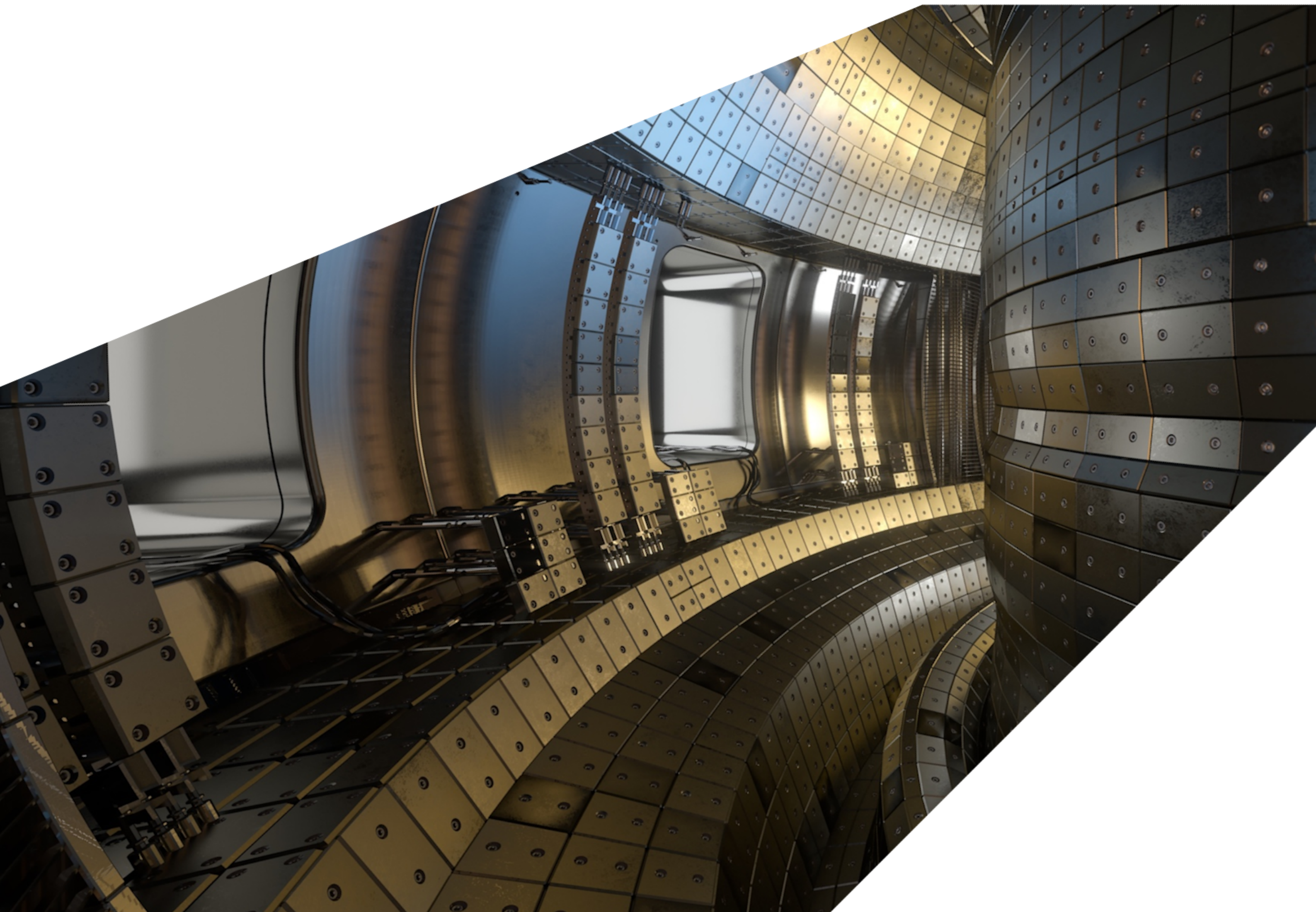
UK Atomic Energy Authority

# ExCALIBUR

Selection of Techniques for Uncertainty Quantification

M5.2 Version 1.00

**Abstract**
The report describes work for ExCALIBUR project NEPTUNE at Milestone M5.2. Since contractual issues with grantees prevented a timely, meaningful exercise in code integration, acceptance and operation, the work consists of a discussion of alternative approaches to the production of surrogates. Neural networks are briefly considered, but the main emphasis is on a comparison between adaptive spline fitting and Gaussian processes. Expected scaling costs with number of datapoints $N_d$ are derived mathematically for both approaches, and verified by use of opensource packages PY-EARTH (spline fitting) and SKLEARN (Gaussian processes).



**UKAEA REFERENCE AND APPROVAL SHEET**

| | | |
|---|---|---|
| | Client Reference: | |
| | UKAEA Reference: | CD/EXCALIBUR-FMS/0063 |
| | Issue: | 1.00 |
| | Date: | 18 March 2022 |

Project Name: ExCALIBUR Fusion Modelling System

| | Name and Department | Signature | Date |
|---|---|---|---|
| Prepared By: | Wayne Arter<br>Joseph Parker<br><br>BD | N/A<br>N/A | 18 March 2022<br>18 March 2022 |
| Reviewed By: | Wayne Arter<br><br>Project Technical Lead | W. Arter | 18 March 2022 |

## 1 Introduction

The main aim of this activity is to ensure that the UQ grantees (successful bidders to ITT T/AW085/21) perform their work such as to bring maximum benefit to ExCALIBUR project NEPTUNE. Unfortunately, despite the previous contractual agreement between UKAEA and UCL around the earlier UQ and MOR ExCALIBUR grants, and although it was decided that UCL were the successful bidders in early August, contractual negotiations were prolonged. Thanks to the delay, at the time of writing (Mid-March 2022), grantee tasks have not been completed to the extent that many code integration, acceptance and operation tasks have been possible.

It has been previously noted [1] that the grantees have tended to focus on specific aspects of UQ, which although relevant to NEPTUNE, may not answer all the needs of the project. Moreover, little has been performed in the way of comparison of the proposed Gaussian Process (GP) approach with older methods of producing surrogates, such as spline interpolation and neural networks (NNs). Although interest in the latter approach has revived considerably of late, other UKAEA and cross-cutting ExCALIBUR projects are currently examining it. If NNs can be shown reliably to solve the basis-selection problem posed in [1, § 5], ie. that they prove capable of say, without supervision, selecting eigenfunction-type bases for adjoint operator equations and pseudospectral bases [2] for non-normal operators, then NNs might take centre-stage as surrogates for NEPTUNE. However, to be fit for UQ, currently major questions need to be answered relating to the stability of the NN approach in extrapolation, and to the determination of their errors. Herein, discussion of NNs is relegated to a brief comparison with GPs in an Annex Section A.

Thus this report mainly consists of a comparison between spline-based and GP approaches. Note that as foreshadowed in ref [1, § 1], the two approaches may anyway be complementary, in that splines are capable of fitting large data-sets with sizes $N_d$ of Petabytes and larger, whereas GPs can only fit relatively small data-sets. As preliminaries, the next Section 2 establishes mathematically why GPs are so costly, scaling as $N_d^3$, and Section 3 provides an introduction to the mathematics of splines, in particular of Multivariate adaptive regression splines (MARS). Thereafter, Section 4 contains the details of the spline and GP comparison, and Section 5 provides a summary.

