Exploring the use of Gaussian Process Emulation as a cheaper alternative to simulation for a Complex Chemical System

Daniel Ellis

Department of Physics, University of York, York YO10 5DD,United Kingdom

May 4, 2015

Abstract

The atmosphere contains millions of compounds each with tens of millions of reactions. Representation of these is usually through the use of simultaneous ordinary equations. This results in a numerically stiff system; whose numerical burden can prove crippling to current air-quality and climate models. Since the end of the line has been reached for finessing the integration techniques used in these systems, new paradigms for the representation of the chemistry are required. One possible solution to these lies in the use of statistically robust, multi-dimensional curve fitting approaches such as Gaussian Process Emulators. This paper looks at the viability of using Gaussian processes to replace the currently computation intensive simulators.

Predictive capability and performance are two of the areas that were explored. It was concluded that although the emulator applies a smoothing effect to the data, this could be improved with the addition of extra training points. Additionally most of the data of a correctly fitted Gaussian process was found to lie within 5% percentage from the simulated value. It was shown that the emulator can be extended to n-dimensions and that it proved significantly faster than its simulator counterpart, especially for longer simulations. For this reason the paper concludes that with the correct design, it is possible to emulate simple atmospheric chemistry.

Word Count: 7477

Words in text: 7477

Words in headers: 274

Number of math inlines: 129

Number of math displayed: 16

Acknowledgements

I would like to thank my supervisors, Prof. Mathew Evans (Chemistry) and Dr. Philip Hasnip (Physics), for the continuous guidance in both the assessment criteria and project content.

I would also like to express my gratitude to Dr. Pete Edwards for the countless hours spent explaining atmospheric chemistry and inner workings of the DSMACC model, without which progress in the project may not have been as fruitful.

Contents

Exploring the use of Gaussian Process Emulation as a cheaper alternative to simulation for a Complex Chemical System

Daniel Ellis

Department of Physics, University of York, York YO10 5DD,United Kingdom

May 4, 2015

1 Background

The International Panel of Climate Change's fifth report advocates a correlation between global warming and the increasing presence of extreme weather, rising sea levels and imminent extinction of certain biological species [NCADAC, 2014, Thomas et al., 2004, Cazenave and Nerem, 2004]. Increase in global-mean surface temperature can be attributed to the increase in greenhouse gas concentrations [IPCC, 2013]. Gases within this category absorb infra-red radiation contributing to the radiative forcing[1] on the Earth system. Since the repercussion of the ever increasing production of anthropogenic greenhouse gasses is unknown, exploration into the effects of these is imperative.

Although research into concentration and emission of atmospheric gas-phase species can be conducted locally, the effects of turbulence, chaos and tropospheric flow result in an interaction on the global scale. This introduces an additional complexity, often resolved by using a statistical approach, concatenating data from several locations and creating a mathematical model representative of atmospheric reactions within that region. This allows the use of mathematical models to explore 'what if' scenarios that are probable in in the near-future.

[1]The influence a factor has in altering the balance of incoming and outgoing energy in the Earth-atmosphere system [IPCC, 2007]

1.1 Simulations in Science

The first use of computational experimentation dates back to 1944 where quantitative investigation of hydrodynamics of nuclear implosion proved problematic. Here hand calculations proved infeasible and approximate methods were un-interpretable. John von Neumann realised that the computerisation of complicated numerical equations could lead to an acceleration of the progress in hydrodynamics. For this he helped in the build and design of the first electronic computer, and the algorithms used to solve the partial differential equations required by implosion hydrodynamics. This was a great success and was suitably reflected in the technical report, suggesting that without the computational group, the work of the Theoretical division would have been impossible
[Los-Alamis-National-Labratory, 1944].

Analogously, the complex nature presented by the study of meteorological, oceanic and land-surface interactions are another field where electrical computation can be applied to facilitate the practiacl application of numerical prediction [Lynch, 2008]. Ideas regarding the numerical models and the climate were first developed in the late 19th century by Abbe, who realised that 'meteorology is essentially the application of hydrodynamics and thermodynamics to the atmosphere' [Willis and Hooke, 2006]. However due to the impracticality of manual calculation, it was not until Neumann's work, that the simulation of these became a practical reality.

In addition to statistical analysis and nuclear implosions, von Neumann held an interest in weather pre-

diction. He believed that in a stalemate over Nuclear Weapons, this could be used to gain an advantage over the Soviets [Jamieson, 2014]. In 1945, Newton joined Jules Charney in convincing the Weather Bureau, Air Force and Navy to establish a joint Numerical Weather Prediction unit. As part of this he later published an article attributing global warming to an increase in Carbon Dioxide [von Neumann, 1995].

Since initial meteorological simulation relied on the application pre-defined laws of thermo- and hydrodynamics, it soon became apparent that such simplistic models are insufficient to replicate the complexity of the atmosphere. Early simulations, such as those run on ENIAC[2], were capable of successful 24 hour weather prediction. Unfortunately, although a great feat, the delay in simulation time resulted in a computation that only just kept up with real-life events [Lynch, 2008]. in 1995 a general circulation model of the atmosphere was created by Phillips [Phillips, 1956]. It was soon discovered that this was insufficient to represent the Earth, and a general Global Climate Model (GCM) was formed from the union of atmospheric, oceanic, cryospheric and land-surface models [NIPCC, 2010]. The outcome of this was a program capable of simulating accurate short-term weather predictions, which can allow an insight into the contributing factors of climate change.

1.2 UKESM1

In 2013 the UK Met Office and the Natural Environmental Research Council (NERC) anounced the development of a new Earth System Modelling suite, UKESM1. Earth System Model (ESM) are a type of GCM that incorporate all aspects[3] of the Earth system. Upon completion, the UKESM1 will be capable of simulating 'climate and Earth system change over past and future decades to centuries' as will facilitate the study of climate and the Earth system's response to idealised forcing on these timescales [Jones, 2013].

[2]Electronic Numerical Integrator and Computer - The first general-purpose computer.

[3]physical, chemical and biological

1.3 The Problem

Forward progression of time in an ESM requires continuous output from all atmospheric, land, sea and ice processes. This means model progression is still constrained by the computation time of its slowest process. In such systems, even with reduced chemistry, atmospheric interactions act as such, with the Chemistry often taking twice as long to compute as the sum of all other processes [Evans, 2014].

The reason behind this is that the Atmospheric chemistry contains thousands of reactions
[Emmerson and Evans, 2009], each with its own set of properties [4]. This results in a numerically stiff system that requires specific integration approaches. Although the use of coarse grids, or simplified chemistry, can alleviate some the numerical burden on the parent model, this often comes at a cost to the quality of science produced. It is for this reason that the end of the line has been reached in finessing integration techniques for atmospheric models and a new paradigm for representing the chemistry is required [Evans, 2014].

1.4 The Solution

Instead of achieving incremental increases in computational speed through refinement and optimisation of the code, the solution would appears to fall in finding a new way to represent the chemistry in air-quality and climate models. One such a solution would be the use of Gaussin Process Emulators (GPE).

1.4.1 Emulators and their uses

To understand a GPE, one must first define the terms of a simulator. A simulator is a mathematical model capable of representing a real life system or phenomenon, in our case the Atmosphere. Here an emulator can be used to reduce the computational burden, and eliminate the need to reduce simulation time, or use low resolution grids. An emulator will apply statistical mechanics to such a system, representing the mathematical function that was previously known as a simulator.

[4]Reactivity, radiative forcing, lifetime, inter-specie interaction etc.

1.4.2 The Gaussian Process Emulator

Gaussian Process Regression was introduced to the field of geo-physics by Krige in 1950
[Krige, 1951, Kuß, 2006]. GPEs from a Bayesian viewpoint were developed in 1970 from the analysis of the behaviour inside an unknown mathematical function [Kimeldorf et al., 1970]. Statistical papers advocating the GPEs over functions were written by
[O'Hagan and Kingman, 1978], and the fundamental idea of building a GPEs from the non-Bayesian framework is developed in [Sacks et al., 1989].

1.4.3 The mathematics behind Gaussian Process Emulators.

An emulator works by predicting the output of a simulator. This can be thought of as a function, $\eta(.)$, whose inputs, $\mathbf{x}(x,\ldots,x_p)\ \epsilon\ \chi \subset \mathbb{R}^p$, correspond to the output,$y = \eta(\mathbf{x})\epsilon\mathbb{R}$. This is of great complexity and considered unknown. The emulator then uses a particular mean, $m(.)$, and covariance function, $V(.,.)$ to map the simulator output.

By convention, if the simulation contains a Gaussian Process distribution, for every $n\epsilon\mathbb{Z}^+$, the joint distribution of $\eta(\mathbf{x}_1),\ldots,\eta(\mathbf{x}_n)$ is multivariate normal $\forall$ $\mathbf{x}_i\epsilon\chi$ (where $i = 1,2,\ldots,n$). $m(.)$ can be any function $\mathbf{f}(\mathbf{x})$ where $\mathbf{x}\ \epsilon\ \chi$ and the covariance function must satisfy the property that $V(\mathbf{x}_i,\mathbf{x}_j)$ satisfies $\mathbf{x}^\dagger V\mathbf{x} > 0$ where all $\mathbf{x}$ are a non-zero positive reals $\mathbf{x}\epsilon\mathbb{R}^{+\mathbf{x}}_{\neq 0}$.

Any initial beliefs about the simulator can be represented using the mean and covariance functions: $m_0(.)$, $V_0(.,.)$. Using the hierarchical formulation, and the notion that $m_0(.) = h(.)^T\beta$, it can be stated that:

$$\eta(.)|\beta,\sigma^2,\delta\ \backsim\ GP(m_0(.),v_0(.,.)) \tag{1}$$

$$h(.) : \chi \subset \mathbb{R}^p \longmapsto \mathbb{R}^q \tag{2}$$

where $h(.)$ is a known function of the inputs chosen such that it incorporates prior knowledge of the simulator. For these, q can vary from the input space dimension, p, and β is an unknown q-dimensional vector of the coefficients.

$$V_0(.,.) = \sigma^2 C_\delta(.,.) \tag{3}$$

The covariance function is given in (3) , where σ^2 denotes an unknown scale parameter. $C_\delta(\mathbf{x},\mathbf{x\prime})$ represents the correlation between the input $\mathbf{x},\mathbf{x\prime}$ and correlation parameter, δ. Coupling this with equation of a Gaussian Function gives the Gaussian Correlation Function:

$$C_\delta(\mathbf{x},\mathbf{x\prime}) = \exp\{-|\mathbf{x}-\mathbf{x\prime}/\delta|^2\},\ for\ \mathbf{x},\mathbf{x\prime}\epsilon\chi \tag{4}$$

A more thorough derivation of the mathematics can be found in [Bastos, 2010] [Bastos and O'Hagan, 2009].

1.4.4 A visual example of the emulator.

GPEs work by mapping a set of polynomial curves to a set of input data based on the probability the output is within a specific region. To provide an accurate fit, the data provided needs to cover the entirety of input space. To do this [McKay et al., 2000] suggests the use of a Latin Hypercube design. This works much like a sudoku puzzle, where selecting data points with no horizontal or vertical overlap and can be used to provide complete spatial coverage.

[O'Hagan, 2014] shows how GPEs can be used to fit a set of data. In a 2 point design, Figure 1, it is possible to show uncertainty between points using 95% confidence lines(dashed). Here maximum uncertainty occurs at a distance furthest between training points. At the location if each point, uncertainty within the emulation is zero.

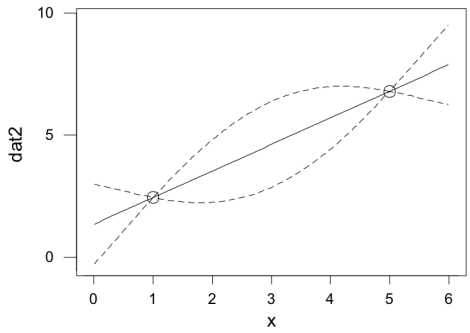


Figure 1: Demonstration of a Gaussian Emulator with 2 training points [O'Hagan, 2014]

Increase in the number of training points is shown to provide an improvement of the emulator. This reduces the maximum uncertainty, Figures (2 - 3).

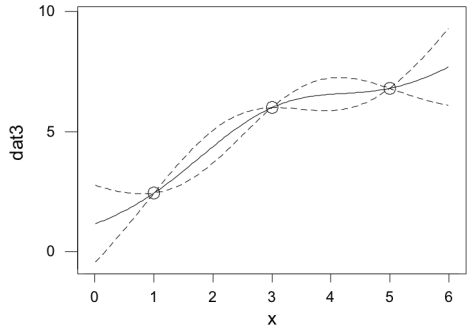


Figure 2: Demonstration of a Gaussian Emulator with 3 training points [O'Hagan, 2014]

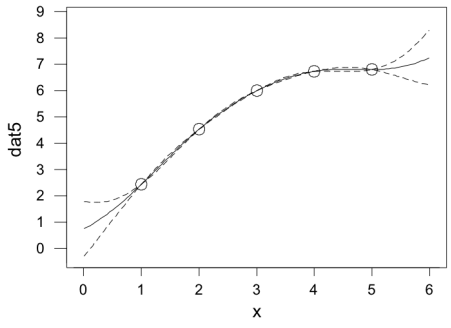


Figure 3: Demonstration of a Gaussian Emulator with 5 training points [O'Hagan, 2014]

Although an increase in the number of training points is seen to increase the accuracy of the emulator, an abundance of points will significantly decrease the computational gain in emulation over simulation, and instead a series of carefully selected input points should be used.

1.4.5 How will the GPE be calculated

For the project GPES shall be implemented through the use of DiceKriging [Roustant et al., 2014] and DiceOptim
[Ginsbourger et al., 2013]. [5]. These are R packages that calculate the coherence and fidelity of GPEs as well as allowing Kriging-based global optimisation criteria, algorithms and parallelism.

1.5 How to simulate atmospheric chemistry.

Computer simulations tend to decompose global simulations into a series of atmospheric columns, Figure 4. These columns can then be broken up into cubes of elevation. These cubes are effectively box models. Here each box output serves as an input to its neighbouring.

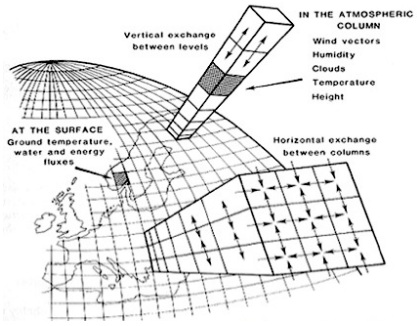


Figure 4: Splitting the surface of the earth into a series of box models. Source: [Henderson-Sellers, 2015]

1.5.1 The Box Model

The simplicity and ability to accurately simulate a large range of parameters and complexities make the Dynamically Simple Model of Atmospheric Chemical

[5] *R* packages produced by the Deep Inside Computer Experiments (DICE) consortium. This concentrates on the production of innovative mathematical methods specifically for the design and analysis of computer methods.

Complexity (DSMACC) box model an ideal candidate for the use on GPEs. Developed in the University of Leeds by Mat Evans and Kathryn Emmerson[6], its initial uses include the juxtaposition of chemistry schemes, when removed from their parent models. This was vital in the identification of sources of error for some chemical transport models [Emmerson and Evans, 2009].

DSMACC is designed to simulate tropospheric composition and calculate the expected calculations of atmospheric species. It is based on the Kinetic PreProcessor format [Sandu and Sander, 2006], which converts a set of chemical reactions and their rate coefficients into a series of forward integratable ordinary differential equations. The model is also compatible with the Master Chemical Mechanism (MCM), a near explicit chemical mechanism describing detailed gas-phase chemical processes in the troposphere [Rickard et al., 2015b]. Finally as source code is provided, a complete set of results is attainable, and modifications can be made if required.

1.6 Emulators in Literature

Many recent applications of GPEs in statistical climate prediction have been by Lindsey Lee. These often cover the exploration of sensitivity analysis and uncertainty in a complex global aerosol model. In her 2011 paper, GPEs are used in the estimation of the percentage output variance caused by uncertainty of the Global Model of Aerosol Processes (GLOMAP). Here it was concluded that with an appropriate experimental design and number of model runs, the uncertainty of the emulator is very small when compared to the original uncertainty of the model [Lee et al., 2011]. In a more recent paper, sources of uncertainty in aerosol-cloud interactions were explored using a variance-based analysis of a global three dimensional model. To allow a full statistical analysis of the outputs, a 3-D aerosol micro-physics model is used in exploring the causes of parametric uncertainty. Here GPE are built for each model grid cell, and Monte Carlo type sampling is employed in the uncertainty analysis calculation [Lee et al., 2013]. The results of these experiments depict the suitability of GPEs as a cheaper alternative for the use in complex atmospheric models.

