplayName', "Periodic", 'LineWidth', 1)
plot(xs, y_iso, 'g', 'DisplayName', "Squared Exponential", 'LineWidth', 1)
Lgnd = legend('show');
```

5 Task E

A GP with a SE covariance function with Automatic Relevance Determination (ARD) distance measure was used to model 2D-input data in `cw1e.mat` (shown in Figure 7), as well as a GP with the sum of two SE-ARD covariance kernels. For 2D data, this function has two length-scale parameters.

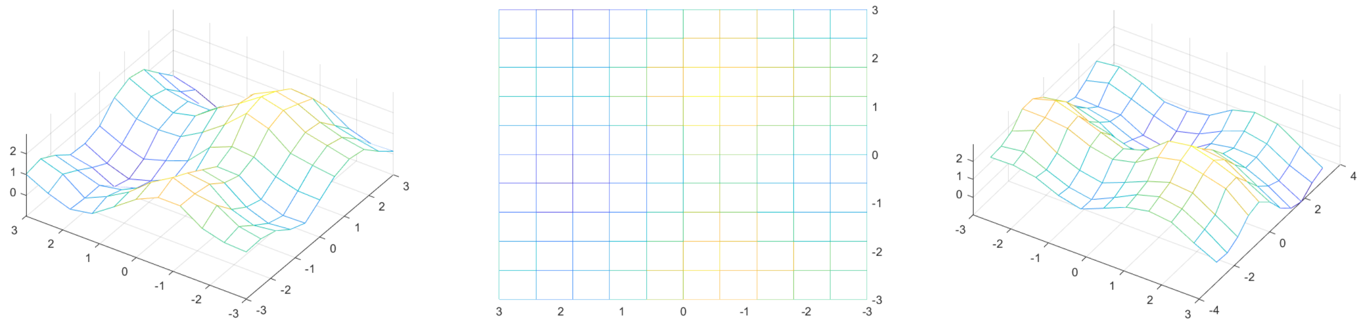

Figure 7: Visualisation of 2D input data.

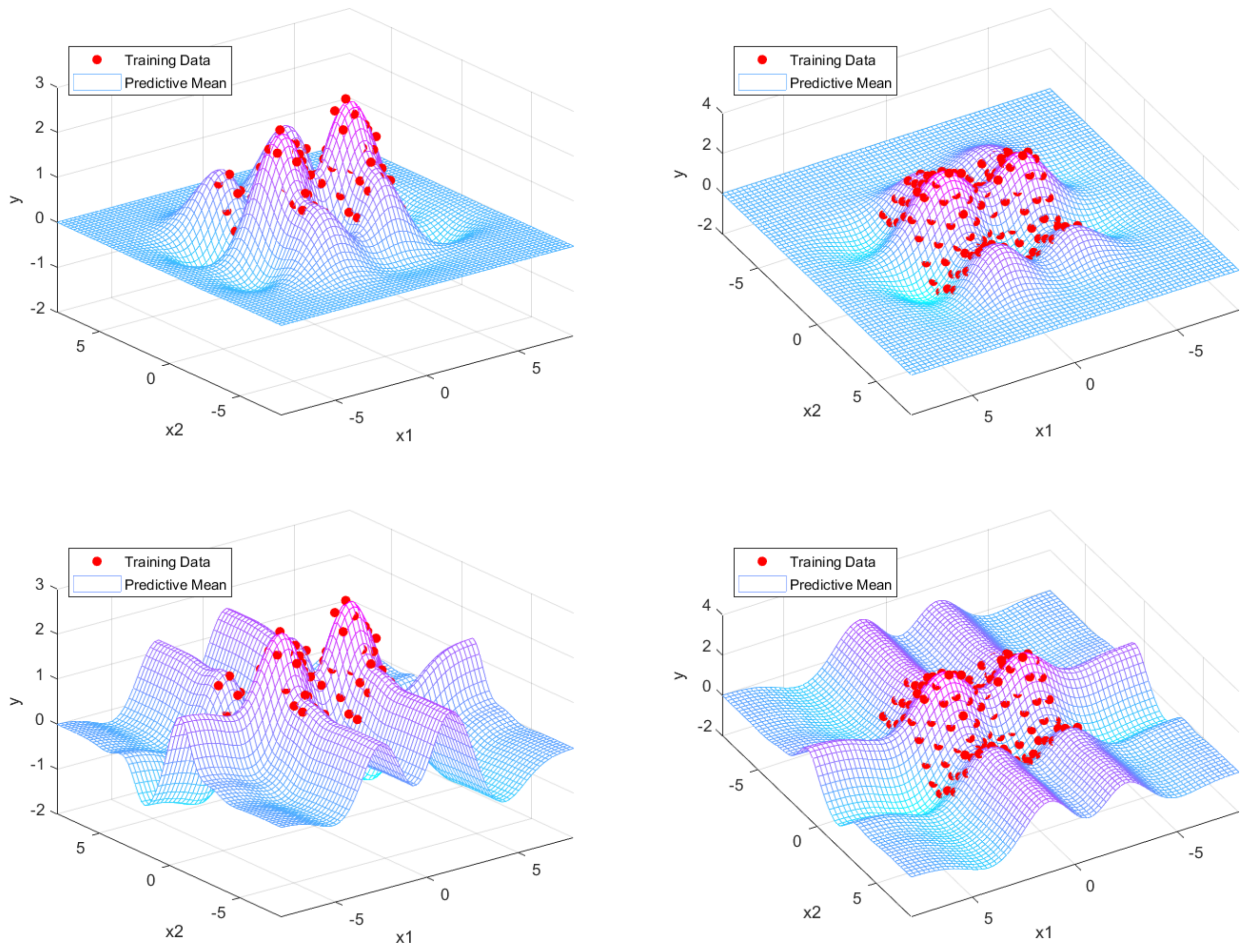

Figure 8: Predictive plots for the single SE-ARD covariance GP (top row) and the additive GP (bottom row), with the training data overlaid.

Figure 8 shows the predictive plots for the two models. The first model predicts only zero outside of the training data. The initial hyperparameters of both were set randomly (i.e. `cov = 0.1*randn(6,1)`) - the optimised hyperparameters and NLMLs of the two models are given in Table 4. The NLML of the summed-covariance model is much lower than the other -

the data was more likely to have been generated by the second model. To further evaluate the two, the data was randomly split into train and test sets (80-20), and the RMSEs were calculated for the model predictions - the double SE-ARD model performed better (0.12 vs 0.15).