## 2 Scaling of Production of Gaussian Process Surrogates

Fundamentally the Gaussian Process (GP) is a stochastic interpolant between the sample data points. The wikipedia entry [3] helpfully shows that for Gaussian statistics, the GP amounts to a Brownian walk which takes in the sample points in turn. It will be seen why there is an optimal mean square error (MSE), in that if the amplitude of the walk is too small, then the interpolant will fail to change enough to go through consecutive points that have very different sample values, too large an MSE and it is becomes increasingly improbable that the walk will 'hit' the points. It also follows that it is vital to accurately characterise the statistics of the stochastic process as they are integral to the interpolation. Ordinary Kriging assumes that the data has a constant mean at least locally. There follows a description of universal Kriging that tries to fit a combination of stochastic function and a deterministic functional form (typically a polynomial) through the sample point.

The most comprehensive work appears to be Sacks et al [4], who suppose that, although the system under is deterministic (so that its output $y$ at a sample point $x$ should always be the same), because of errors $y(x)$ can be regarded as resulting from a random function *aka* stochastic process as follows:

$$Y(x) = \Sigma_{m=1}^{M} c_m f_m(x) + Z(x) = \mathbf{c} \cdot \mathbf{f}(x) + Z(x) \tag{1}$$

where the $f_m(x)$ are a basis that may be chosen by the modeller, $c_m$ are fitting parameters to be determined, and $Z(x)$ is a (separate) random process at every $x$ with zero expected value. The assumption represented by Equation (1) is referred to as universal Kriging, distinguishing it from the simple Kriging described in ref [5, § 2.3.1]. Further suppose that the experimental design, ie. the set of samples, is

$$\text{Experimental design } \mathbf{S} = (x_1, x_2, \ldots, x_N) \tag{2}$$

leading to observed values $\mathbf{y} = (y(x_1), y(x_2), \ldots, y(x_N))$. If $\mathbf{y}$ is all the information known about the system, then it is reasonable further to suppose that at any point $x$, $y(x)$ may be predicted as a linear combination of the observations

$$\tilde{y}(x) = \boldsymbol{\beta}(x) \cdot \mathbf{y} \tag{3}$$

where the $N$-vector of coefficients $\boldsymbol{\beta}(x)$ is chosen to minimise the mean square-error

$$MSE\left(\tilde{y}(x)\right) = \mathbb{E}[(\boldsymbol{\beta}(x) \cdot \mathbf{y} - Y(x))^2] \tag{4}$$

subject to the constraint

$$\mathbb{E}[(\boldsymbol{\beta}(x) \cdot \mathbf{y} - Y(x))] = 0 \tag{5}$$

Substituting Equation (1) in the argument of Equation (5) gives

$$\boldsymbol{\beta}(x) \cdot \mathbf{y} - Y(x) = \Sigma_{n=1}^{N} \beta_n(x) \left(\mathbf{c} \cdot \mathbf{f}(x_n) + Z(x_n)\right) - \left(\mathbf{c} \cdot \mathbf{f}(x) + Z(x)\right) \tag{6}$$

giving

$$\mathbb{E}[(\boldsymbol{\beta}(x) \cdot \mathbf{y} - Y(x))] = \mathbb{E}[\mathbf{c} \cdot \Sigma_{n=1}^{N} \left(\beta_n(x)\mathbf{f}(x_n) - \mathbf{f}(x)\right)] \tag{7}$$

since $\mathbb{E}[Z(x)] = 0$ at each value of $x$. Thus the constraint Equation (5) implies

$$\Sigma_{n=1}^{N} \beta_n(x)\mathbf{f}(x_n) = \mathbf{f}(x) \tag{8}$$

which may be written in matrix-vector form as

$$F^T \boldsymbol{\beta}(x) = \mathbf{f}(x) \tag{9}$$

where the size $N \times M$ matrix

$$F = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \dots & f_M(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_M(x_2) \\ & & \dots & \\ f_1(x_N) & f_2(x_N) & \dots & f_M(x_N) \end{pmatrix} \tag{10}$$

Substituting Equation (8) in $(\boldsymbol{\beta}(x) \cdot \mathbf{y} - Y(x))^2$, all that remains are the terms in $Z$, so

$$\begin{aligned} MSE\,(\tilde{y}(x)) &= \mathbb{E}[(\Sigma_{s=1}^n \beta_s(x) Z(x_s) - Z(x))^2] & (11) \\ &= \mathbb{E}[\left(\Sigma_{s=1}^n \beta_s(x) Z(x_s) . \Sigma_{q=1}^n \beta_q(x) Z(x_q) - 2\Sigma_{s=1}^n \beta_s(x) Z(x_s) . Z(x) + Z(x)^2\right)] & (12) \end{aligned}$$

Define the (normalised) correlation matrix by

$$\mathbb{E}[Z(x_i)Z(x_j)] = \sigma^2 R(x_i, x_j) \tag{13}$$

so that its entries $R_{ij}$ are symmetric by construction, and the vector

$$\mathbf{r}(x) = (R(x_1, x), R(x_2, x), \dots, R(x_n, x)) \tag{14}$$

then

$$MSE\,(\tilde{y}(x)) = \sigma^2\,(1 + \boldsymbol{\beta}(x) R \boldsymbol{\beta}(x) - 2\boldsymbol{\beta}(x) \cdot \mathbf{r}(x)) \tag{15}$$

which must be a minimum as a function(al) of $\boldsymbol{\beta}$ subject to the constraint system Equation (9). Introducing $\boldsymbol{\lambda}(x)$ as a $M$-vector of Lagrange multipliers corresponding to the constraint, it follows (since varying Equation (9) gives rise simply to a term $F^T$) that

$$R\boldsymbol{\beta}(x) - \mathbf{r}(x) + F\boldsymbol{\lambda}(x) = 0 \tag{16}$$

thus to obtain $\boldsymbol{\beta}(x)$ requires solving the coupled system

$$\begin{pmatrix} 0 & F^T \\ F & R \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}(x) \\ \boldsymbol{\beta}(x) \end{pmatrix} = \begin{pmatrix} \mathbf{f}(x) \\ \mathbf{r}(x) \end{pmatrix} \tag{17}$$

which it is convenient to rearrange as

$$\begin{pmatrix} R & F \\ F^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}(x) \\ \boldsymbol{\lambda}(x) \end{pmatrix} = \begin{pmatrix} \mathbf{r}(x) \\ \mathbf{f}(x) \end{pmatrix} \tag{18}$$

and once a solution $(\boldsymbol{\beta}(x), \boldsymbol{\lambda}(x))$ has been obtained then substituting for $R\boldsymbol{\beta}$ from Equation (16) in Equation (15) and using Equation (9) gives

$$MSE\,(\tilde{y}(x)) = \sigma^2\,(1 - \boldsymbol{\beta}(x) \cdot \mathbf{r}(x) - \boldsymbol{\lambda}(x) \cdot \mathbf{f}(x)) \tag{19}$$

The theory of block matrices in Section B specifically Equation (47), gives the solution to Equation (18) in terms of a negative inverse Schur complement $S = -S_C^{-1} = (F^T R^{-1} F)^{-1}$ as

$$\begin{pmatrix} \boldsymbol{\beta}(x) \\ \boldsymbol{\lambda}(x) \end{pmatrix} = \begin{pmatrix} R^{-1} - R^{-1} F S F^T R^{-1} & R^{-1} F S \\ S F^T R^{-1} & -S \end{pmatrix} \begin{pmatrix} \mathbf{r}(x) \\ \mathbf{f}(x) \end{pmatrix} = \begin{pmatrix} R^{-1} - F^{+T} S^{-1} F^{+} & F^{+T} \\ F^{+} & -S \end{pmatrix} \begin{pmatrix} \mathbf{r}(x) \\ \mathbf{f}(x) \end{pmatrix} \tag{20}$$

where the right pseudo-inverse of $F$, $F^+ = SF^TR^{-1}$ (note $SF^TR^{-1}.F = I$) has been inroduced. It is also worth noting that $S$ only exists if $N \geq M$ and it is then symmetric.