Other uses of GPE in literature include the carrying out of sensitivity analysis on climate data by [Sanderson et al., 2008] and in [Andrianakis and Challenor, 2011]; a technical report testing the reliability of super-parametrisation and emulation in the field of deep water convection. In [Andrianakis and Challenor, 2011] several models: a reference model, a coarse model with convective adjustment, a super parametrised model and an emulated super-parametrised model are compared. The conclusions drawn from this experiment were that even-though the super parametrised models are only marginally better the coarse model, there is still a large potential for improvement. What is more important is the likeness of both the emulated and non-emulated super-parametrised model. This result proves promising for the use of emulation as a method of reducing computational cost whilst still maintaining crucial accuracy.

1.7 Summary and Intentions

Complexity in the chemistry of the atmosphere applies a considerable strain on global models. As code optimisation only provides a marginal speed-up, the end of the line has been reached with current integration techniques, and new paradigms to simulation need to be found. One possible solution is the use of meta-modeling and emulation.

This project will explore the feasibility of using Gaussian Process Emulators(GPE) as a means to simulate atmospheric interaction. This will be done using the DSMACC box model [Emmerson and Evans, 2009] and will follow the structure of Figure 5. Once a validated emulator is attained, this will be benchmarked against the simulator, and its computational performance compared.

[6]Subsequent testing and developments were conducted by Barron Henderson (UNC), Dylan Millet (UMN), Micheal Berkley(U. Edinburgh),and Daniel Stone (U. Leeds)

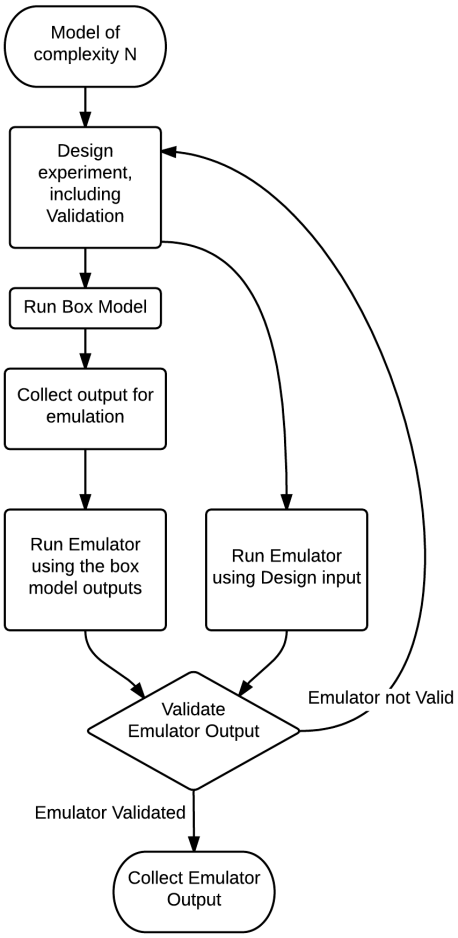


Figure 5: A flow chart of the testing and evaluation process of an emulator.

It is expected that for longer and more complex simulations, a carefully trained GPE will provide a significantly faster computation whilst maintaining the majority of the simulator information.

2 Understanding the Chemistry

Scientific research stems from the desire to obtain a complete understanding of nature and its effect on humanity. Ozone, O_3, plays a vital role in human health, the climate and food security [Shindell et al., 2012]. Its location in the atmosphere determines the effect it has on living organisms. For this reason higher (stratospheric) and lower (tropospheric) ozone are sometimes referred to as 'good' and 'bad' ozone respectively.

2.1 Stratospheric Ozone

90% of the total atmospheric ozone resides in the stratosphere [Vallero, 2014]. This is formed through the photolysis of O_2 from solar photons, (5). This type of ozone serves to protect the Earth from harmful UV radiation, reducing crop damage, skin-cancer and immune-system imparements. Absence of this will be the end of all surface life [Mathez and Webster, 2013].

$$\begin{aligned} O_2 + hv &\rightarrow O + O \\ O + O_2 &\rightarrow O_3 \end{aligned} \tag{5}$$

2.2 Tropospheric Ozone

Tropospheric ozone on the other hand not only contributes to global warming, but it can also damage the repository tissues of both plants and animals [McKee, 1993]. Its primary production in the troposphere has been shown to be the reaction of primary pollutants such as NOx ($NO + NO_2$) and VOCs (Volatile Organic Compounds) [Platt and Stutz, 2008].

In the absence of VOCs, ozone production is the result of the splitting of NO_2 using a photon, Figure 6. Heat is absorbed from the reaction through M (any atmospheric molecule), following which a single oxygen atom combines with O_2 to form ozone:

$$\begin{aligned} NO_2 + hv &\rightarrow NO + O(^3P) \\ O(^3P) + O_2 + M &\rightarrow O_3 + M \end{aligned} \tag{6}$$

This reaction also works in reverse, where NO can be oxidised to produce $NO2$:

$$O_3 + NO \rightarrow NO_2 + O_2 \tag{7}$$

It can be seen that steady state will eventually be reached between the Ozone-NOx cycle:

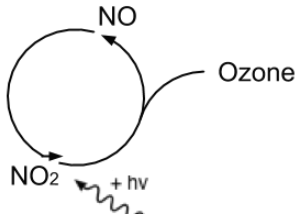


Figure 6: A visual representation of the production / loss of ozone through its interaction with NO and NO_2

This is shown in the Leighton Relationship:

$$\frac{NO}{NO_2} = \frac{J_{NO_2}}{k_{NO+O_3}} \tag{8}$$

Here ratio of $NO : NO_2$ is given by the frequency of NO_2 photolysis (ozone production) divided by the rate of ozone decomposition (due to a reaction with NO) [Finlayson-Pitts and Pitts, 2000].

2.2.1 The Role of VOCs

Net-ozone is production occurs with the oxidation of VOCs, forming carbon dioxide and water. This happens through the reaction of a hydroxyl, peroxy (O-O single bond) radical and NO:

$$HO_2 + NO \rightarrow NO_2 + OH \tag{9}$$

OH can then be converted into HO_2 or RO_2, where R can be hydrogen or any organic fragment (10).

$$OH + CO \rightarrow H + CO_2$$

and

$$RH + OH \rightarrow H_2O + R$$

$$R + O_2 + M \rightarrow RO_2 + M \tag{10}$$

HO_2 and RO_2 then react with NO closing the HO_x/RO_x cycle, (9) & (11) - Figure 7.

$$RO_2 + NO \rightarrow NO_2 + RO \tag{11}$$

Since HOx and NOx are recycled within the reaction, under certain conditions ozone production can be highly efficient [Crutzen, 1973].

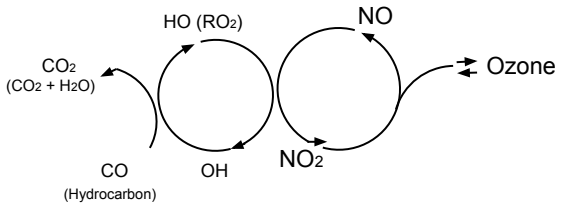


Figure 7: The Hydroxyl-NOx and Ozone cycles.

2.2.2 The Influence of H_2O.

For a net ozone-production to exist, there has to be a HOx-NOx cycle. Hydroxyl radicals, used in section 2.2.1, are predominatly formed through the reaction of ozone and water, (12). It therefore follows that a system devoid of water will simply be reduced to that of the Ozone-NOx cycle, Figure 6.

$$O_3 + h\nu \rightarrow O(^1D) + O_2$$

$$O_{1D} + H_2O \rightarrow 2 \times OH \tag{12}$$

($O(^1D)$ is the excited state of oxygen).

2.2.3 What shall be used for the project.

Due to its importance in the atmosphere and effect on humanity, the primary species observed in this study will be Ozone. Since tropospheric chemistry is significantly more complex than those in the stratosphere, and the health implications from ozone are greatest

here this shall be taken as the area of study. Other species may be also be considered at at a later point, once the ozone reactions have been successfully emulated.

3 Chemistry in the Model

DSMACC compatibility with the MCM allows the use of a wide selection of species within the model. In the MCM chemical reactions, Figure 8 are stored as a series of equations and rate coefficients, Figure 9. When inputted into the model these are then converted into a set of differential equations using the Kinetic Preprocessor [Sandu and Sander, 2006].

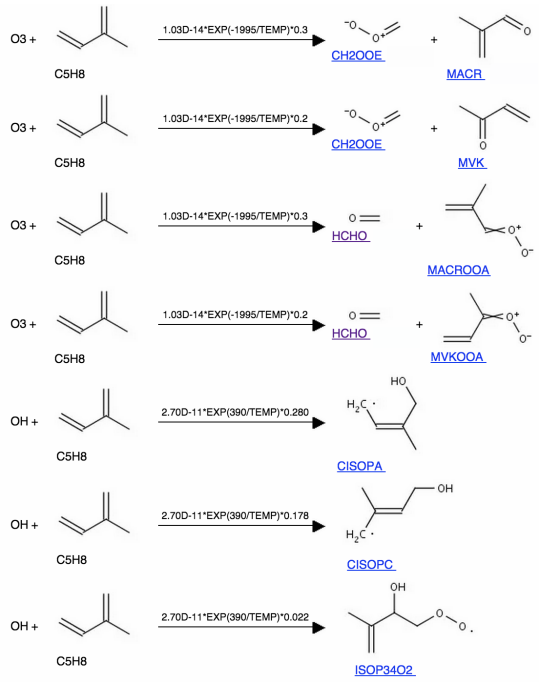


Figure 8: A visual representation of some of the reactions for isoprene from the MCM. Source: [Rickard et al., 2015b]

```
{1710.}   HMACO3 + NO3 = HOCH2CO3 + HCHO + NO2 :    KRO2NO3*1.74
{1711.}   HMACO3 = HMACO2H :   1.00D-11*RO2*0.3     ;
{1712.}   HMACO3 = HOCH2CO3 + HCHO :  1.00D-11*RO2*0.7     ;
{1713.}   HMGLYOOA = HMGLYOO :  KDEC*0.18    ;
{1714.}   HMGLYOOA = HOCH2CO3 + CO + HO2 :   KDEC*0.82     ;
{1715.}   HMACRO2 + HO2 = HMACROOH :  KRO2HO2*0.625  ;
{1716.}   HMACRO2 + NO = HMACRO + NO2 :  KRO2NO  ;
{1717.}   HMACRO2 + NO3 = HMACRO + NO2 :     KRO2NO3       ;
{1718.}   HMACRO2 = H13CO2C3 + CO + OH :     K14ISOM1      ;
{1719.}   HMACRO2 = HMACRO :   9.20D-14*RO2*0.7     ;
{1720.}   HMACRO2 = HMACROH :  9.20D-14*RO2*0.3     ;
{1721.}   OH + INCNCO2H = MACRNB + NO2 :     1.66D-12      ;
{1722.}   INCNCO3H = MACRNB + NO2 + OH :     J(41)  ;
{1723.}   OH + INCNCO3H = INCNCO3 :   4.74D-12      ;
{1724.}   INCNPAN = INCNCO3 + NO2 :   KBPAN  ;
{1725.}   OH + INCNPAN = MACRNB + NO2 + NO3 :  1.14D-12    ;
{1726.}   MMALANHYO2 + HO2 = MMALNHYOOH :    KRO2HO2*0.706  ;
{1727.}   MMALANHYO2 + NO = MMALANHYO + NO2 :  KRO2NO  ;
{1728.}   MMALANHYO2 + NO3 = MMALANHYO + NO2 :  KRO2NO3    ;
{1729.}   MMALANHYO2 = MMALANHYO :    9.20D-14*RO2*0.70    ;
{1730.}   MMALANHYO2 = MMALNHY2OH :   9.20D-14*RO2*0.30    ;
{1731.}   C23O3CCHO + NO3 = C23O3CCO3 + HNO3 :  KNO3AL*5.5  ;
{1732.}   C23O3CCHO + OH = C23O3CCO3 :  2.15D-11    ;
{1733.}   C23O3CCHO = CO + MCOCOMOXO2 + HO2 :  J(15)  ;
{1734.}   CH3COPAN + OH = HCHO + CO + CO + NO2 :    1.02D-13
{1735.}   CH3COPAN = CH3COCO3 + NO2 :  KBPAN  ;
{1736.}   COHM2CO3 + HO2 = COHM2CO2H + O3 :  KAPHO2*0.15   ;
{1737.}   COHM2CO3 + HO2 = COHM2CO3H :  KAPHO2*0.41  ;
{1738.}   COHM2CO3 + HO2 = GLYOX + HO2 + OH :  KAPHO2*0.44  ;
{1739.}   COHM2CO3 + NO = GLYOX + HO2 + NO2 :  KAPNO  ;
{1740.}   COHM2CO3 + NO2 = COHM2PAN :  KFPAN  ;
{1741.}   COHM2CO3 + NO3 = GLYOX + HO2 + NO2 :  KRO2NO3*1.74  ;
```

Figure 9: A selection of the reactions present in the *organic.kpp* file downloaded from the MCM.

3.1 Effects of different MCM schemes on the model.

Numerically stiff systems are inherently computationally inefficient. Every time we wish to progress the model a single step through time, all relevant reactions need to be computed. It is therefore expected that increasing model complexity (i.e. more species, and thus more reactions) will provide an exponential gain in computation time.

Initially a Methane scheme (8 species, 71 reactions), shall be used for the simulation. Once a working emulator is established, more complex species can be applied.

As more reactive shemes contain a larger number of dependant species, these also contain a greater number of reactions in need of computing, Figure 10. This means computational time increases exponentially with specie number. For example it can be seen how the complete MCM (5812 species, 17194 reactions) can provide a significantly larger computational burden than the much simpler methane model.

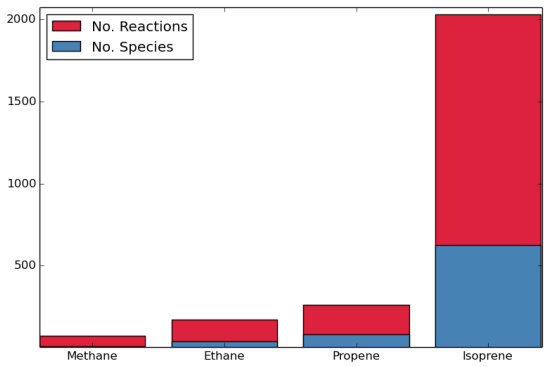


Figure 10: A bar plot comparing the number reactions with species for different MCM schemes.

Specie Name	Min. Range	Max. Range
O3	10^{-9}	10^{-6}
NO	10^{-12}	10^{-9}
H2O	10^{-6}	10^{-2}
CH4	10^{-7}	10^{-5}
HCHO	10^{-12}	10^{-9}
H2O2	10^{-12}	10^{-9}
CH3OOH	10^{-12}	10^{-9}
C2H4	10^{-12}	10^{-9}
C2H6	10^{-10}	10^{-8}
C3H8	10^{-11}	10^{-9}
C5H8	10^{-13}	10^{-9}

Table 1: A table of approximate specie ranges in the atmosphere rounded to the nearest order of magnitude, [Evans, 2014]

4 Unit Testing.

Experimental work requires the testing and calibration of apparatus. This allows for a high level of science and replicable results. Similarly it is of equal importance to benchmark computational programs against previous simulations and physical results, as will be the case with DSMACC, DiceKriging and DiceOptim.

4.1 The Dynamically Simple Model of Atmospheric Chemical Complexity

As Ozone production requires sunlight, it is important to check DSMACC can model photon interaction. Next the effects of total NOx shall be explored, concluding with the addition of a simple VOC to check for net ozone production. For this Methane shall be used as it is present in background air and contributes to ozone production through both itself and CO (which is formed during its oxidation) [Platt and Stutz, 2008].

4.1.1 Initial Conditions (ICs)

As location and pressure (elevation) are required for the box model, standard atmospheric pressure [BIPM, 2015] and York (UK) were selected:

- *1013.25 hPa*
- *53.9583 °N*
- *1.0803 °W*

Global averages for the concentrations are used when possible. For later models, the simulation ranges in Table 1 as these are thought to be representative of the real world system. Although total NOx shall be constrained for the simulation, NO, NO2 and teh VOCs willl be allowed to vary. Default initial conditions can be found in the *get.value()* function. This uses the python program *makeics_shell.py* to correctly format the initial conditions file for the DSMACC box model.

4.1.2 Sunlight hours and ozone production

Since ozone formation only occurs during daytime, it is possible to test the model's ability to replicate the diurnal cycle. The initial condition, JDAY=0, start the model at 00:00h, January 1st. Sunrise and sunset can be calculated from a 48 hour simulation (Figure 11), and are found to be at 08:30 and 16:30 respectively. As predicted by the Leighton relationship (8), ozone production is directly related to the solar intensity. This means that it peaks at noon, with ozone titrated out of the system by its reaction with NO (7) during night.