Model	l_1	l_2	$\sigma_{s,1}$	l_3	l_4	$\sigma_{s,2}$	σ_n	**NLML**	**RMSE**
Single SE-ARD	1.551	1.460	1.191	-	-	-	0.099	-10.291	0.150
Double SE-ARD	1.435	1864	1.069	1045	1.015	0.713	0.092	-52.368	0.123

Table 4: Optimised hyperparameters, negative log marginal likelihood and root mean squared error (RMSE) on the test set for each model.

To visualise the 95% error bars, slices were taken, with one input held constant, whilst the other was varied, as shown in Figure 9. Neither model overfits outside of the training data range, shown by the large error bars, but the second model predicts non-zero values outside the training range, and is more confident within the training range. The additive model is a better fit of the data, but is more complex, with double the number of covariance hyperparameters.

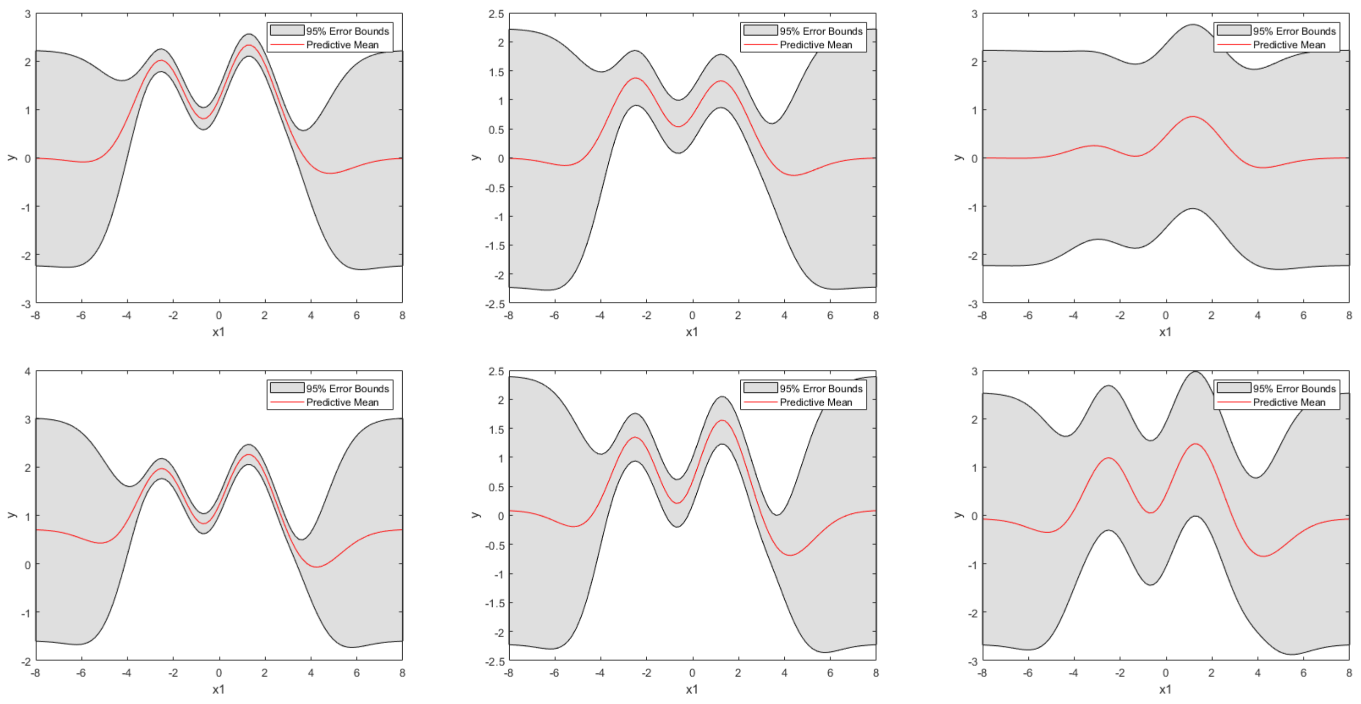

Figure 9: Slices of the input space, showing model predictive means and 95% confidence bands. Top row: single SE-ARD covariance GP. Bottom row: sum of two SE-ARD covariance GP. First column: $x_2 = 0$. Second column: $x_2 = -3.5$. Third column: $x_2 = +5$.

Listing 5: Code to optimise GP with two SE-ARD covaiance functions summed, and plot 3D precitive plot.