The formulae in the Sacks et al paper then follow from the solution

$$\begin{aligned} \boldsymbol{\beta}(x) &= (I - R^{-1}FSF^T)R^{-1}\mathbf{r} + R^{-1}FS\mathbf{f} = (R^{-1} - F^{+T}S^{-1}F^+)\mathbf{r} + F^{+T}\mathbf{f} & (21) \\ \boldsymbol{\lambda}(x) &= SF^TR^{-1}\mathbf{r} - S\mathbf{f} = F^+\mathbf{r} - S\mathbf{f} & (22) \end{aligned}$$

upon introducing

$$\tilde{\mathbf{c}} = SF^TR^{-1}\mathbf{y} = F^+\mathbf{y} \tag{23}$$

and using the fact that for a general matrix $F$ is symmetric, then for arbitrary vectors $\mathbf{x}$ and $\mathbf{z}$, $F^T\mathbf{x}\cdot\mathbf{z} = \mathbf{x}\cdot F\mathbf{z} = \mathbf{x}F\mathbf{z}$, so that Equation (21) yields

$$\tilde{y} = \boldsymbol{\beta}\cdot\mathbf{y} = \tilde{\mathbf{c}}\cdot\mathbf{f} + R^{-1}\mathbf{r}\cdot(\mathbf{y} - F\tilde{\mathbf{c}}) \tag{24}$$

Setting $\tilde{\mathbf{c}} = \mathbf{0}$ recovers simple Kriging.

Another formula is

$$MSE\left(\tilde{y}(x)\right) = \sigma^2\left(1 - \mathbf{r}(R^{-1} - F^{+T}S^{-1}F^+)\mathbf{r} - 2\mathbf{r}F^+\mathbf{f} + \mathbf{f}S\mathbf{f}\right) \tag{25}$$

## 3 Spline theory

This section consists of background material for splines based on wikipedia entries [6, 7, 8].

### 3.1 Spline interpolation

Suppose we wish to fit a curve piecewise to a set of $N_d + 1$ data points $\{(x_i, y_i)\}_{i=0}^{N_d}$, called *knots*, using $N_d$ functions, called *splines*, defined for each $i = 1, \ldots, N_d$ as $y = q_i(x)$ on the interval $[x_{i-1}, x_i]$. We could do this with $N_d$ linear functions defined on each of the intervals. However, this will not generally give a smooth result at the knots. In order to have the freedom to set the derivatives at the knots, we need to consider cubic (or higher-order) polynomials. These degrees of freedom are determined by specifying that the splines must be $C^2$-differentiable at the knots,

$$q_i'(x_i) = q_{i+1}'(x_i), \qquad q_i''(x_i) = q_{i+1}''(x_i), \tag{26}$$

for $i = 1, \ldots, N_d$.

The determination of spline coefficients in general is covered by de Boor [9]. In the cubic case, we have $N_d$ intervals each containing a cubic, we have $4n$ parameters to determine. At each of the $N_d - 1$ interior knots, we have 4 equations,

$$\begin{aligned} q_i(x_i) &= y_i, & (27) \\ q_{i+1}(x_i) &= y_i, & (28) \\ q_i'(x_i) &= q_{i+1}'(x_i), & (29) \\ q_i''(x_i) &= q_{i+1}''(x_i), & (30) \end{aligned}$$

giving $4n - 4$ conditions in total. At the edge knots, we only have

$$q_0(x_0) = y_0, \tag{31}$$
$$q_{N_d}(x_{N_d}) = y_{N_d}, \tag{32}$$

and so require only a further conditions to be able to determine all coefficients. There are different options for doing this, but a common choice is the “natural conditions”

$$q_0''(x_0) = 0, \tag{33}$$
$$q_{N_d}''(x_{N_d}) = 0. \tag{34}$$

Writing the system symmetrically, the problem becomes one of inverting a tridiagonal system, an $\mathcal{O}(N_d)$ complexity procedure.

## 3.2 Spline smoothing

This determines a cubic spline fitting function to observed noisy data, $Y_i = f(x_i) + \epsilon_i$, where $\epsilon_i$ is additive noise, assumed to have zero mean. The fitting function is $\hat{f}$ is the function that minimizes

$$\sum_{i=1}^{N_d} \left(Y_i - \hat{f}(x_i)\right)^2 + \lambda \int \hat{f}''(x)^2 \, \mathrm{d}x, \tag{35}$$

over all twice differentiable functions.

Detailed inspection of ref [7] suggests that once again a tridiagonal matrix inversion is needed to find the vector of values at the data points/knots, $\hat{f}(x_i)$. These are then used as coefficient in the expansion in spline function,

$$\hat{f}(x) = \sum_{i=1}^{N_d} \hat{f}(x_i) f_i(x), \tag{36}$$

where $f_i(x)$ are the set of spline basis functions.

## 3.3 Adaptive spline fitting

Multivariate adaptive regression spline is also known as MARS. “It is a non-parametric regression technique and can be seen as an extension of linear models that automatically models nonlinearities and interactions between variables.”

The basic idea is to replace a linear regression model with a piecewise linear regression model, using *hinge functions*,

$$\max(0, x - c) \qquad \text{or} \qquad \max(0, c - x) \tag{37}$$

where constant $c$ is called the knot. MARS model fits are sums of functions of three types:

- a constant,
- a constant times a hinge function, and
- a constant times a product of hinge functions.

The overall number of terms, and the number of hinge functions that may be multiplied together are parameters defined by the user; otherwise, the model is parameter-free.

$$\hat{f} = \sum_i a_i B_i(x) = a_0 + \sum_i a_i \max(0, x - c_i) \tag{38}$$

### 3.3.1 Implementation

The python package SKLEARN provides an implementation of MARS. MARS is itself a trademark, so the implementation of the MARS model is instead called EARTH, in Python PY-EARTH [10]..

### 3.3.2 Smoothness

MARS differs from spline fitting in that MARS splines are not smooth. This is potentially attractive, for many solutions to fluid dynamical problems exhibit shocks or at least internal/boundary layer behaviour where the flow solution exhibits local rapid variation that is hard to capture in detail, hence resembles discontinuity.

Nonetheless, SKLEARN offers a parameter called “smooth” that produces a differentiable fit, following [11, § 3.7]. That paper is interesting on the need to make smooth fits:

> If the intent is accurate estimation of the function (as opposed to its derivatives of various orders), then there is little to be gained by imposing continuity beyond that of the function itself. If the true underlying function nowhere has a very large local second derivative, then a small additional increase in accuracy can be achieved by imposing continuous first derivatives. Also, continuous (first) derivative approximations have considerably more cosmetic appeal. There is, however, little to be gained and in fact much to lose, by imposing continuity beyond that of the first derivative, especially in high dimensional settings.
>
> The difficulty with higher order regression splines centers on so called end effects. The largest contribution to the average approximation error (2), (3) emanates from locations x near the boundaries of the domain. This phenomenon is well known even in univariate smoothing ($N_d = 1$) and is especially severe in higher dimensions. As the dimension of the covariate space increases, the fraction of the data points near a boundary increases rapidly. Fitting high degree polynomials (associated with high degree regression splines) in these regions leads to very high variance of the function estimate there. This is mainly due to the lack of constraints on the fit at the boundaries

This implies that piecewise-linear models should be adequate in higher dimensions, however it is not clear "how high is high". It seems likely that an ordinary 2-D or 3-D flow field will benefit from higher order, and possible that a kinetic (5-D or 6-D) representation will also benefit.