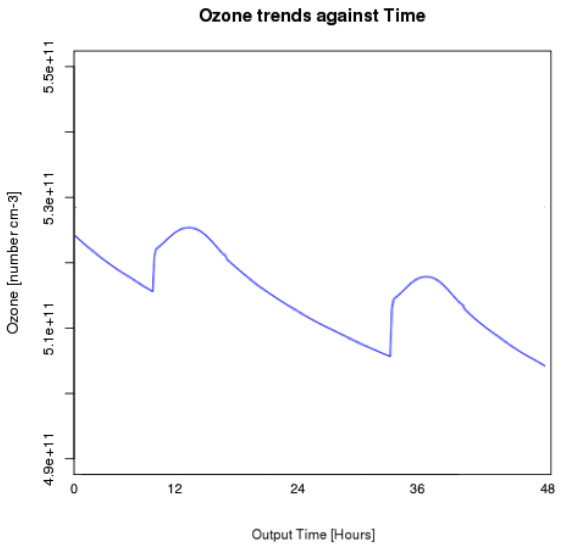


Figure 11: Ozone output over a two day period for a simulation with no methane.

Since seasonal changes affect the solar zenith angle (how high the sun is in the sky), this will also affect the quantity and length of ozone production. Dates in Table 2 are selected as to take a uniform sample at different points in the year. Figure shows the summer month of June to have the greatest period of ozone production (duration and concentration), and December to have the smallest. As would be expected the Spring and Autumn months are similar and fall between these two values.

Date	JDAY
19 March	79
21 June (longest day)	173
20 October	294
22 December (shortest day)	357

Table 2: Dates spanning the space of 1 year and their corresponding Julian Days.

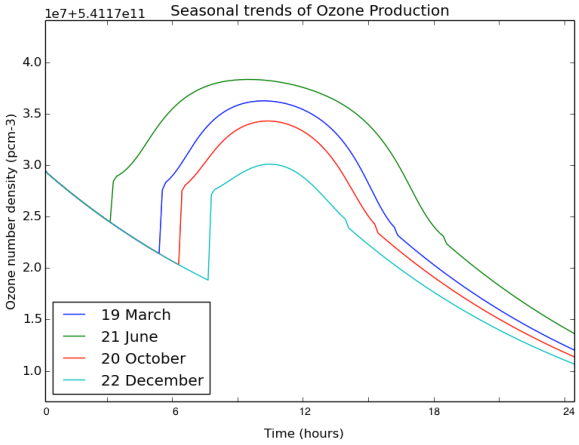


Figure 12: 24 hours from a simulation showing the difference in seasonal ozone production based on available sunlight.

4.1.3 System Evolution with Time

It is important to see how the model evolves with time. To do this initially a simulation without any VOCs over a 50 day period shall be computed, Figure 13. Using the Leighton relationship, depending on initial concentration, ozone is expected to either rise or fall until and equilibrium is reached with NOx. The ratio of ozone against NOx will determine the rate at which this happens, with some initial concentrations pushing the system into steady state quicker.

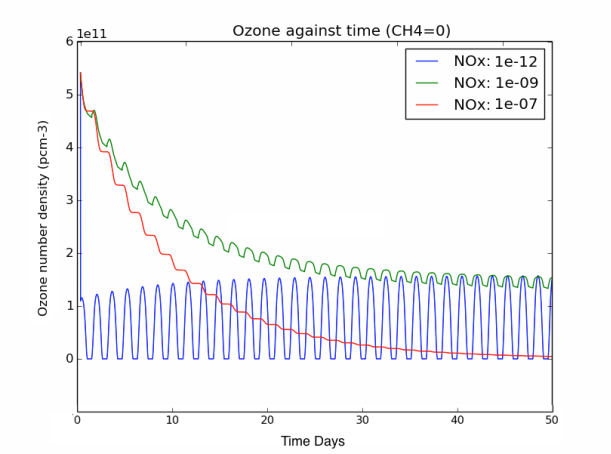


Figure 13: Ozone production over a period of 50 days with no initial methane.

Addition of VOCs to the system are expected to induce a net ozone production. In Figure 14 methane is introduced to the system. As the ozone production process is cyclic, it will once again reach steady state. However since a net-production was experienced, this is higher than in Figure 13.

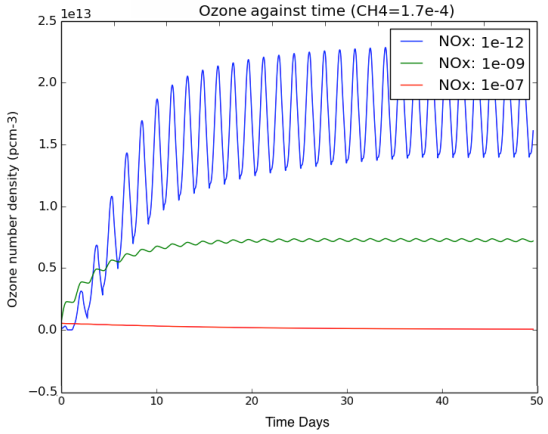


Figure 14: Ozone production over a period of 50 days with an intial methane concentration of 1.7×10^{-4}

If VOCs are not kept constant, they will soon decay into other species, reducing the overall ozone production. This is seen in Propene,C_3H_6, (lifetime ~day Figure 16, where steady state is achieved at a level lower than the initial maximum, Figure 15. If at a high enough initial concentration, or constrained, the ozone production will continue to increase, Figure 14.

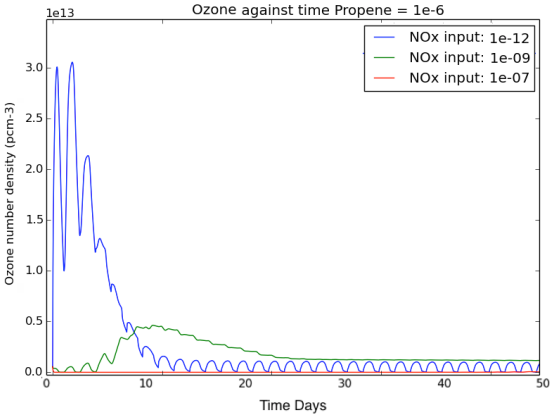


Figure 15: Ozone mixing ratio against timestep output for a model with Propene as the only VOC.

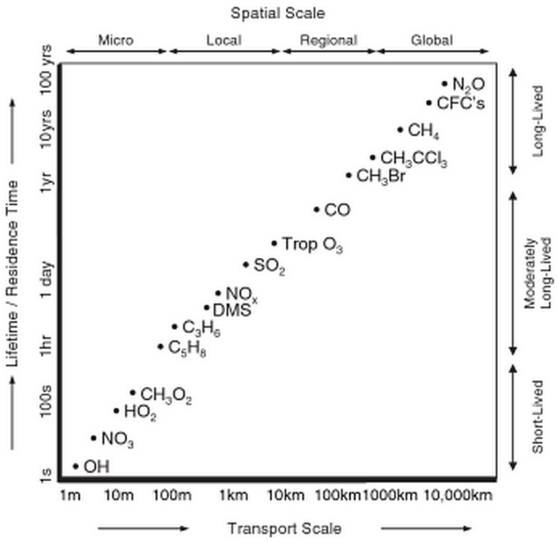


Figure 16: Species lifetime against spatial transport. Source: [Platt and Stutz, 2008]

4.1.4 NOx

Finally the evolution of NOx species with time can be observed. NOx plays a critical part in ozone production, and the generation of accurate results rely on this being correct. During daytime, NO is created. NO_3 is also formed from the reaction of NO_2 and ozone, however due to its short lifetime with respect to photolysis , Figure 16, it rapidity photolises back into O_3, (13).

$$\begin{aligned} NO_2 + O_3 &\rightarrow NO_3 + O_2 \\ NO_3 + h\nu &\rightarrow NO_2 + O \\ O + O_2 + M &\rightarrow O_3 + M \end{aligned} \tag{13}$$

At night, with the absence of solar photons, an equilibrium between NO_3 and N_2O_5 is formed. This further reduces the NO_2.

$$NO_3 + NO_2 \leftrightharpoons N_2O_5 \tag{14}$$

In Figure 17 it can be seen that during the day NO_2 is converted into in NO, and at night into N_2O_5 and NO_3. Upon sunrise all the NO_3 is suddenly photolised into NO_2. As this is also coupled to N_2O_5 a near

vertical reduction of both these species can be seen.

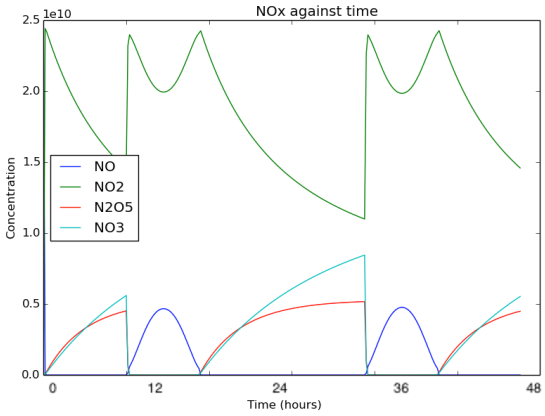


Figure 17: Changes in the constituent parts of the NOx cycle over a 48 hour period

4.2 Testing the GP Emulator (DiceKriging)

In addition to the DSMACC simulator, it is important to unit test the emulator and ensure that it provides results in agreement with other well known cases. [Fang et al., 2005] suggests a simple 1 dimensional example as an introductory test. Here a functioning kriging predictor should be capable of closely simulating a sine curve through a handful of training points, Figure 18.

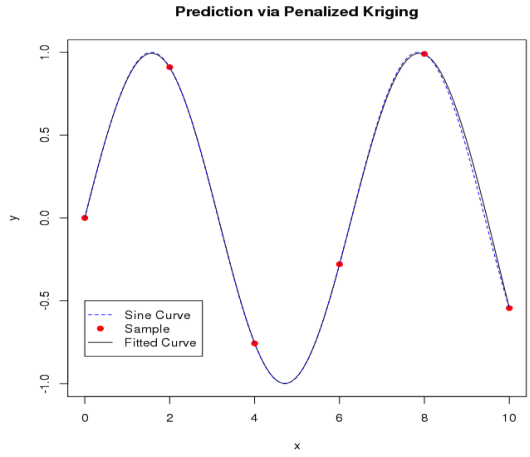


Figure 18: An example plot for the example presented in [Fang et al., 2005]

4.2.1 Effects of the Covariance Kernel

In Gaussian processes any assumptions about the predicted function are encoded in the covariance function, V_0. This consists of a correlation function $C_\delta(\mathbf{x}, \mathbf{x'})$ and an unknown scalar parameter σ^2 [Bastos, 2010]. The uncertainty between simulator and design points ($\mathbf{x}$, $\mathbf{x'}$) can be represented by a family of correlation functions (kernels).

Since ensuring positive definiteness for $C\delta$ is not a trivial task, the Gaussian, Matern and Exponential kernels (Table 3) are included in the DiceKriging package [Roustant et al., 2012].

Gaussian:	$g(h) = \exp\left(-\frac{h^2}{2\theta^2}\right)$.
Matérn $\nu = 5/2$:	$g(h) = \left(1 + \frac{\sqrt{5}\lvert h\rvert}{\theta} + \frac{5h^2}{3\theta^2}\right)\exp\left(-\frac{\sqrt{5}\lvert h\rvert}{\theta}\right)$.
Matérn $\nu = 3/2$:	$g(h) = \left(1 + \frac{\sqrt{3}\lvert h\rvert}{\theta}\right)\exp\left(-\frac{\sqrt{3}\lvert h\rvert}{\theta}\right)$.
Exponential:	$g(h) = \exp\left(-\frac{\lvert h\rvert}{\theta}\right)$.
Power-Exponential:	$g(h) = \exp\left(-\left(\frac{\lvert h\rvert}{\theta}\right)^p\right)$.

Table 3: Covariance Kernels implemented in DiceKriging. Souce: [Roustant et al., 2012]

Kernels determine the prediction of untested space between consecutive sample points. A series of unconstrained simulations with each kernel was applied to five training points in Figure 19. It is seen that each kernel provides a different level of 'smoothness' for the associated random process.

Sample paths for the centred Gaussian process using the Gaussian kernel have derivatives of all orders, and can be seen to be almost analytically smooth. The Matern kernel with parameter ν is mean-square differentiable at order k *iff* $\nu > k$ (eg. matern5_2 is twice as differentiable as matern3_2). In the case of $\nu = \frac{1}{2}$ continuous covariance equivalent to the exponential kernel. More information on the characteristics of each kernel can be found in [Rasmussen and Williams, 2006].

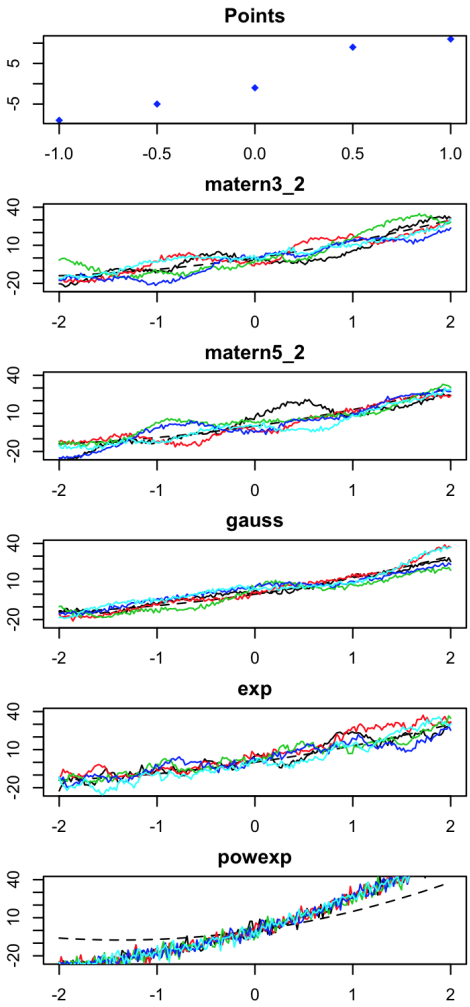


Figure 19: Unconditional simulations for each available kernel.

The simulated kernel charactaristics can also be unit tested against an interactive version of DiceKriging, Figure 20. Here the function $1.7 - \frac{\sin(12x)}{2(1+x)} + 2x^5 \cos(7x)$ is used with the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS). Again the Gaussian kernel appears to have the greatest smoothing effect, and based on its inherent 'smoothness' it shall be used for the rest of the paper.

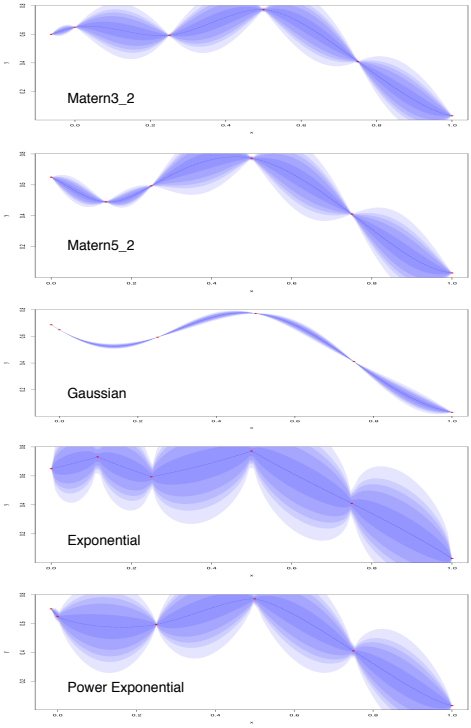


Figure 20: Changing the Kernels of an example function, $y = 1.7 - \frac{\sin(12x)}{2(1+x)} + 2x^5 \cos(7x)$. The different shades of purple represent different percentage uncertainty between input points (red). Source: [Richet, 2015]

4.2.2 Extending the Kriging estimator to 2D

Parallel optimisation of expensive-to-evaluate deterministic functions is possible using the DiceOptim package. This builds on the DiceKriging package and allows for some additional evaluation functions. A usual case study in global optimisation, 2D-normalised Branin-Hoo function [Picheny et al., 2013] shall be explored using DiceOptim.

From [Roustant et al., 2012], Figure 21 was created. This shows a slight smoothing of the actual function for the emulated case, although all other characteristics are preserved. It is also possible to compare the emulator outcome with that of the published example, Figure 22, in [Roustant et al., 2012]. Or copy of the program is seen to provide the expected result.