```
%initial hyperparams
cov = 0.1*randn(6,1); % [log(ell_1), log(ell_2), log(sf), log(ell_1), log(ell_2), log(sf)
lik = 0; % initial likelihood - log noise st dev
hyp = struct('mean', [], 'cov', cov, 'lik', lik); % hyperparameter struct

% optimised hyperparams by minimising negative log likelihood
hyp_opt = minimize(hyp, @gp, -100, @infGaussLik, mean_func, cov_func, lik_func, x, y);

a = 8; % Range for input test data
N = 70; % Number of input test data samples
[xs1, xs2] = meshgrid(linspace(-a, a, N)', linspace(-a, a, N)');
xs = [xs1(:), xs2(:)];
[mu, s2] = gp(hyp_opt, @infGaussLik, mean_func, cov_func, lik_func, x, y, xs);

% Produce 3D plot of predictive mean for test input, with training data overlayed.
```

```
plot1 = scatter3(x(:,1), x(:,2), y, 'r', 'filled', 'DisplayName', "Training Data");
hold on;
plot2 = mesh(xs1, xs2, reshape(mu, N, N), 'DisplayName', "Predictive Mean");
colormap(cool);
```

Listing 6: Code to split data into train and test sets.

```
% Load data
data = load('cw1e.mat');
x = data.x;
y = data.y;
xy = [x, y];

% Random train-test splits
rng(1)
a = 0.8; % Fraction of data for training
M = int16(a*length(x));
random_xy = xy(randperm(size(xy, 1)), :); % Randomly switch rows
xy_train = random_xy(1:M,:); % Select train set
save("xy_train.mat", "xy_train") % Save train set to file

xy_test = random_xy(M+1:length(x), :); % Select test set
save("xy_test.mat", "xy_test") % Save test set to file
```

Listing 7: Code to calculate root mean squared error on test set predictions.

```
[mu_1, s2_1] = gp(hyp1_opt, @infGaussLik, mean_func, cov_func1, lik_func, x_train, y_train, x_test);
[mu_2, s2_2] = gp(hyp2_opt, @infGaussLik, mean_func, cov_func2, lik_func, x_train, y_train, x_test);

error1 = mu_1 - y_test;
error1 = error1.^2;
error2 = mu_2 - y_test;
error2 = error2.^2;

disp(sqrt(sum(error1)/length(y_test)))
disp(sqrt(sum(error2)/length(y_test)))
```

References

[1] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.