A smooth fit could be achieved by using higher-order hinge functions,

$$b_q(x-t) = \max(0, (x-t)^q) = (x-t)^q_+, \tag{39}$$

with the lowest-order smooth fit being quadratic, $q = 2$. However, this suffers from the end effects mentioned above. Therefore instead, MARS uses a modified basis set with formulae that appear complicated but in fact which are essentially hinge functions with the corner smoothed over using cubic polynomials. These basis functions are denoted $C(x|s = \pm1, t_-, t, t_+)$, where $s$ corresponds to the sidedness of the function (the sign in usual hinge functions). Instead of being characterised by a single knot, $t$, these are characterised by three knots, $t_-$, $t$, and $t_+$. They are continuous and smooth, but have discontinuities in the second derivative at $t_\pm$. Central knots $t$ are placed in the same location as in the piecewise linear case. The side knots $t_\pm$ are positioned midway between the central knots.

## 4 Comparison of MARS and GPs

Here, the quality of approximate function fits to data from MARS and Gaussian Processes are compared. Data drawn from a smooth function is considered in Section 4.1, while data drawn from the same function with additional Gaussian noise is considered in Section 4.2. Non-smooth continous functions and discontinuous functions will be discussed in future reports.

The tests described are implemented in Python using the SKLEARN packages [12] for Gaussian Processes and py-earth [10] (a contributed SKLEARN package) for MARS. The latter implements the MARS algorithm as described by Friedman [11].

### 4.1 Fitting a noise-free smooth function

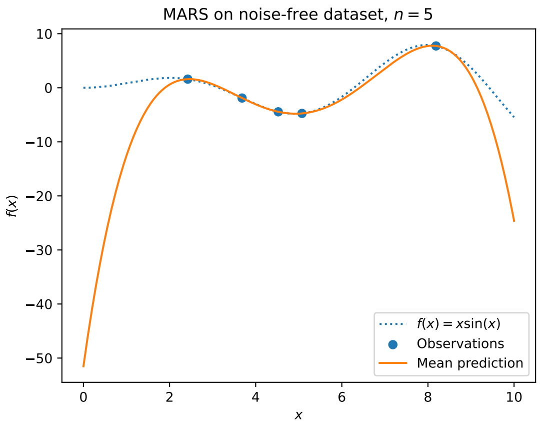
The first test function is

$$f(x) = x\ \sin(x) \tag{40}$$

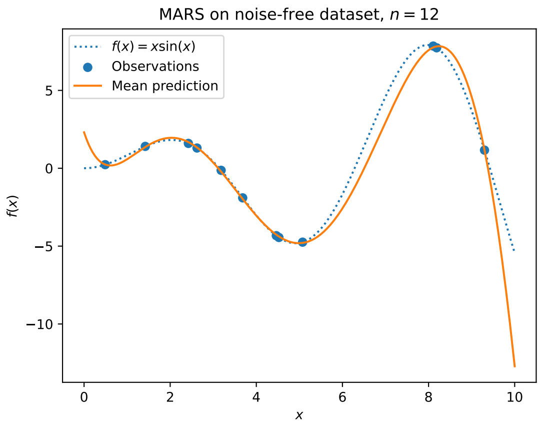
The function $f(x)$ is discretized on a uniform grid of 1000 points, and MARS and GPs are used to create fits for this function using a data sample of $N_d \ll 1000$ random training points. The seed of the random number generator is fixed so that both methods use the same training points, and moreover that the set of $N_d + 1$ training points is the same as the set of $N_d$ training points with one additional point.

#### 4.1.1 MARS

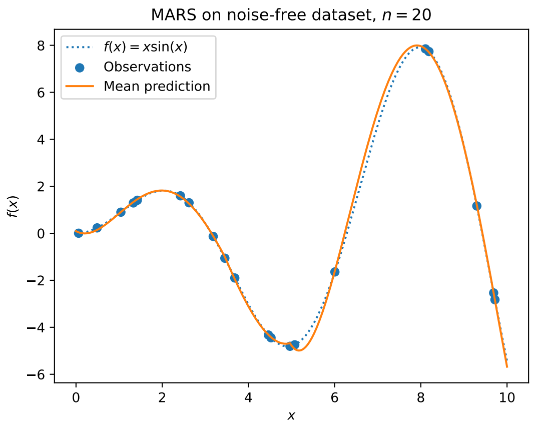
MARS fits to the function (40) are shown in Figure 1 for various $N_d$. These are generated using the smoothed hinge basis functions $C(x|s)$, allowing up to 100 summed terms, and for each term to



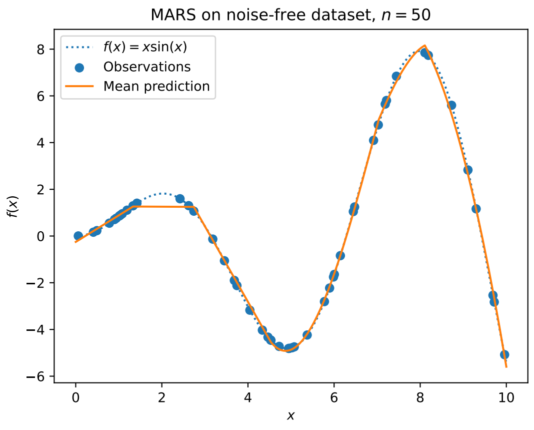
(a)



(b)



(c)



(d)

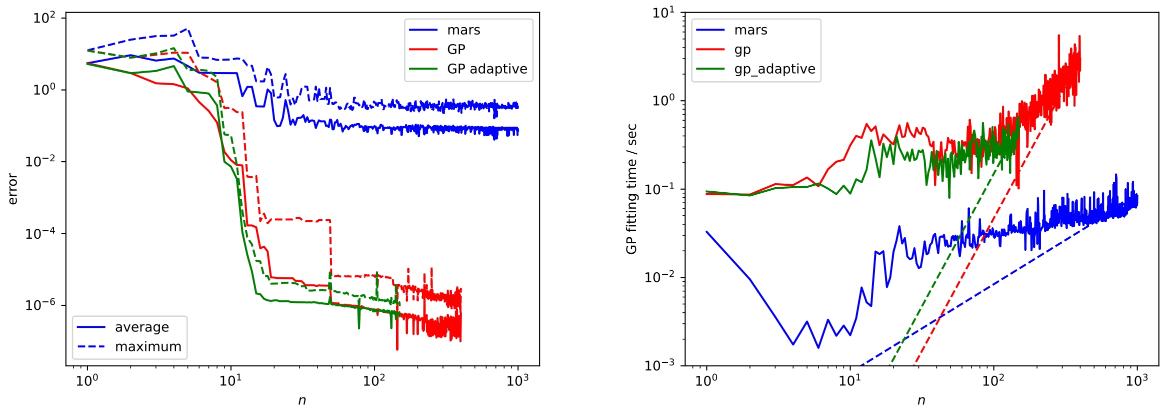
Figure 1: MARS fits to the function $f(x) = x\sin(x)$ for training sets of various sizes, $n = N_d$.

contain the product of up to 100 basis functions. This generates smoother fitting functions without a noticeable increase in computational cost.