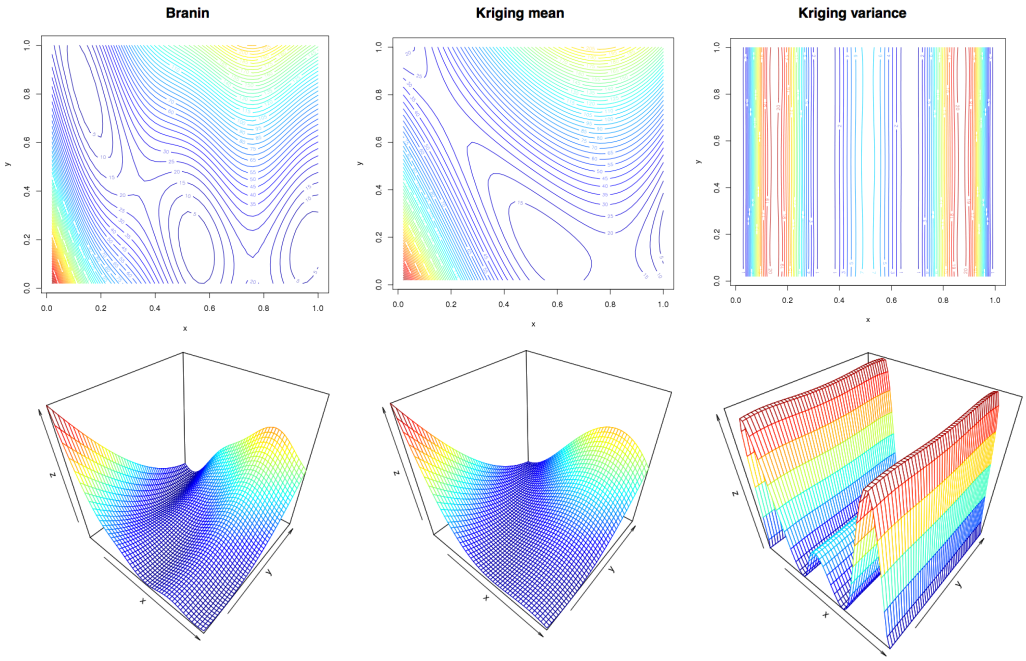


Figure 21: Simulation of the Branin-hoo function. From the left: The analytical solution of the Branin function, Kriging mean (generated from a 16 point uniform distribution) and the variance for the Kriging model.

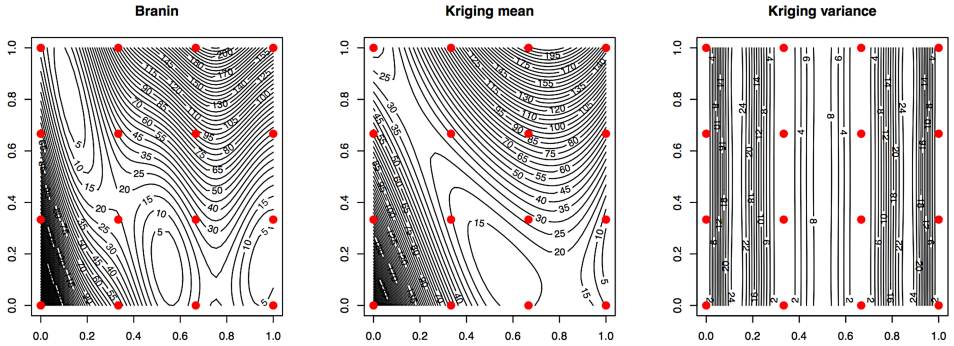


Figure 22: Results for the above from source: [Roustant et al., 2012]

5 Applying GPEs to a complex chemical system (1D)

Having reproduced a series of examples using DiceKriging and the DSMACC model, it is now possible to combine the two.To reduce computation time in the testing phases of the emulator, the reaction coefficient of methane was increased to one similar to isoprene, (A.2.3). This serves to increase the speed of the chemistry.

5.1 Comparison with published data

[Madronich, 2014] provides a figure of the ozone production curve, Figure 23. From this it can be seen that at low concentrations of NOx, ozone production is positively correlated with VOCs. With the increase of NOx, the system becomes increasingly VOC sensitive. In its abundance, the NOx reacts with oxone producing a net-loss on the RHS. It is this dynamic and challenging nature chemistry of the atmosphere hard to predict.

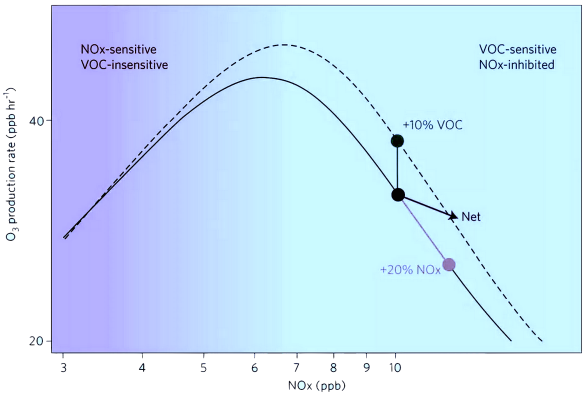


Figure 23: NOx-VOC sensitivity for a graph of NOx vs Ozone production. Source: [Madronich, 2014]

5.2 The Ozone Curve

In generating an ozone curve, the model output needs to be taken for a range of NOx concentrations after a certain length of time. Since the intended range covers several orders of magnitude, normal generator methods are not applicable. A function allocating a series of inputs for each magnitude was written, Figure 24, and was used to generate Figure 25.

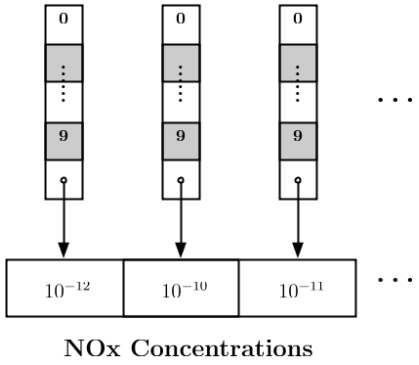


Figure 24: A visual representation of the sequence production process.

In reference to 23, it is seen that at low NOx concentrations ozone remains at its input concentration (21.981 *ppb*), and experiences a net ozone gain within the VOC sensitive region. After this, high concentrations of NOx react with O_3 removing it from the system.

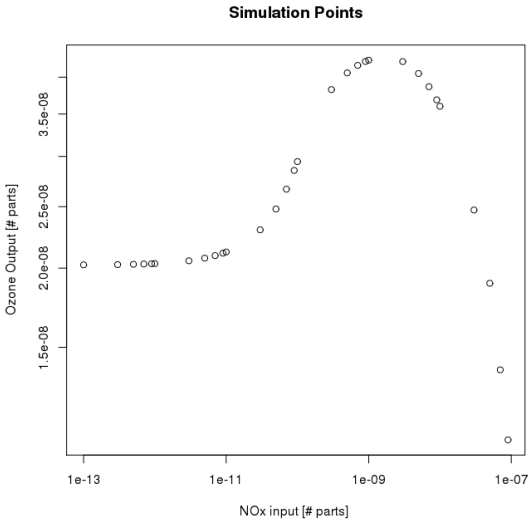


Figure 25: Ozone vs NOx graph for a series of sequential points

5.3 Kriging Trend

To create a working GPE we require two things: a probability distribution to predict untied locations be-

tween input locations (the kernel), and information on the general shape of the data (the trend). Without a kernel, the GPE will lose its probabilistic properties and effectively act as a linear regression. If a kernel is specified, but no trend, the weighting properties will create a bad fit, Figure 28(**a**). Figures 28(**a-f**), show the effect of increasing the order of the polynomial used in predicting the trend. From these results it can be seen that an order two polynomial appears to best replicate the model non-linearity.

5.4 Improving the Model

As can be seen in Figure 28, even with the correct trend, the kriging mean does not appear to provide a good fit for the model. Initially it was noticed that uniform sampling does not necessarily provide a continuous mapping of the inputs. For this reason Latin sampling was used to generate model inputs.

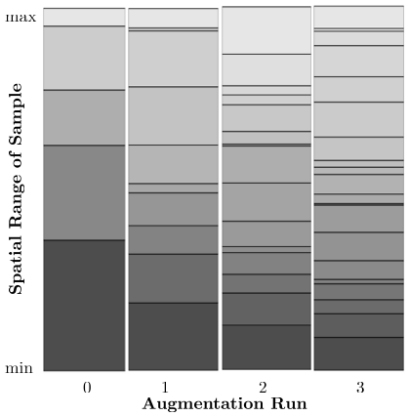


Figure 26: The augmention of a 1D Latin Sample. Run 0 is the original 5 point latin sample. 5 extra points are tehn added each consequitive run using the augmentLHS function. Vertical bands represent the spatial difference between neigbouring points as a total percentage.

5.4.1 Latin Sampling

[Bohling, 2005] states that uniformly spaced data will yield results similar to generic interpolation algorithms, and that completely random samples run the risk of leaving regions spatially uncovered [Bastos, 2010]. The suggested solution to this comes through the use of Latin hyper samples [McKay et al., 2000]. These provide complete spatial coverage with minimal sampling points.

The *lhs* package can be used to generate a Random Latin Sample [Carnell, 2013]. Additional points may be augmented by dividing the original design into (*initial point + new point*) intervals, whilst keeping the original design [Carnell, 2007]. Figure 26 shows an initial 5 point lhs design, with the sequential augmentation of a further 5 points each run.

Although the input function from earlier may be updated to use Latin samples, Figure 27, it was found that the use of logarithmic inputs removed the need for this.

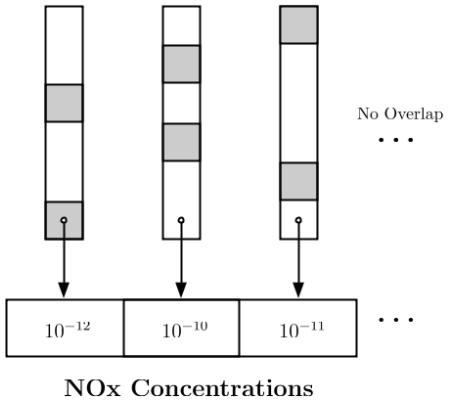


Figure 27: Updating the uniform samples from Figure 24 to those of a random LHS scheme.

5.4.2 The Logarithmic Scale

Since both sequential and Latin sample generators struggle with the scale of the inputs, It was considered that the kriging emulator may have a similar problem. To solve for this a $\log_{10}$ scale was used for the data. This was chosen over ln as it preserves any non-linearities in the data and can easily be rescaled back to its original size. A combination between logarithmic inputs and Latin Sampling shows a significant improvement in the predictive capability of the emulator, Figure 31.

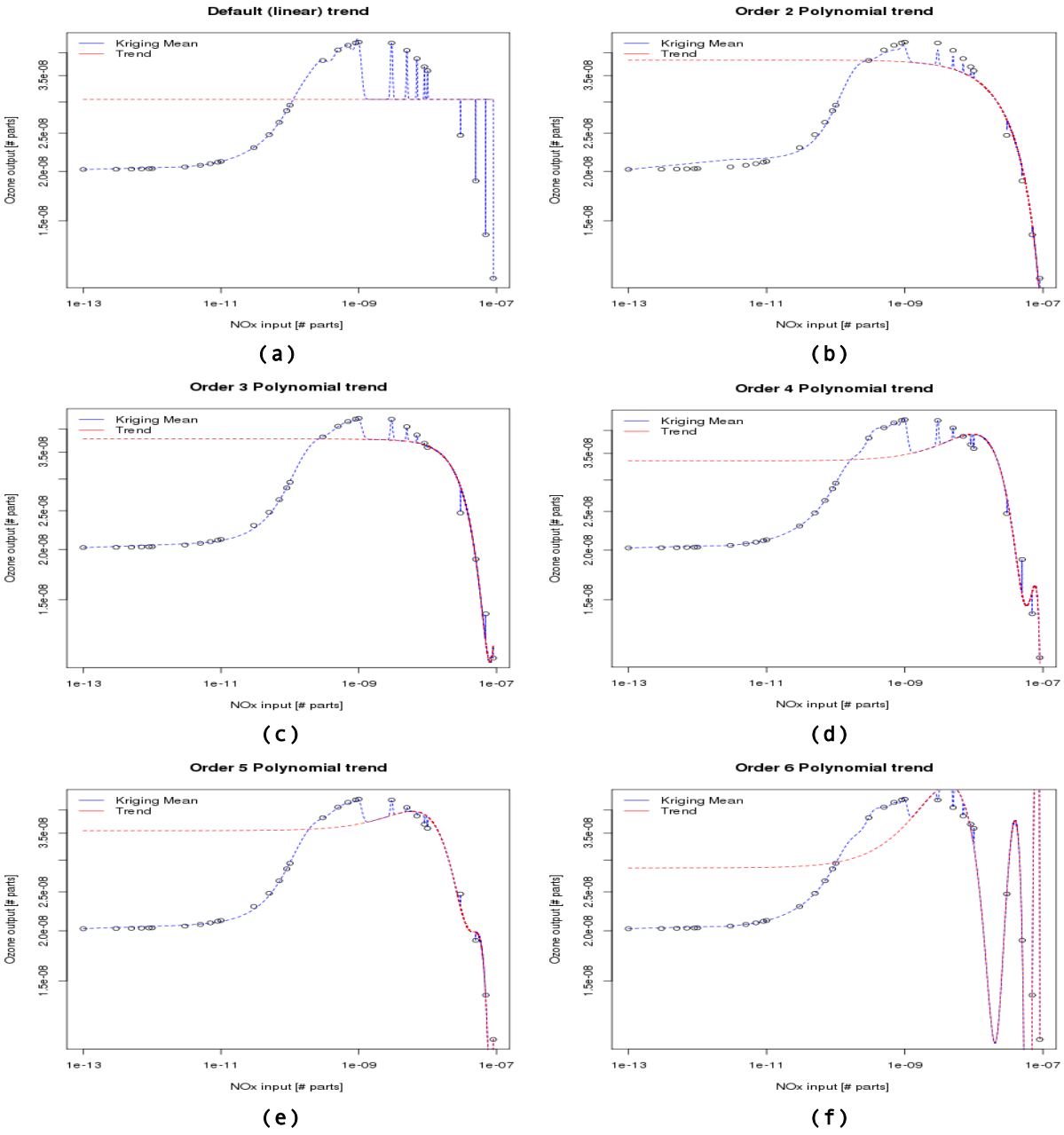


Figure 28: Exploring polynomial regression trends and their effect on the Kriging Mean. Plots a-f represent polynomial fits of order 1-6 respectively.

5.4.3 Height Adjustment

It was previously noted that differences in ozone concentration are not reflective of the spacing between NOx input concentrations. In an attempt to provide a continuous ozone sample a height algorithm was developed. The idea behind this is that if two consecutive points have an absolute difference in height over a predefined threshold, using bisection a new training point can be generated between the two.

Unfortunately it was found that although for smaller samples, an improvement may be seen, the weiging nature or the GPE mean that any groupings are likely to be approximated as a single point and may skew the predictor: Figure 29.

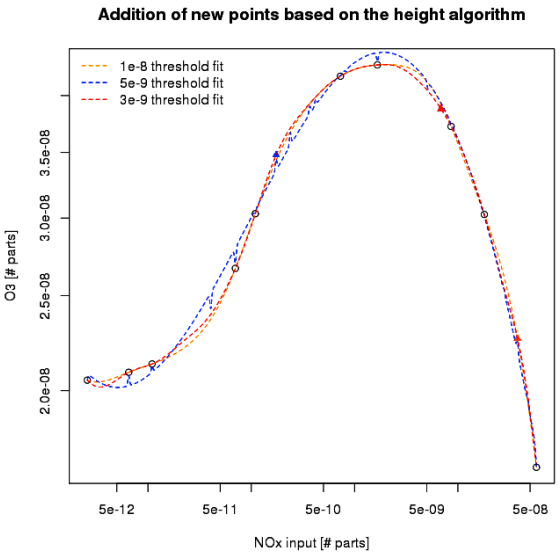


Figure 29: A plot comparing three kriging simulations based on the height thresholds of 10ppb, 5ppb and 3ppb. Here additional points are shown in triangle form, where each triangle colour corresponds to that in the legend.

5.5 Increasing the number of Training Points.

As efficiency in prediction is required, a small number of inputs will be beneficial. However, too few points will provide a poor representation of the chemistry, and in turn an inadequate emulator.
[Chapman et al., 1994] recommends that the minimum number of points used for an emulator are atleast ten times the dimension of the emulator. Since even spatial coverage is desired, the number of points for an emulator shall be determined by the number number of magnitudes multiplied by an integer (1,..,n). This integer will then be increased until an accurate fit is achieved.

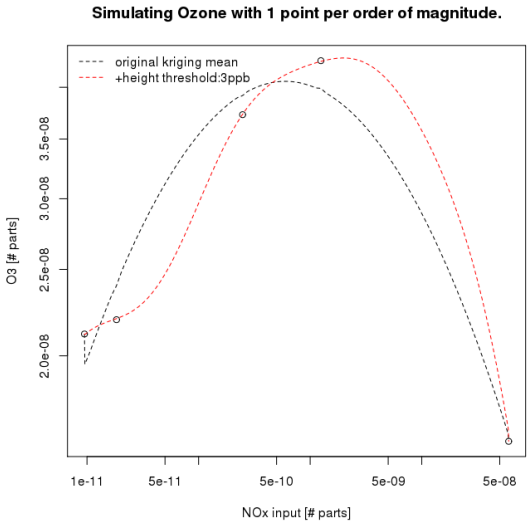


Figure 30: Kriging mean and height adjustment algorithm (threshold: 3ppb) for 1 point per order of magnitude.

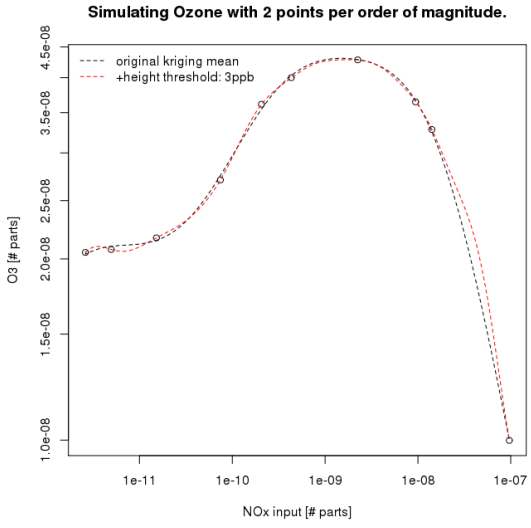


Figure 31: Kriging mean and height adjustment algorithm (threshold: 3ppb) for 2 points per order of magnitude.

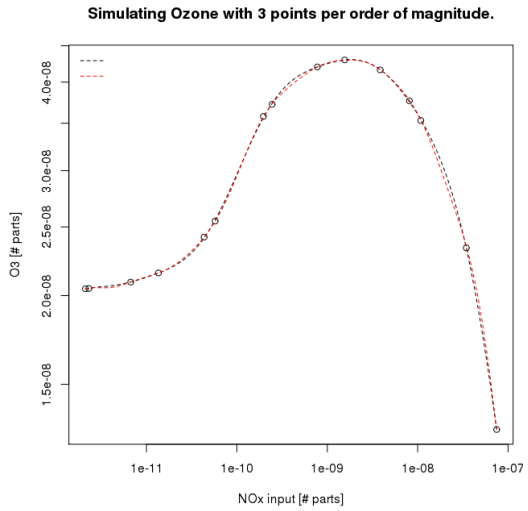


Figure 32: Kriging mean and height adjustment algorithm (threshold: 3ppb) for 3 points per order of magnitude.

Figures (**30 -32**) show an improvement of prediction with the number of points used. Any more than three points, only a marginal improvement is seen, Figure 33 and therefore, a 3 point design shall be used to save on emulator generation time. Use of the logarithmic latin sampling technique has produced a accurate fit, eliminating the need for a height adjustment function (red line).

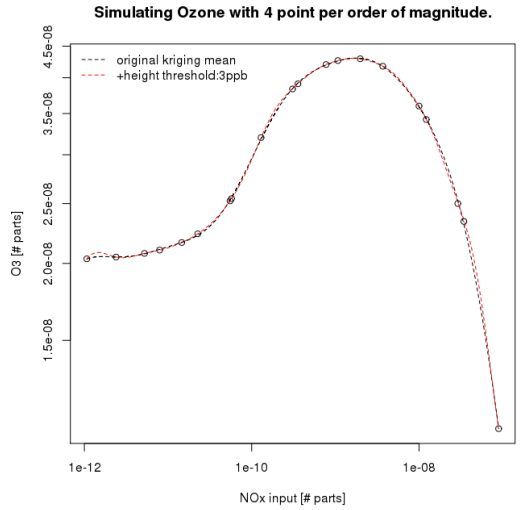


Figure 33: Kriging mean and height adjustment algorithm (threshold: 3ppb) for 4 points per order of magnitude.

6 Emulating with two inputs (2D)

In two dimensions inputs can be provided in the **x-y** plane, where outputs are visualised along the **z** axis. For the input species, NOx and methane (CH_4)[7] shall be used.

A simulation grid of all possible outputs may be computed for use in validating the emulator. A grid of medium coarseness, (100 points / magnitude^2) requires 2.5×10^3 calculations for 5×5magnitude grid. Although not affecting the emulator outcome, constant generation of these can reduce the fidelity of the training tests. For this reason the 'perfectly parallel' nature of this problem will be exploited using parallel programming.

6.1 Parallelising the code

In R contains two packages for the use of multiple cores. These are the SNOW (distributed memory) and MCcore (shared memory) packages. As with DSMACC and DiceKriging, these shall be unit tested against their serial counterparts, and then against eachother before being used in the program.

6.1.1 Result Accuracy

It was noted that certain results returned contained incorrect valies. Consistency in model results placed the problem with the parallel programs [8].

As this problem occurs in both packages, multiple simulations of each value can be computed. Should the same value be calculated three times, it can be accepted as true, Figure 34. This decreases the probability an incorrect value will be selected.

[7] Chosen as it is the simplest VOC scheme available on the MCM

[8] Most likely either memory access issues, problems with the shared directory or write errors during the reduction stage of the

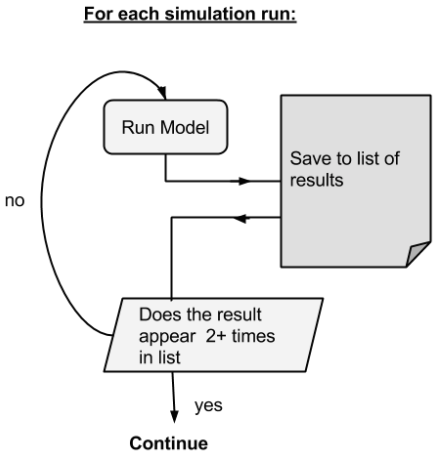


Figure 34: A flow chart represnting the repeat loop that ensures parallel results are correct.

6.1.2 Comparing both types of Parallelisation

With both parallel programs producing concurrent results, it is possible to compare their computation time, Figure 35. Here the mccores package consistently outperforms its SNOW counterpart. This is most likely because for each set of calculations SNOW needs to initialise a new computing cluster. Furthermore the single line execution of multicore, allows for a neater, and more legible, code structure, making it better suited for this project.

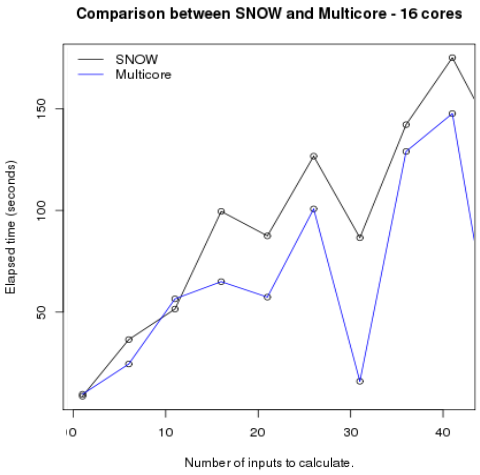


Figure 35: Computation times of the SNOW-like foreach package and the multicore mclapply package for the same inputs.

6.1.3 Serial vs Parallel

Having chosen a parallelisation program, it is now important to benchmark it against computation in serial. As GPE computations will use 16 cores, this will be used in the test. Figure 36 shows that repeating calculations to obtain a correct result serves to increase the computation time. From the results the intercept is used to impose the condition that parallelisation is only used iff the number of results calculates is 34. This helped reduce the computation time for the simulation grid to **2.05** hours from **21.18** .

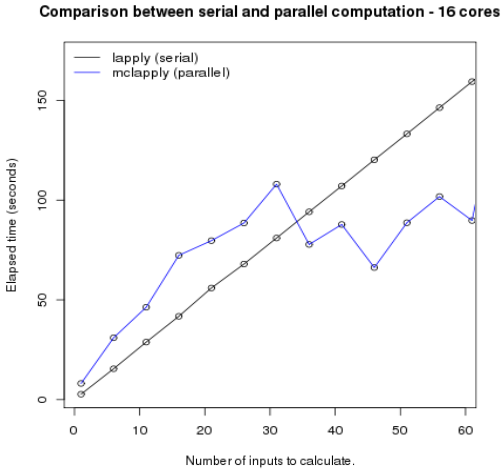


Figure 36: Comparison between a single serial run and the mclapply parallel run including the quality assurance test in Figure 34

6.2 Simulation Grid

Although it is not feasible to calculate a simulation grid for every emulation, its existence could prove invaluable in the training of the model, and verification checks. Figure 37 is an emulated grid for the 2D NOx-methane emulator. In two dimensions the complexity of the emulator becomes more evident. Her he main peak is representative of the methane reaction to produce oxygen. However since methane oxidises to form other compounds, the also exists a second stationary point producing ozone[9]. Complexity such as this are what makes atmospheric difficult so simulate.

[9]This is most likely CO.

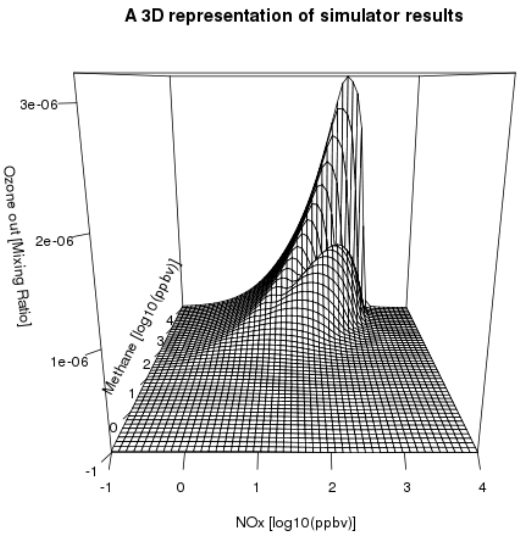


Figure 37: A 3D plot of the simulation grid generated. Here the inputs are in $\log_1 0$ particles per billion volume (ie. mixing ratio $\times 10^{-9}$). Ozone output is in a non-normalized mixing ratio as output by the model.

To form a more representative image of the system, a $\log_{10}$scale may be applied:

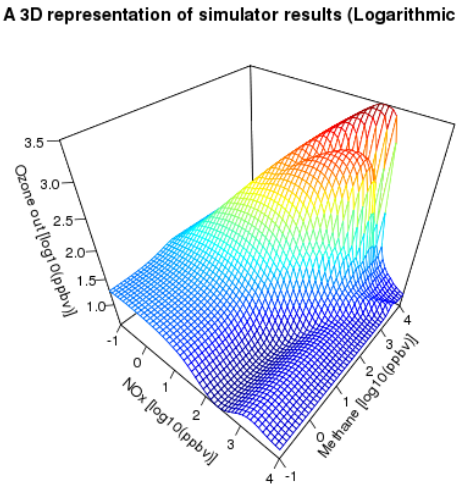


Figure 38: Figure 37 with a $\log_{10}$(ppbv) normalised mixing ratio for Ozone output.

Figure 38 can now be used to compare emulated results with reality. Using this it is possible to explore the number of training points against prediction quality for a 2D GPE.

6.3 Number of points against prediction quality

A Latin-rectangle can be used to generate a latin sample in 2D, r.h.s Figure 39. A grid (light grey[10]) outlining each square magnitude is drawn for reference when comparing point distribution. It was found that the pseudo-random nature of the latin-square selection results in uneven coverage for lower number of training points, Figure 39 (top right). As the number of points is increases, the better coverage becomes aparent, with all grid boxes being filled. This is also reflected in the emulated results (middle and left columns).

It is found that increasing the number of training points provides better spatial coverage, and in turn a more realistic emulation of the system. Following from this, the 6 point design shall be used to explore emulator accuracy.

7 Testing the quality of the emulator

So far emulator accuracy has been evaluated through the use of prior knowledge of the system. Having a achieved the expected shape representing the chemistry, the simulation grid may now be used to locate the find strengths and weaknesses of the emulator as well as devising as series of verification tests.

[10]May not be visible on some printouts, although clear on the version submitted online. For those without on-line access, these are just extensions to the x and y-ticks on the axis

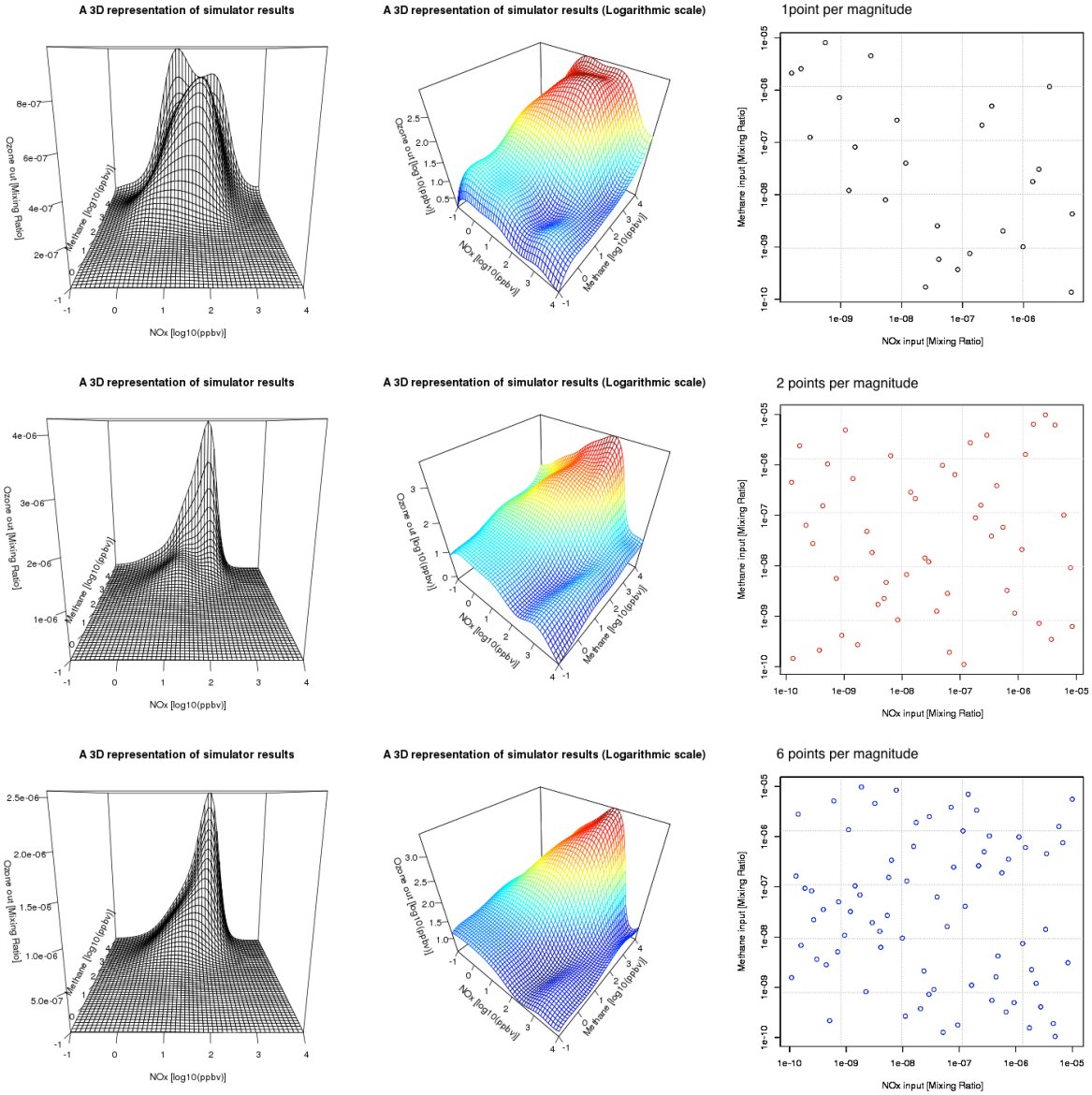


Figure 39: **By Row:** Emulations with 1, 2 and 6 points per magnitude box (shown by the dashed lines). **By Column:** Emulation output in - Ozone in Mixing Ratio output, Logarithmic-normalised Ozone Mixing Ratio and Latin-Rectangle design used for the inputs.

7.1 Comparing Contour Plots

It is possible to compare two distributions through the superimposition of one on the other. The simplest way to do this would be to overlay their isopleths[11].