The fits produced by MARS require at least $N_d = 10$ data points to be reasonable, but many more points ($N_d$ around $50$) to be numerically accurate. Even with many overall data points, regions of the domain with no data may be fitted poorly: see around $x = 2$ in the $N_d = 50$ plot, Figure 1d.

The error in the fit and run time of the code are shown for MARS as the blue line in Figure 2. To produce the error, the MARS model is generated using a training set $\mathcal{T}_{N_d}$ of $N_d$ point, $f_{MARS}(x|\mathcal{T}_{N_d})$, and evaluated on the full grid of 1000 points. The absolute error $|f(x) - f_{MARS}(x|\mathcal{T}_{N_d})|$ is plotted as a function of $N_d$. The relative error $|f(x) - f_{MARS}(x|\mathcal{T}_{N_d})|/|f(x)|$ is not used, as $f(x)$ is near zero in many places, making this metric uninformative. The error in Figure 2a decreases with increasing $N_d$ until reaching a plateau at around $N_d = 50$. That is, using a larger training set than $N_d = 50$ is not worthwhile, as the model fit does not improve.

The run time in Figure 2b is generated using Python internal timers to measure the fitting stage



(a) (b)

Figure 2: The errors (a) and run times (b) for MARS (blue), Gaussian Processes (red) and Gaussian Processes with adaptive sampling (green). ($n = N_d$ of text.)

of the algorithm only – the timings do not include sampling or function evaluations. Moreover, each $N_d$ is treated as a separate case. In principle, as the training sets are nested it is possible to leverage information from previous runs to reduce the cost of fitting, but that is not done here.

The cost of MARS increases roughly linearly in the range $3 \leq N_d \leq 30$, before increasing roughly like $N_d^{1/3}$ for $N_d \geq 30$. This compares favourably with the theoretical $\mathcal{O}(N_d)$ scaling.

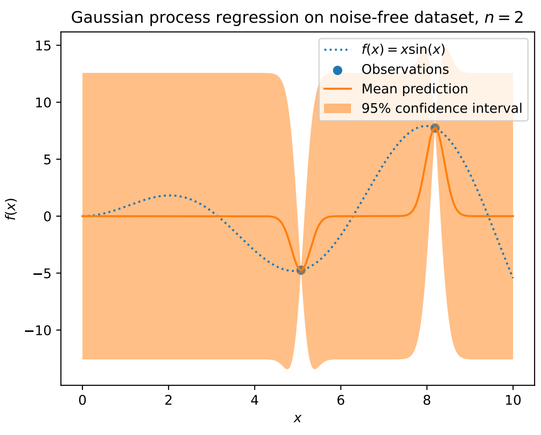
### 4.1.2 Gaussian Processes

Gaussian Processes fits to the function (40) are shown in Figure 3 for various $n = N_d$. In this noise-free case these are generated by interpolating data points using the RBF kernel, which is a distribution with squared exponential form (not a radial basis function). Along with the predicted function value, the GP fit also generates a standard deviation for the fit, from which confidence intervals can be produced: 95% confidence intervals are also shown as the shaded regions in Figure 3.

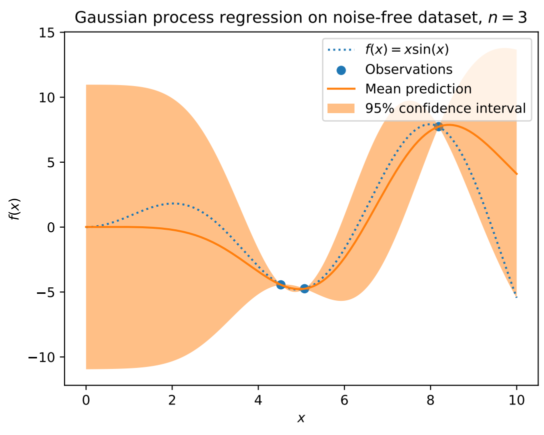
GPs produces meaningful results with as few as three data points (see Figure 3b). The fits become extremely accurate for $N_d \geq 8$ when interpolating within the training set (see Figure 3c), and are essentially perfect fits by eye by $N_d = 10$ (Figure 3d).

The absolute error of the GP fit is plotted as the red line in Figure 2a. The error drops off extremely rapidly in the range $8 \leq N_d \leq 15$ before settling at a small plateau. The decay in error is much faster than that achieved by MARS (blue), and the final plateau reached is five orders of magnitude smaller.

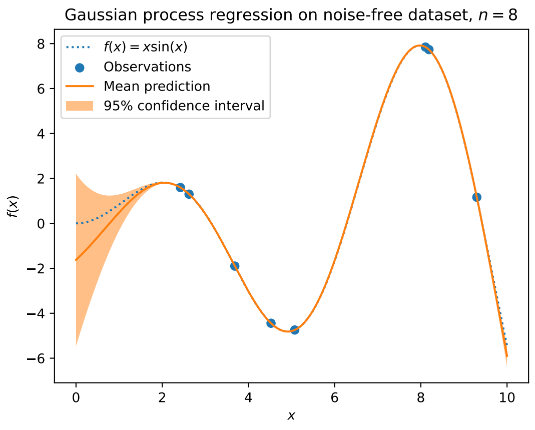
In the range where the error decays, there are large discrete jumps where the error decreases dramatically from $N_d$ to $N_d + 1$. These occur when the new training point added is outside the range of the existing training set. This means that the GPs are (accurately) interpolating over a



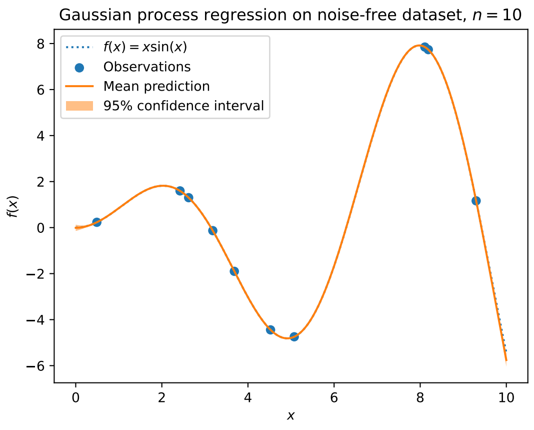
(a)



(b)



(c)



(d)

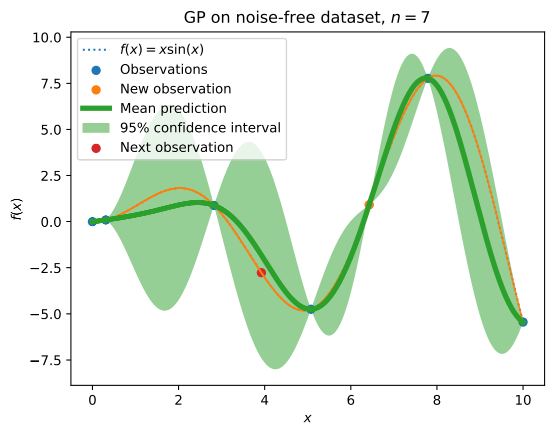
Figure 3: GP fits to the function $f(x) = x\sin(x)$ for training sets of various sizes, $n = N_d$.

larger range of the domain and (less accurately) extrapolating over a smaller range. The error is therefore much reduced.