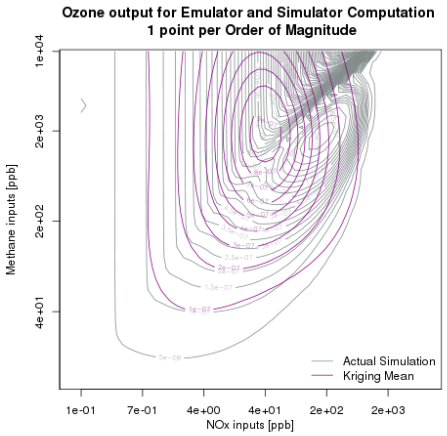
Using Figure 40, the need for greater spatial coverage (more training points) is reiterated. Addition of more points results in the stationary point of the GPE shifting between the two peaks (b). The reson for this is that the smoothing nature of GPEs averages the surrounding points and predicts a peak based on the rising gradients. To solve for this additional points need to be calculated.

Although the rapid changes of the peak have not been represented, the GPE appears to have picked up and replicated model asymmetry from the broader isopleths (c).

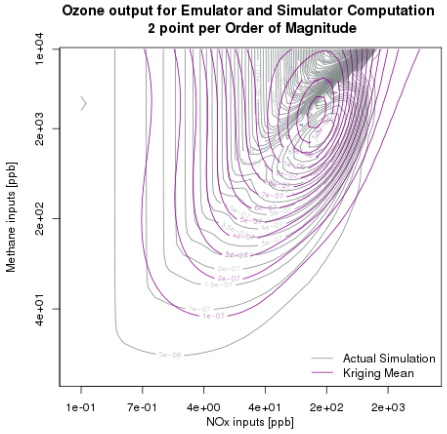
7.2 Calculating the quality of estimation

An alternative method to compare model and emulation would be to plot their difference on the same graph, Figure 41. This can be further improvement can be obtained by normalising the data and looking at percentage difference, , Figure 42. From this areas of under and over estimation can be seen. Calculating the total difference can be done through taking the modulus[12] of this, Figure 43. The stationary point can be used to assess the quality of the emulation. Should directional information about the estimates be required, a coloured contour map can be used, Figure 44. Comparing this with the Figure 40, it is seen that the emulator's weakness lies in the sharp rise of the methane peak.

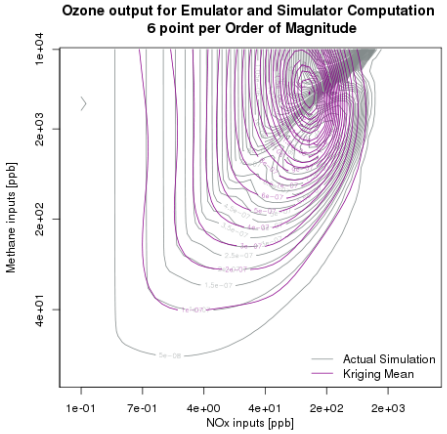
[11] lines of constant height (contour lines)
[12] absolute value



(a)



(b)



(c)

Figure 40: Simulation vs Emulation contours Comparing the latin designs used in Figure 39 against the simulation grid, Figure 38, using 1, 2 and 6 point designs. To improve clarity, the range of contours of the emulator was set to half of those of the simulator.

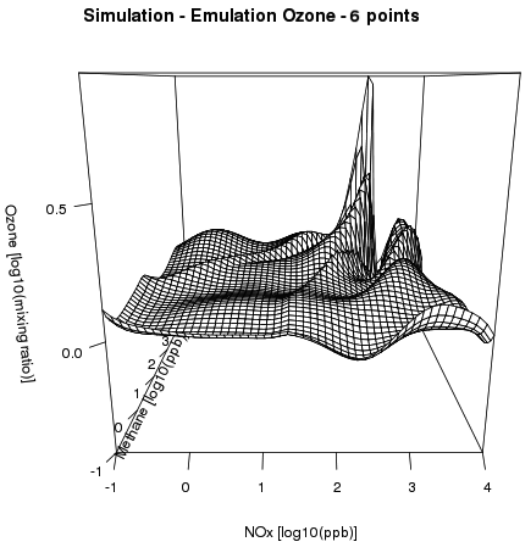


Figure 41: A plot of the logarithmic differences between the DSMACC simulator and GP Emulator. Here a value of 0.5 corresponds to half an order of magnitude.

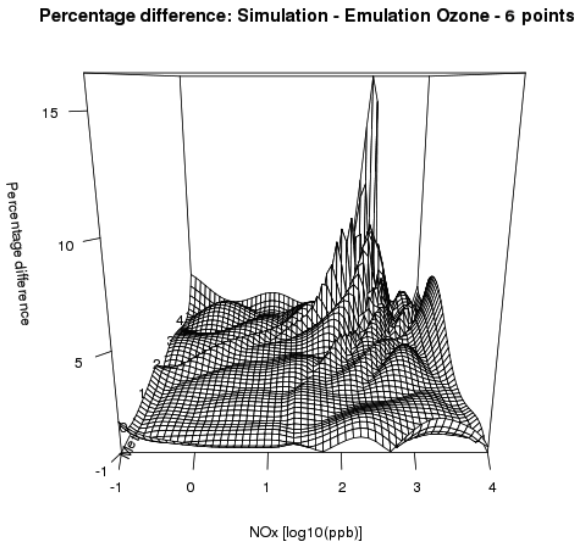


Figure 43: Absolute percentage difference of the emulator from the approved simulation value. As before the y-axis is Methane

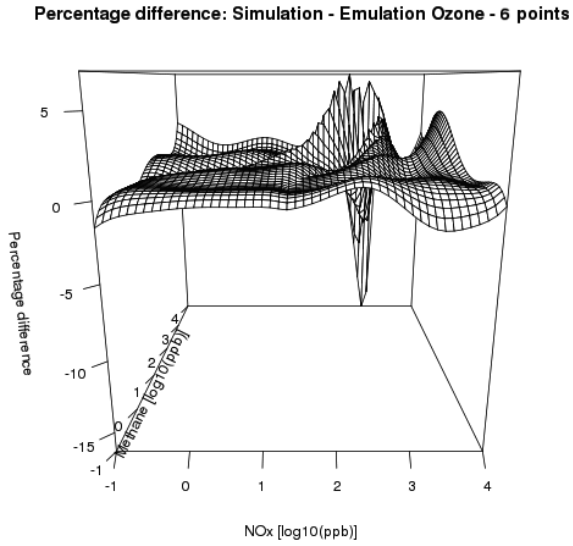


Figure 42: Percentage difference of the emulator from the approved simulation value.

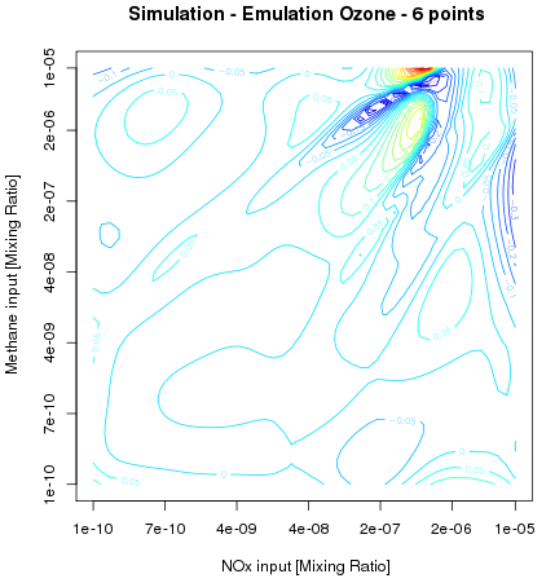


Figure 44: A colour contour map indicating a region of under-estimation (blue horseshoe top right) and over-estimation (red spot at methane input 10^{-5})

Through comparing the emulator against simulation results it was seen that the GPE is capable of repeating the general trends of the chemistry, although it tends to smooth sudden discontinuities should insufficient training points be provided. The maximum difference between model and GPE was 15% . If through the us of carefully selected training points, this can be reduced to 5%, that would make the GPE a viable candidate for replacing the simulator.

8 Improving the emulator

As simulation grid generation is not feasible for each emulator, there exist several statistical tests that can be applied to GPEs. The usefulness of these shall be determined from the computed simulation grid.

8.1 Expected Improvement Criterion

The EI criterion is the idea that sampling a new point $\mathbf{x}$ will bring an improvement of $min(y(\mathbf{X})) - y(\mathbf{x})$ if $y(\mathbf{x})$ is below the current minimum [Bastos, 2010]. Similar to the 95% confidence levels, this will be zero at simulation points since there is complete certainty within them. It is possible to simulate an additional point at the maximum EI value. This can be used to improve the model, although the drawback is that only one point can be added as the GP has to be recompiled each iteration.

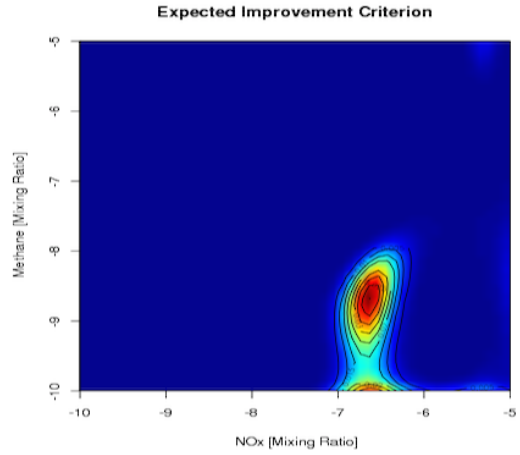


Figure 45: An image plot of a grid where the EI criterion has been applied

As the EI is influenced by the weighting nature of the GPE, it relies on this data and us unable to recognise the discernible difference between GPE and simulator. With the problem lying in the complexity of the chemistry, this provides little or no improvement. Instead validation techniques focusing on the fitted data shall be explored.

8.2 Uncertainty in prediction

As GPs work on probabilistic estimation, use of the 95% confidence levels can be used to calculate the uncertainty for each prediction. As these are symmetric around the kriging mean, the area enclosed by these can be calculated and the absolute error plotted, Figure 46.

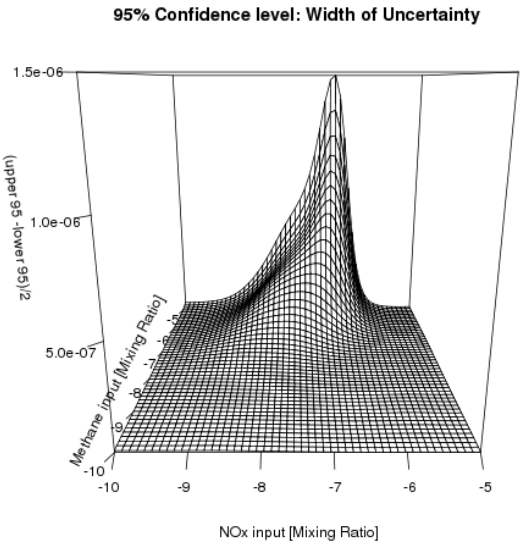


Figure 46: A plot representing the difference between upper and lower 95% confidence levels divided by two. Doing so indicates where the predictors greatest uncertainty allowing for additional training points to be calculated.

When normalised using the lower95 confidence level, Figure 47 is produced. This is identical to the kriging standard deviation [13]. Analysis of the this shows areas of greatest uncertainty at the boundaries. This is expected since these are often poorly represented,

[13] Although plotted, this was not included as it is identical

meaning that the GPE cannot approximate them correctly. To account for this future grids shall always be generated slightly larger than the required range.

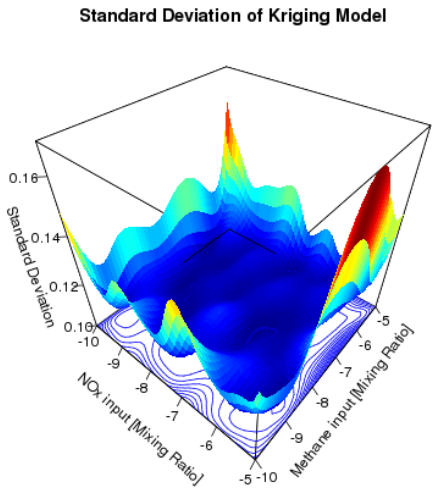


Figure 47: Figure 46 normalised around the lower 95 % confidence level. This is identical to the kriging standard deviation (see blue contour)

8.3 Slices

To explore emulator performance against itself it is also possible to decompose the 2D GPE into a series of 1D plots, comparing these to a 1D emulator. This can be used to check how an GPEs scale with dimension.

For this test slices of greatest ozone variation were taken, referencing the simulation grid. As slices of NO_x and CH_4 can be taken from the 2D GPE, the equivalent were generated using a 1D GPE. Comparing the results, Figures (**49-50**) show that on general a greater amount of smoothing is applied to the 2D emulator. Comparing both plots it is apparent the the number if input points[14] do not scale linearly, and that in order to achieve the same quality as that of the 1D plot, 3^3 training points are required.

[14] 6 per order of magnitude for 2D and 3 for 1D

9 Pre-Validation Checks

Emulator verification ensures that the design accurately captures the models trend/output. Although manual observation is a good diagnostic tool, it cannot be used for an automated process. The Leave-One-Out, and Quantile-Quantile plots are often used in statistics for evaluating Kriging models, and can be written into the verification stage of the program.

9.0.1 Testing Predictive Capability

The Leave-one-out verification is used in [Rougier et al., 2006] to validate the emulator of the HadSM3 climate model. In removing a training point and recalculating its value using the emulator, it is possible to assess emulator performance. Here the closer the points are to the $x = y$ line, the better more accurate the predictor.

Using Figure 48 it can be seen that overall most of the predicted values are similar to the exact value. Deviation from the $x = y$ line (for values $> 10^{-8}$) again confirms that the model struggles around the two stationary points.

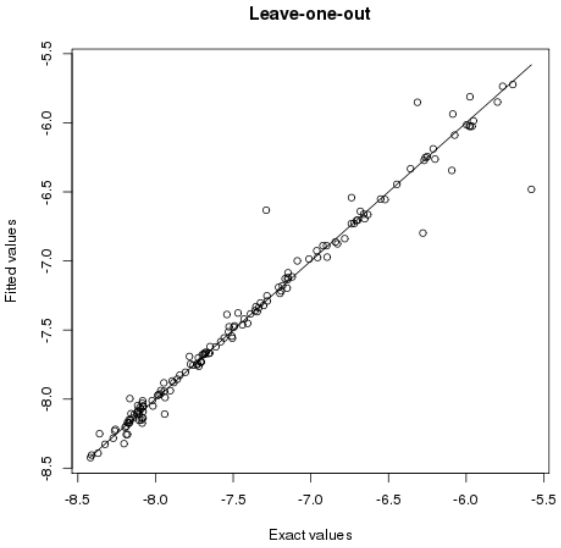
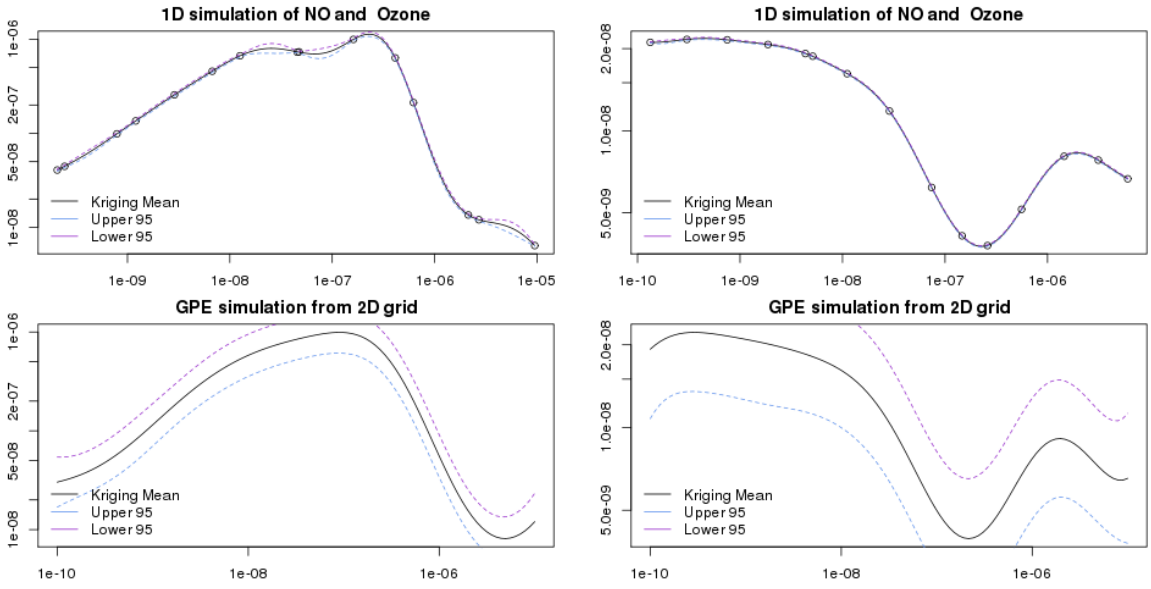
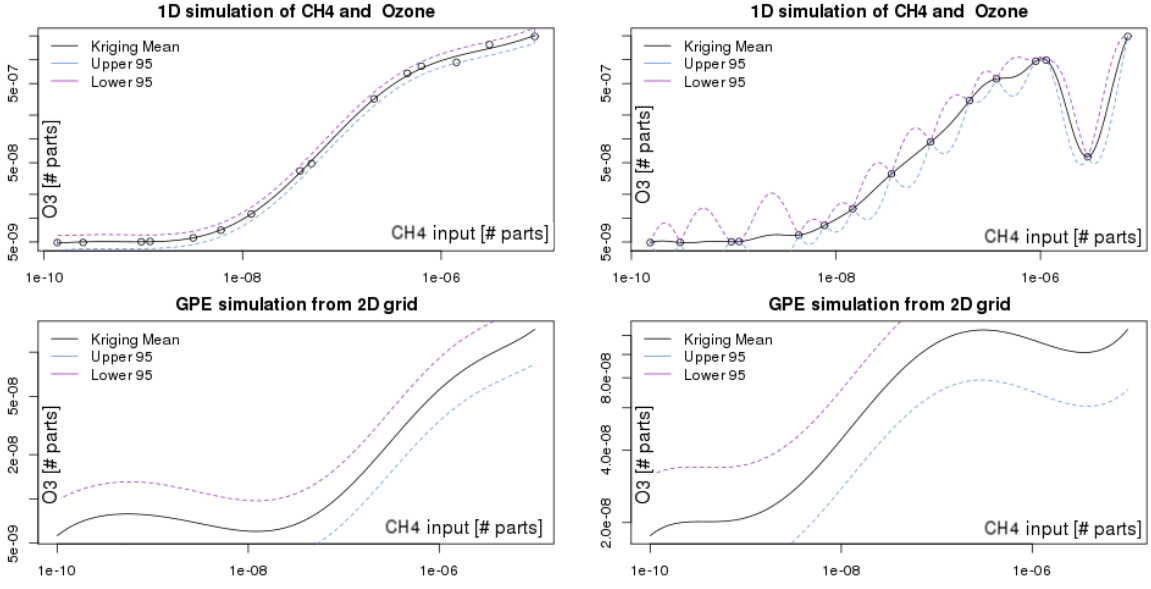


Figure 48: The leave one out verification applied to the 6 point per magnitude, 2D emulation used for the previous section.



(a) (b)

Figure 49: Comparing the 2D emulator results against those of the 1D emulator: Methane concentrations of 10^{-6} (3 in log10(ppbv)) and 10^{-9}(0 in log10(ppbv)) are used in (a) and (b) respectively.



(a) (b)

Figure 50: Comparing the 2D emulator results against those of the 1D emulator: NOx concentrations of 10^{-9} (0 in log10(ppbv)) and 10^{-} (2 in log10(ppbv)) are used in (a) and (b) respectively.

9.0.2 Standardised Quantile-Quantile plot

Qunatile juxtaposition can be used for the comparison of two probability distributions, P_1 and P_2.
[Wilk and Gnanadesikan, 1968]. Probabilities of a similar distribution will have a trend following the $x = y$ line. Calculation of the gradient and intercept can then be used for diagnostic purposes, (15), and quality control.

$$\frac{\delta P_1}{\delta P_2} = \begin{cases} < 1, & \text{predictive variability over-estimate} \\ 0, & \text{probability distributions equal} \\ +ve, & \text{predictive variability under-estimate} \end{cases} \tag{15}$$

The q-q plot in Figure 15 suggests that there exist long tails at the ends of the data distribution [15]. This indicates that the distribution involved is not a normal one, but more likely exponential. Future work may include the testing of the emulator using an exponential kernel.

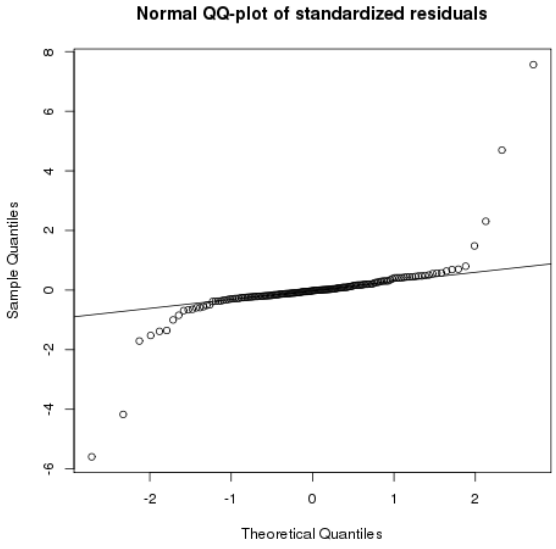


Figure 51: A quantile-quantile plot for the 6 point per magnitude, 2D emulation used for the previous section.

[15] Left end of pattern is below the line; right end of pattern is above the line

10 Validation

Once the emulator has been verified, It is important to validity its predictive capability. To do this the 95% confidence levels may be used to compare trialled data points against those predicted from the GPE. This can be done by subjecting a newly generated Latin Hyper-Sample[16] to the process in Figure 52.

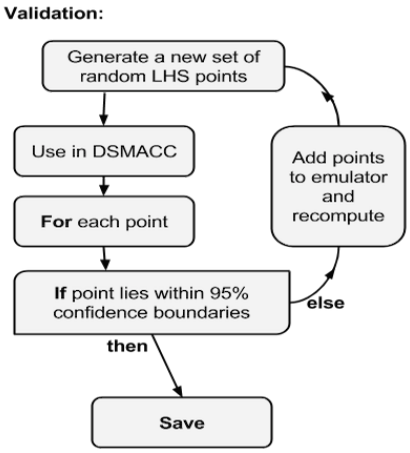


Figure 52: A flow chart of the Validation Process.

In this case, the simulation grid can be used to indicate regions failing validation, Figure 53 In cases of failed points, these are appended to the training array and the model is recalculated.

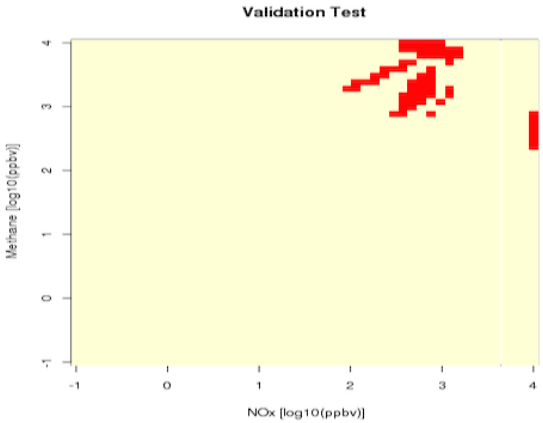


Figure 53: An image of the validation procedure. Red squares are regions not within the 95% confidence levels of the emulator.

[16] A Latin sample covers the entirety of input space

11 GPEs in Higher Dimensions

Eventhough complicated systems can be represented by an ensemble of many 2D GPEs, this is not very practical. Instead it is better to increase the dimensionality of the emulator.

11.1 The 3D GPE

When using three inputs, it is possible to break a model down into a series of 2D plots for evaluation, Figure 56. This is similar to the 1D slices. NOx and methane shall be kept in the model for consistency, and water vapour shall be added as the third parameter. This was chosen as without it the hydroxyl-ozone cycle cannot function.

Although water is seen to have a slight effect, its large quantity in the atmosphere does not cause any significant changes in the atmospheric concentrations of the Figures[17]. At concentrations of NOx and Methane the system becomes sensitive to the ratio of water to methane. Fortunately these levels do not occur naturally in the troposphere and do not need to be emulated.

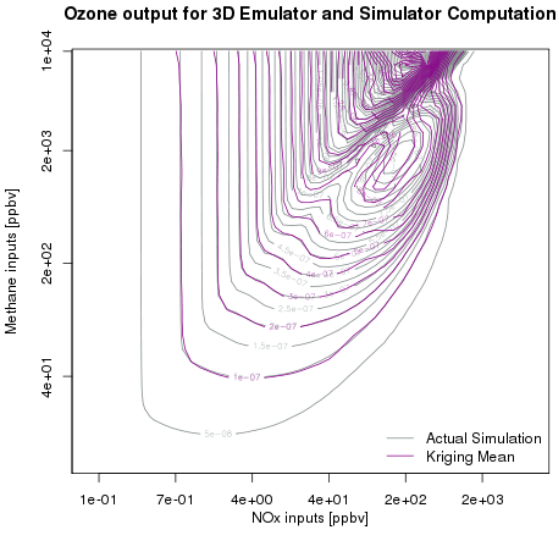


Figure 54: A contour plot of the simulation grid and 3D GPE prediction

[17] For atmospheric ranges see Table 1

11.1.1 Testing the 3D model against the simulated grid.

Since it is possible to compute a 2D simulation of NOx and methane, the 3D GPE can be subjected to the same tests as in the previous section. For the 3D emulation a set of 30 training points/magnitude was selected to replicate the accuracy provided by a 3 point 1D design. Comparing the design of Figure 54 to Figure 40, a clear indication of the isopleths and both stationary points is now present.

Comparing absolute percentage difference, Figure 55 reveals an overall improvement on the errors, although there still is a large error peak of 14.9%. To solve for this, either a larger training sample is required, or a selection of carefully placed points. The reduction in errorc can also be seen in the colour contour plot, Figure 57, which contains large area of little of no difference from the simulator.

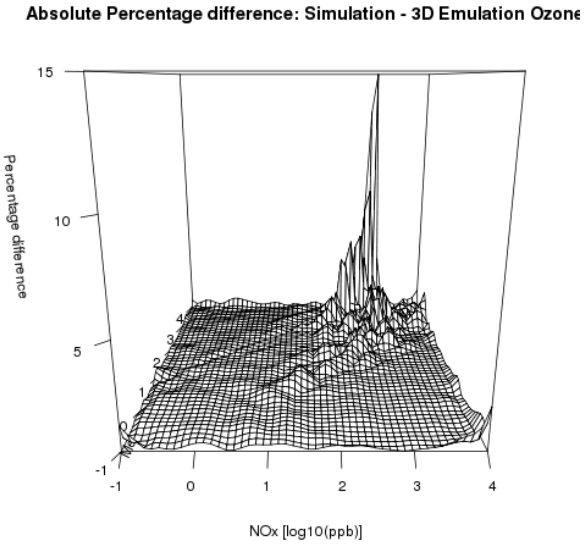


Figure 55: Absolute percentage difference of the 3D GPE against the simulation grid

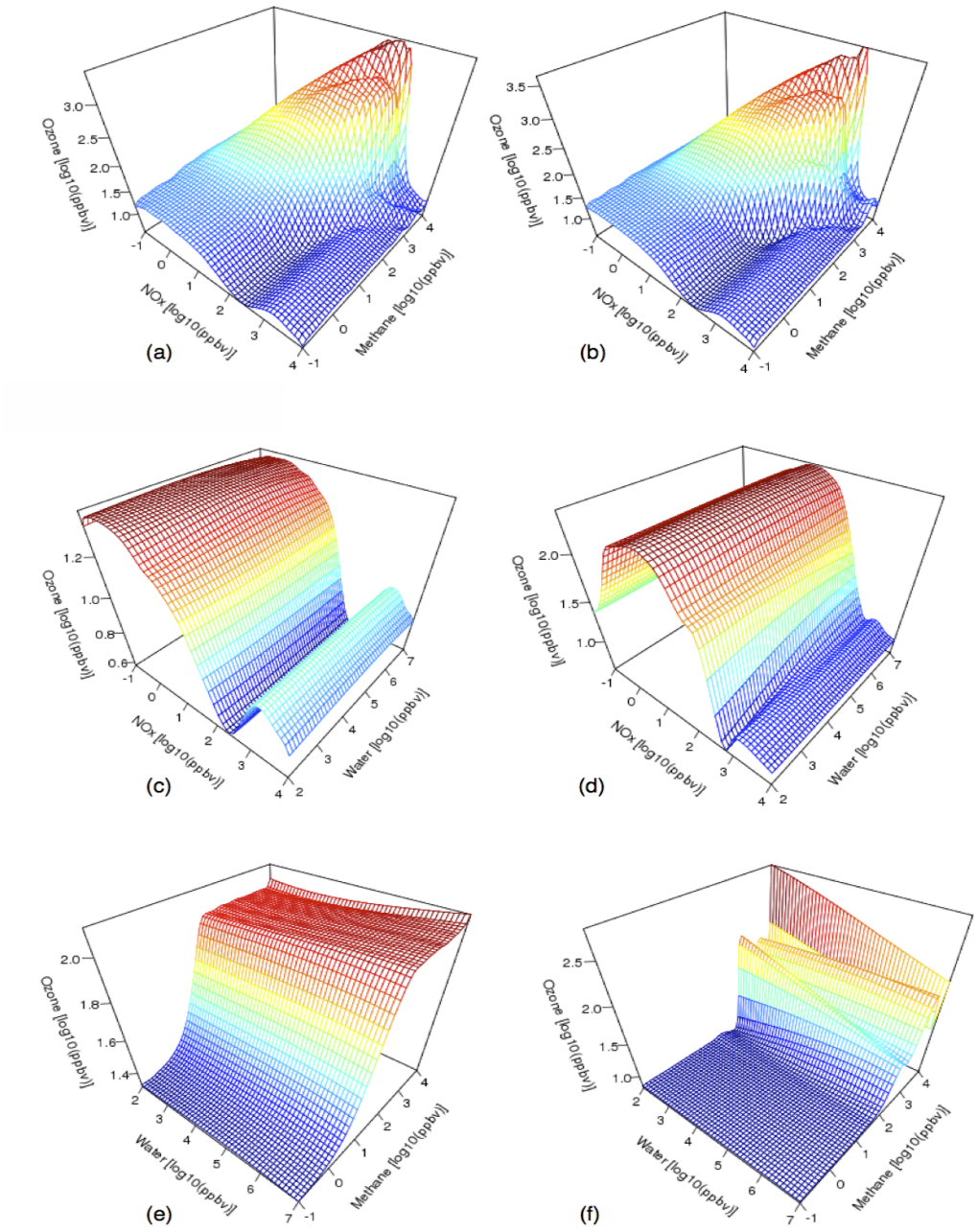


Figure 56: Decomposing the 3D emulator into a series of 2D plots. **By Row:** First NOx and methane are plotted with water concentrations of 10^{-2} and 10^{-6} (a)&(b). Next NOx against water are plotted at methane concentrarions of 10^{-9} and 10^{-6} (b)&(c). Finally water against methane is plotted at NOx concentrarions of 10^{-9} and 10^{-6} (d)&(e).

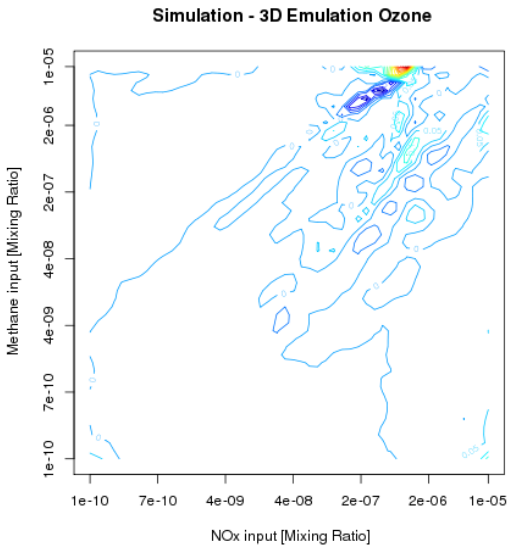


Figure 57: A coloured contour plot representing the polarity and magnitude of the percentage error between emulator and simulator grid. Cyan = An exact match, blue = an underestimate, red= overestimate

12 Other Species

Although this report concentrates on a handful of species, it is possible to emulate many others. However the primary aim is to test GPEs ability to replicate a complex chemical system, many less reactive species have not been tested. Figure 60 shows a range of other species and their effect on ozone. A 2D simulation with these against NOx is used in testing the GPEs performance.

13 Benchmarking

As the aim of the aim of the project is to find an alternative to the current, computationally expensive, simulator, it is important to compare the computation speed for the GPE against the model. Conditions of scaling with species, different sized samples of training points, dimensionality and the number of calculation points shall be explored.