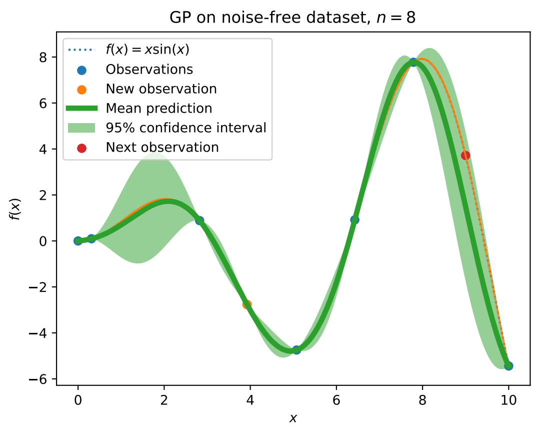
The run time for GPs is plotted as the red line in Figure 2b. Each case takes around ten times longer than a MARS fit, but for $N_d \lesssim 200$ requires less than 1 second. Moreover, the observed scaling is noticeably more favourable than the theoretical scaling of $\mathcal{O}(N_d^3)$, even at the largest values of $N_d$ considered.

### 4.1.3 Adaptive sampling with GPs

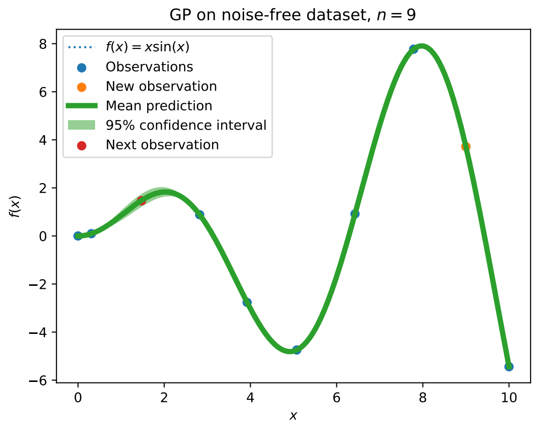
So far, randomly selected training sets have been used. However, it has been observed that some regions of the domain are fitted better than others – in particular interpolation within the training set is more accurate than extrapolation. This raises the prospect that faster convergence could be achieved, if the training points could be placed where needed.



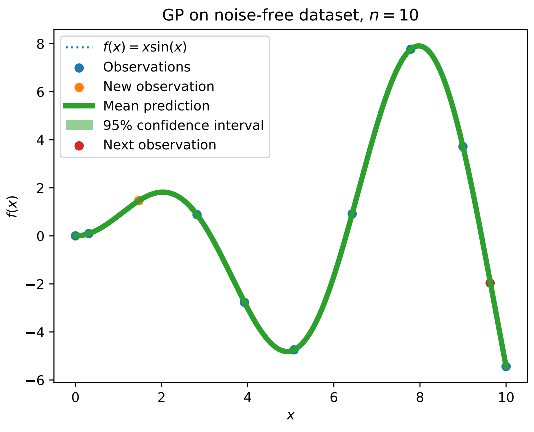
(a)



(b)



(c)



(d)

Figure 4: GP fits to the function $f(x) = x\sin(x)$ for adaptively sampled training sets of various sizes, $N_d$.

To achieve this, the confidence intervals produced by the Gaussian Process routine are used. To go from the training set $\mathcal{T}_{N_d}$ to the set $\mathcal{T}_{N_d+1}$, the new point is added at the position where the confidence interval is widest. $\mathcal{T}_1$ is chosen as a random point.

An example of this is shown in Figure 4 where fits for the cases $N_d = 7$ to $N_d = 10$ are given. In each plot, the red circle shows the function value at the position where the confidence interval is widest. That point is added to the training set and used in the fit for $N_d + 1$. Note that the new data point dramatically improves the function fit near that point, but also generally improves the fit across the whole domain. For example, comparing Figures 4a and 4b, the new point at $x = 4$ dramatically reduces the error near $x = 4$, but also reduces the error around $x = 2$ and $x \in [5, 8]$.

Note also that by $N_d = 7$ (Figure 4a), the extrema of the grid $x = 0$ and $x = 10$ have already been added to the training set. One property of this sampling algorithm is that it tends to immediately add the grid end-points so that there is no extrapolation.

The absolute error of this approach is plotted as the green line in Figure 2a. After some initial noise, the error is smaller than that for randomly-sampled GPs (red) and reaches the minimum error plateau sooner. This is not a dramatic improvement however, as the GP fit to a smooth function is already very good. Preliminary work suggests that adaptive sampling becomes more important when treating non-smooth or discontinuous functions.

The run time is plotted as the green line in Figure 2b. As the time only includes the GP fit and not the sampling, the behaviour is the same as that for GPs to within timing noise. Note that the adaptive sampling itself is cheap, amounting to finding the maximum value in a length 1000 vector that is produced automatically by the GP routine.

## 4.2 Fitting a smooth function with noise

In this second test, the same function $f(x) = x\sin(x)$ is considered but now with additional superimposed Gaussian noise with unit amplitude and standard deviations $\sigma \in \{0.01, 0.1, 1, 10\}$.

No modification is needed for MARS in this case, as it does not interpolate the training data. The GP approach however must be modified: in the preceding case, GPs interpolated the data which leads to overfitting and inaccurate models when the data is noisy. To deal with this, SKLEARN's `GaussianProcessRegressor` function is provided with the parameter $\alpha = \sigma^2$, which it interprets as the variance of additional Gaussian measurement noise on the training observations. Note that in a problem with real data, this step would entail making an estimate for the variance of the noise $\sigma^2$.

An example of a GP with $N_d = 10$ and $\alpha = \sigma^2 = 1$ is shown in Figure 5. Note that unlike in the noise-free case, the mean prediction does not interpolate the observations, and the confidence intervals do not vanish at the data points.

Otherwise, the behaviour of function fits is very similar to those shown in Figures 1, 3 and 4, and so these are not reproduced. Likewise, the timing plot is very similar to that in Figure 2b.

The significant difference is in the error graph, plotted in Figure 6. The absolute error is plotted against training set size $n = N_d$ for MARS (blue), GPs (red) and adaptively-sampled GPs (green), with different markers showing the value of $\sigma$. As in the noise-free case (Figure 2a), there is generally a reduction in error with $N_d$, and then a plateau where the error does not reduce further. As before, the error for GPs approaches decreases more quickly and, in the cases with smaller $\sigma$, attain a lower error plateau than the MARS approach. However, there is now the significant difference that by the very nature of having noisy data, there is a lower bound of the size of $\sigma$ that *any* model would be able to attain. This means that when the noise level is $\sigma \gtrsim 0.1$, both GPs and MARS produce models of the same accuracy, but MARS does so at around one-tenth of the computational cost. In cases where the is less noise $\sigma \lesssim 0.1$, GPs still produce much more accurate models.

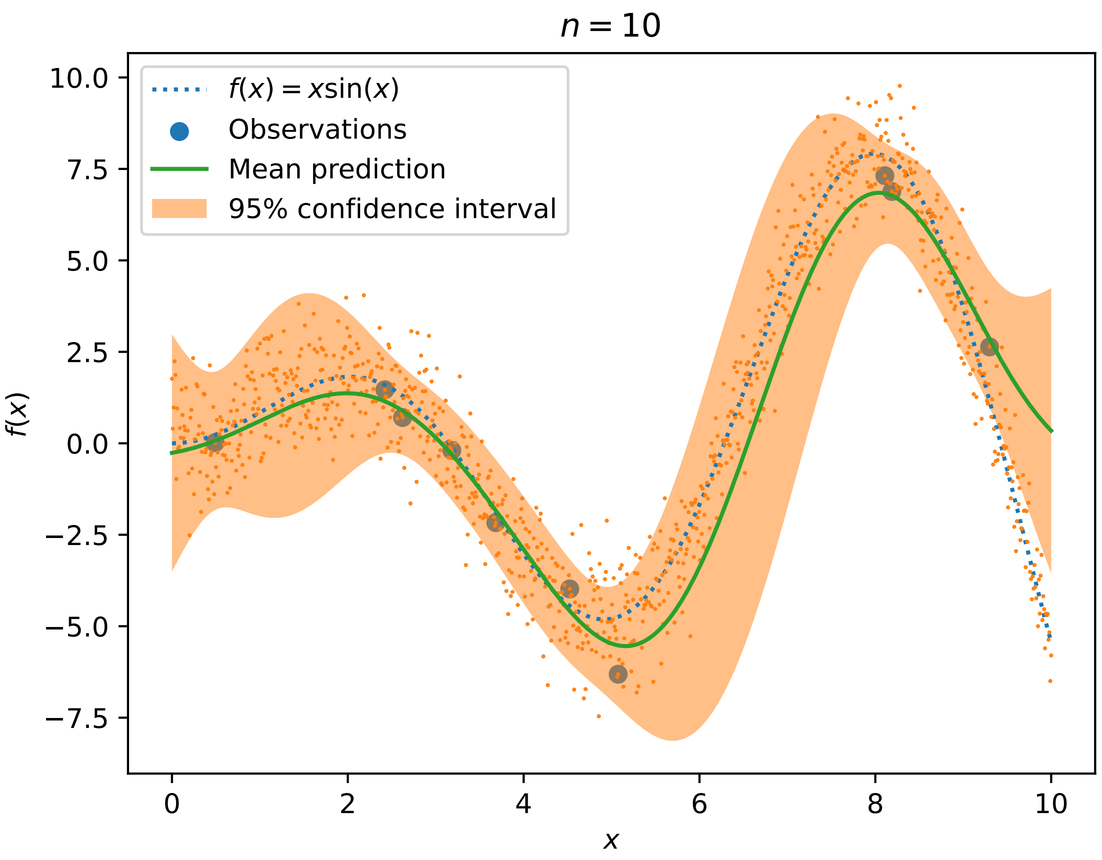


Figure 5: Fitting GP to noisy data

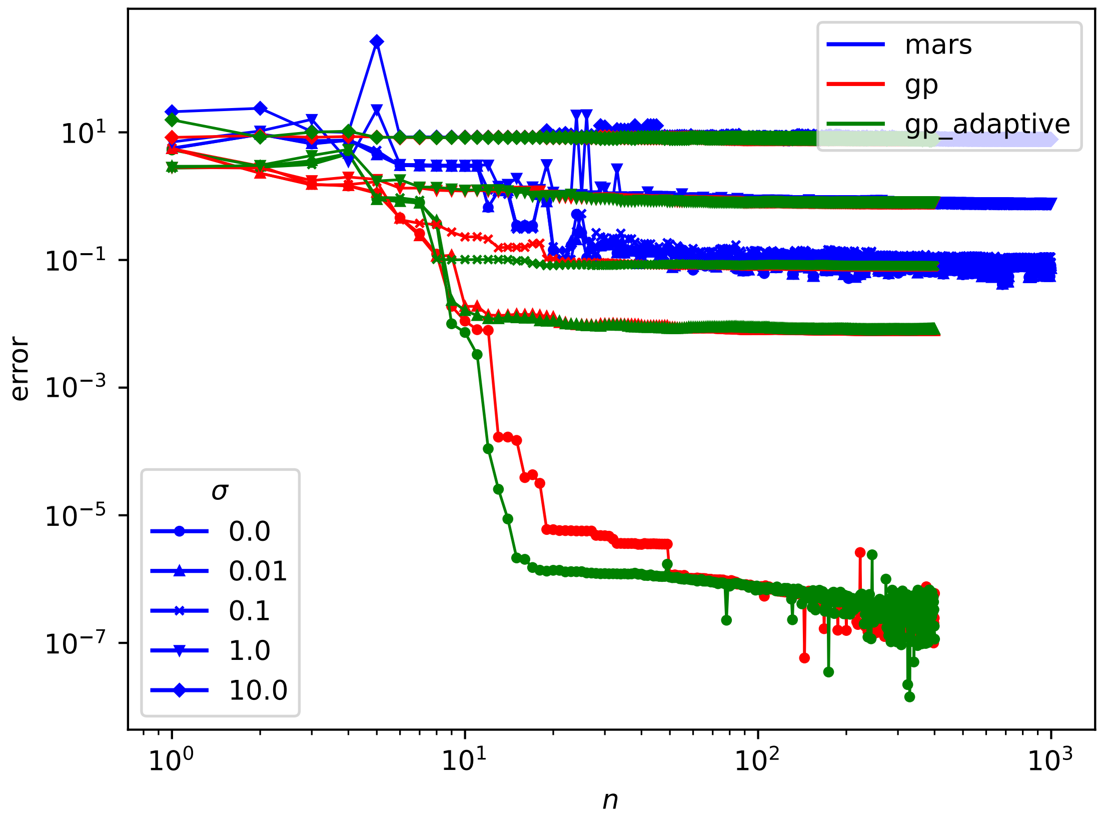


Figure 6

Figure 7: GPs and MARS. Fitting error for noisy data, as $n = N_d$ varies.

## 5 Summary

This report has compared MARS and GP fitting for smooth function both with and without added Gaussian noise.

In the smooth, noise-free case, GPs converge very quickly to the test function, requiring only $\sim 10$ training points. The computation is not expensive with each case running in less than a second, and the observed scaling is rather weaker than the theoretical $\mathcal{O}(N_d^3)$. Using an adaptively-sampled training set does increase the rate of convergence, but the improvement is not significant for smooth functions where randomly-sampled points already provides very quick convergence.

In contrast to GPs, MARS converges more slowly, requiring $\sim 50$ training points, and has quite a high error plateau so that it never converges exactly to the test function. The MARS computation is however around ten times faster than the GP computation, and the observed scaling is weaker than the theoretical $\mathcal{O}(N_d)$.

The behaviour of the fits is not significantly altered by the addition of Gaussian noise to the test function. The main difference is that the standard deviation of the random noise $\sigma$ sets a lower bound on the absolute error that any fitting algorithm can achieve. When the noise level is large $\sigma \gtrsim 0.1$, neither algorithm can attain a good fit, and so MARS is preferable as the cheaper approach. However, when the noise is small $\sigma \lesssim 0.1$, GPs are preferable as more accurate and still relatively inexpensive.

#### 5.0.1 Future work

This report has compared MARS and GPs approaches for fitting smooth functions with and without imposed Gaussian noise. These functions might represent mean heat transport by each of a set of a turbulent calculations. Regarding the structure of the turbulent flows themselves, it is of interest as to how these approaches handle rapidly varying, effectively discontinuous functions. This will be the subject of future reports, but preliminary work indicates the following:

- The quality of fits from all approaches deteriorates around discontinuities.
- To achieve good fits it is necessary to both use adaptively-sampled training sets, and to divide the domain into disjoint regions on which the test function is smooth and continuous.
- As such, the ability to identify discontinuities in data will be very important.

The last point implies eg. that fluid calculations should record information about where techniques such as shock-fitting have been employed to eliminate spurious oscillation, since these may in effect constitute discontinuities.

## Acknowledgement

*The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.*

## References

[1] W. Arter and E. Threlfall. Management of external research. Supports UQ Procurement. Technical Report CD/EXCALIBUR-FMS/0040-M5.1, UKAEA, 2021. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0040-M5.1.pdf.

[2] L.N. Trefethen and M. Embree. *Spectra And Pseudospectra: The Behavior of Nonnormal Matrices And Operators*. Princeton University Press, 2005.

[3] Kriging wiki. https://en.wikipedia.org/wiki/Kriging, 2020. Accessed: October 2020.

[4] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[5] W. Arter, E. Threlfall, and J. Parker. Report on user layer design for Uncertainty Quantification. Technical Report CD/EXCALIBUR-FMS/0024-M3.1.3, UKAEA, 2020. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0024-M3.1.3.pdf.

[6] Spline interpolation. https://en.wikipedia.org/wiki/Spline_interpolation, 2022. Accessed: March 2022.

[7] Smoothing spline. https://en.wikipedia.org/wiki/Smoothing_spline, 2022. Accessed: March 2022.

[8] Multivariate adaptive regression spline. https://en.wikipedia.org/wiki/Multivariate_adaptive_regression_spline, 2022. Accessed: March 2022.

[9] C. de Boor. *A practical guide to splines*. Springer, New York, 1978.

[10] Multivariate adaptive regression spline (MARS) software. http://contrib.scikit-learn.org/py-earth/content.html#api, 2022. Accessed: March 2022.

[11] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

[12] scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/index.html, 2022. Accessed: March 2022.

[13] G.H. Golub and C.F. Van Loan. *Matrix computations. 2nd Edition.* Johns Hopkins, Baltimore, 1989.

## A Annex A: Comparison of GP and NN surrogates

Both Gaussian processes (GP) and neural networks (NN) are basically techniques for interpolation, which may be shown to be very expensive to derive relative to spline-fitting.

1. GP
   (a) At each position/time $x$ there is a Gaussian random variable. However
   (b) the correlation between different $x$ need not be Gaussian, and
   (c) the correlation function $R(x, x')$ essentially defines how the interpolant is produced, at $N^3$ cost, $N$ is number of samples.
   (d) The correlation function has (hyper)parameters which may be calculated by optimisation - maximising the likelihood function over relatively small number of parameters, but optimiser may not converge
   (e) May do amazingly well as a consequence of the Central Limit Theorem
   (f) Likely to be stable when extrapolating if $R$ smooth
2. NN
   (a) Defined by localised functions at nodes, in general functions of functions of . . . , where the basic function usually has simple form
   (b) Fitting by optimisation techniques for number of parameters $\propto$ number of nodes - techniques rely on having many samples and optimiser may be slow to converge or overfit
   (c) Theorems that NN approximation has resistance to “curse of dimensionality”
   (d) Usually unstable when extrapolating
   (e) Lack of routine estimates for error in surrogate

## B Annex B: Advanced Results in Matrix Theory

The inverse of a block matrix, ie. a matrix written as

$$B = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ structure } \begin{pmatrix} N \times N & N \times M \\ M \times N & M \times M \end{pmatrix} \tag{41}$$

may be deduced from the formulae obtained using Gaussian elimination for $2$ simultaneous linear equations (expressed using the $2 \times 2$ matrix $A$)

$$A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \text{ where } A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{42}$$

As is well-known, Gaussian elimination proceeds by scaling the $x$-coefficient of the first equation to be equal to that $a_{21}$ of the second, ie. by the factor $a_{21}a_{11}^{-1}$ and then subtracting the two equations. This may be expressed as a matrix multiply, viz.

$$\begin{pmatrix} 1 & 0 \\ -a_{21}a_{11}^{-1} & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ 0 & s_c \end{pmatrix} \tag{43}$$

where the coefficient

$$s_c = a_{22} - a_{21}a_{11}^{-1}a_{12} \tag{44}$$

It follows that the inverse of $A$, provided $s_c \neq 0$ is

$$A^{-1} = \begin{pmatrix} 1 & a_{21}^{-1}a_{12} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11}^{-1} & 0 \\ 0 & s_c^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -a_{21}a_{11}^{-1} & 1 \end{pmatrix} \tag{45}$$

The Schur complement is the analogue of the $s_c$ factor for block matrices, viz.

$$S_C = A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{46}$$

when writing Equation (45) with $a_{ij} \to A_{ij}$ ($s_c \to S_C$, invertible) gives the block factorisation. Multiplying out the product gives

$$B^{-1} = \begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}S_C^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}S_C^{-1} \\ -S_C^{-1}A_{21}A_{11}^{-1} & S_C^{-1} \end{pmatrix} \tag{47}$$

## B.1 Additional Results

Using the generalised Sherman-Morrison-Woodbury formula, the first entry simplifies. The formula, from Golub & van Loan [13, Eq. (2.1.4)] is

$$(T + UV)^{-1} = T^{-1} - T^{-1}U(I + VT^{-1}U)^{-1}VT^{-1} \tag{48}$$

where $T$ is an $N \times N$ matrix, $U$ is $N \times M$ and $V$ is $M \times N$ (cf. use of $V^T$ in [13]). To apply, write $U = -WC^{-1}$, then

$$(T - WC^{-1}V)^{-1} = T^{-1} + T^{-1}W\left[C^{-1}(I - VT^{-1}WC^{-1})^{-1}\right]VT^{-1} \tag{49}$$

and observe that the term is square brackets simplifies on remembering that where inverses of matrices $S_j$ exist, $S_1^{-1}S_2^{-1} = (S_2S_1)^{-1}$, so that

$$(T - WC^{-1}V)^{-1} = T^{-1} + T^{-1}W\left[C - VT^{-1}W)\right]^{-1}VT^{-1} \tag{50}$$

With the obvious substitutions, this implies

$$(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} = A_{11}^{-1} + A_{11}^{-1}A_{12}[S_C]^{-1}A_{21}A_{11}^{-1} \tag{51}$$

Thus

$$B^{-1} = \begin{pmatrix} (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & -A_{11}^{-1}A_{12}S_C^{-1} \\ -S_C^{-1}A_{21}A_{11}^{-1} & S_C^{-1} \end{pmatrix} \tag{52}$$

There are results of the form

$$(T - UV^{-1}W)^{-1}UV^{-1} = T^{-1}U(V - WT^{-1}U)^{-1} \tag{53}$$

perhaps more memorable as

$$(T^{-1} - UVW)^{-1}UV^{-1} = TU(V^{-1} - WTU)^{-1} \tag{54}$$

These are easily proven by multiplication by the terms in parenthesis, when (in the second case Equation (54), two terms in $UVWTU$ cancel. The importance of Equation (53) is that the off-diagonal terms in $S_C^{-1}$ in Equation (52) may be replaced by terms expressed using $(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1}$.