13.1 Exploring computation time against number of points

For a single test point, Figure 61, computation time of the emulator is seen to be 0.2% of the simulation time. In increasing the number of points to 10, Figure 62, it rises to 0.6%, and 40 to 4%, Figure 63. As a lower percentage of the simulation time is better, it is seen that the model gets worse with the number of calculation. However even at 40 calculations, a 96% decrease in computation time is still experienced.

In regards to specie complexity, it can be seen that for the species that are selected computation time for both the simulator and emulator remain roughly constant, so this is not a problem.

13.1.1 How the number of training points affect computation time.

In the validation section it was established the GPEs with a greater number of training points will better represent the features of the chemistry. It is therefore important to see if the computation time of a GPE changes significantly with number of training points. From Figure 58, it can be seen that increasing the number of training points in a design has a negligible affect on its computation time. The result from this suggests that it is possible to further increase the quality of an emulator without affecting its performance.

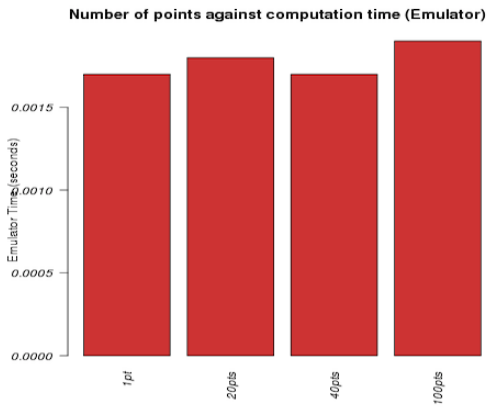


Figure 58: A bar plot showing the change in time taken to predict 10 unknown datapoints using a 2D emulator with a varying amount of training points.

13.2 Emulator dimensionality

As atmospheric chemistry is quite complex, an emulator will be required to compute many species simultaneously. This will require higher dimensions. Comparing the performance of a 1D,2D and 3D emulator, Figure 59, shows an increase in computation time. Although still significantly faster than the simulator, further work will need to be done to establish whether i) this can be reduced, and ii) if a limit upon where emulating a system takes sufficiently long, that the simulator is prefered.

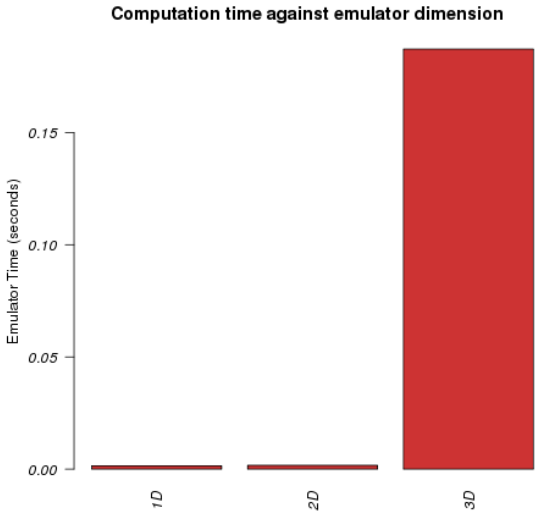


Figure 59: A bar plot showing the change in time taken to predict 10 unknown datapoints using a1D, 2D and 3D emulator .

13.3 Changing the simulation length

Although the simulator may be efficient for short simulations, this is not necessarily the case for longer runs. Figure 64 compares the efficiency of the emulator. Using this a distinct correlation between length and computational gain in using an emulator can be seen. Using this it can be concluded that for longer simulations, emulation is certainly preferable.

14 Conclusions

The viability of using gaussian process emulators as a means to replace the complex chemistry present in atmospheric models was explored. It was found that although a smoothing effect was experienced over discontinuities over the data, asymmetry and the general structure of the chemistry are preserved by the emulator. Although a range of diagnostic tools were explored, the greatest improvement was found to result from using a larger set of training data. In order to provide a 'good' fit, training data is required to span the entirety of the input space. The easiest way to do do this was found to be using a logarithmic scale and the *lhs* library.

Since the atmospheric chemistry involves the interaction of many species, the model was tested in 3 dimensions, which can easily be extended to n-dimensions. It was found that increasing the dimension size, requires t^n training points to generate the same fit as the t-point 1D example. The difficulty with higher dimension GPEs is that they are hard to visualise, and in turn validate visually. For this one of the verification techniques presented need to be used. It is possible to visualise n-dimensional using Trellis plots, although these were beyond the scope (and time) of this project.

Finally benchmarking the emulator against the simulator revealed that minimal effects on the emulator performance are seen from adding additional training points. This means that the GPEs fitting capabilities can be can be improved without consequence. Unfortunately computation time was also found to increase with dimension. Since the test sample was limited, these results should be taken as an indicator that further work needs to be done on higher-dimension emulators. Finally, as the call time for the GPE remains constant, increasing the simulation time-scale result in an ever improving emulator-simulator ratio.

In conclusion, it is possible to emulate a complex chemical system, and although GPEs apply a slight smoothing to the model, all but the most rapid changes in chemistry are preserved. This would make Gaussian Process Emulators ideal for locations with limited chemistry, such as deserts and oceanic air columns.

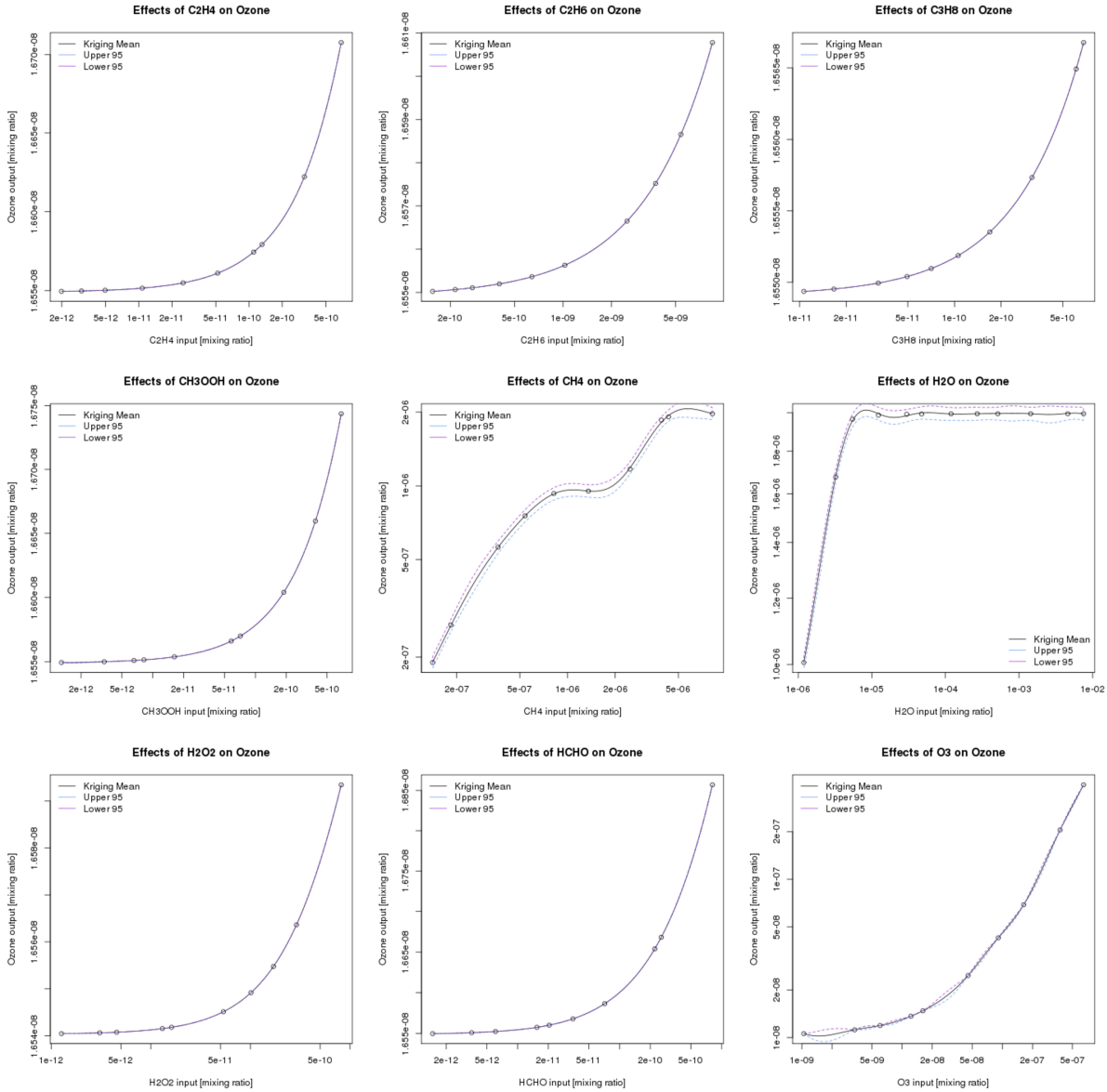


Figure 60: Emulating the effects different species have on ozone production in one dimension.

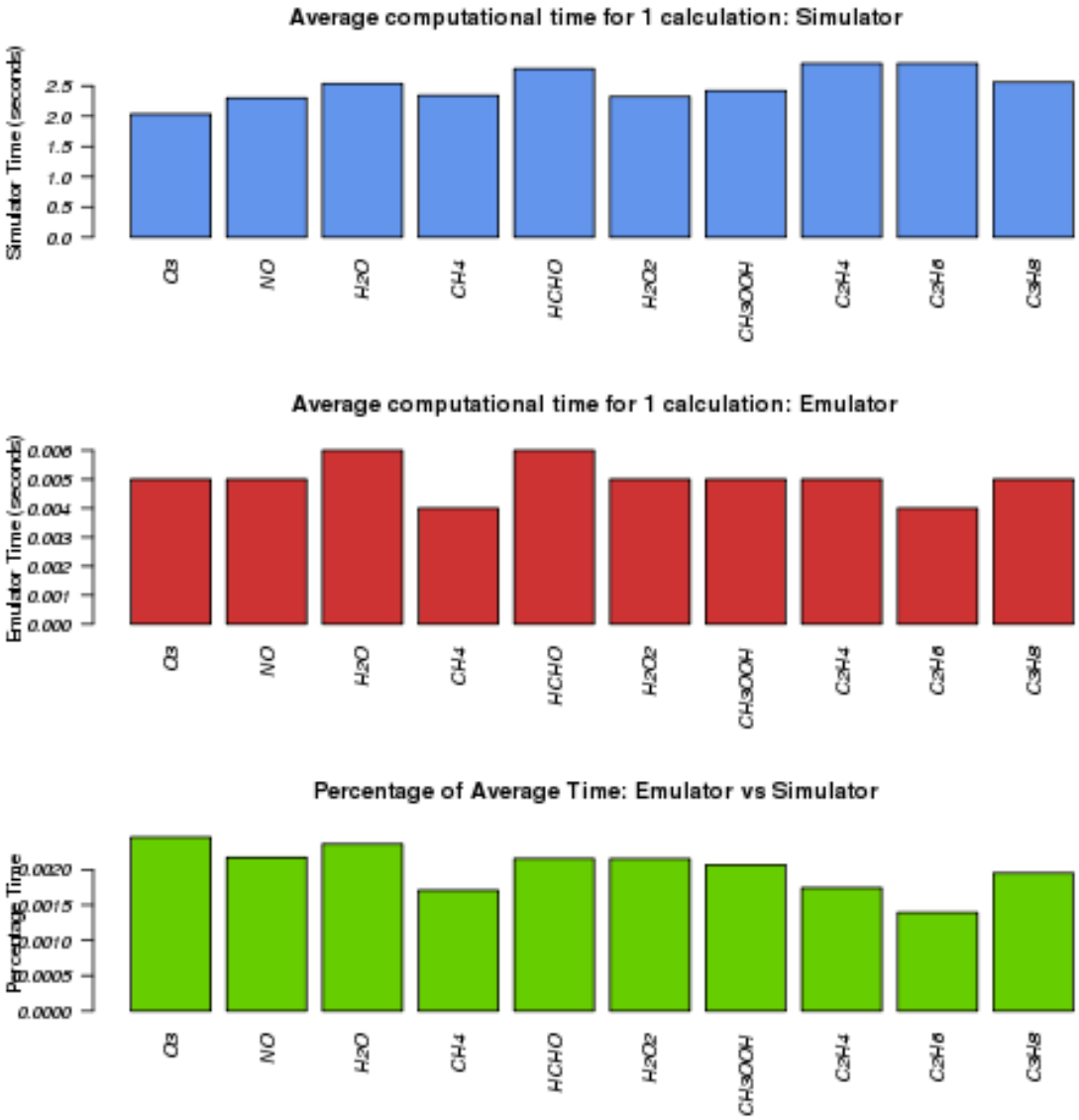


Figure 61: computation time for 1 point for a range of MCM schemes. This uses a 2D emulator

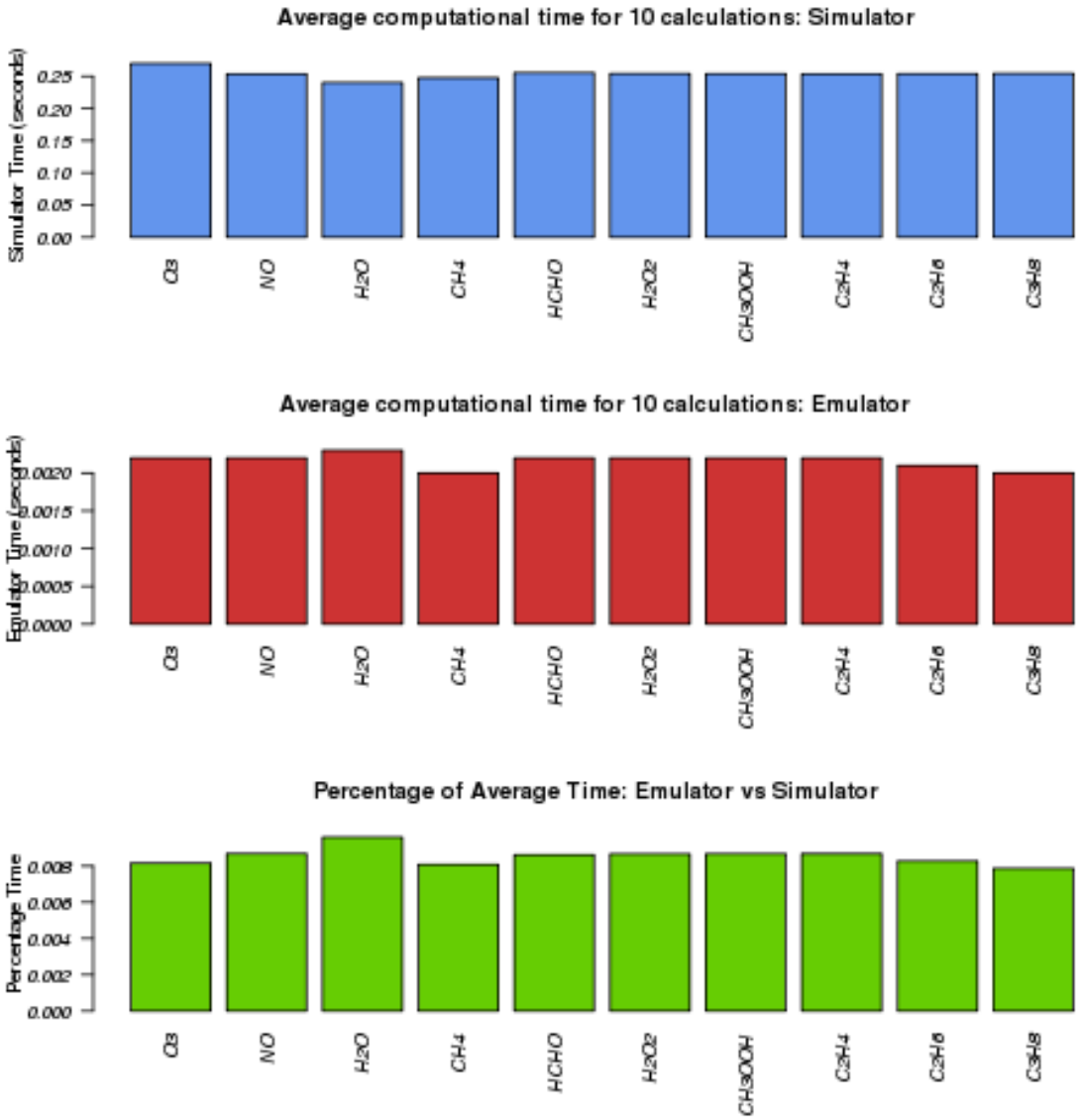


Figure 62: computation time for 10 points for a range of MCM schemes. This uses a 2D emulator

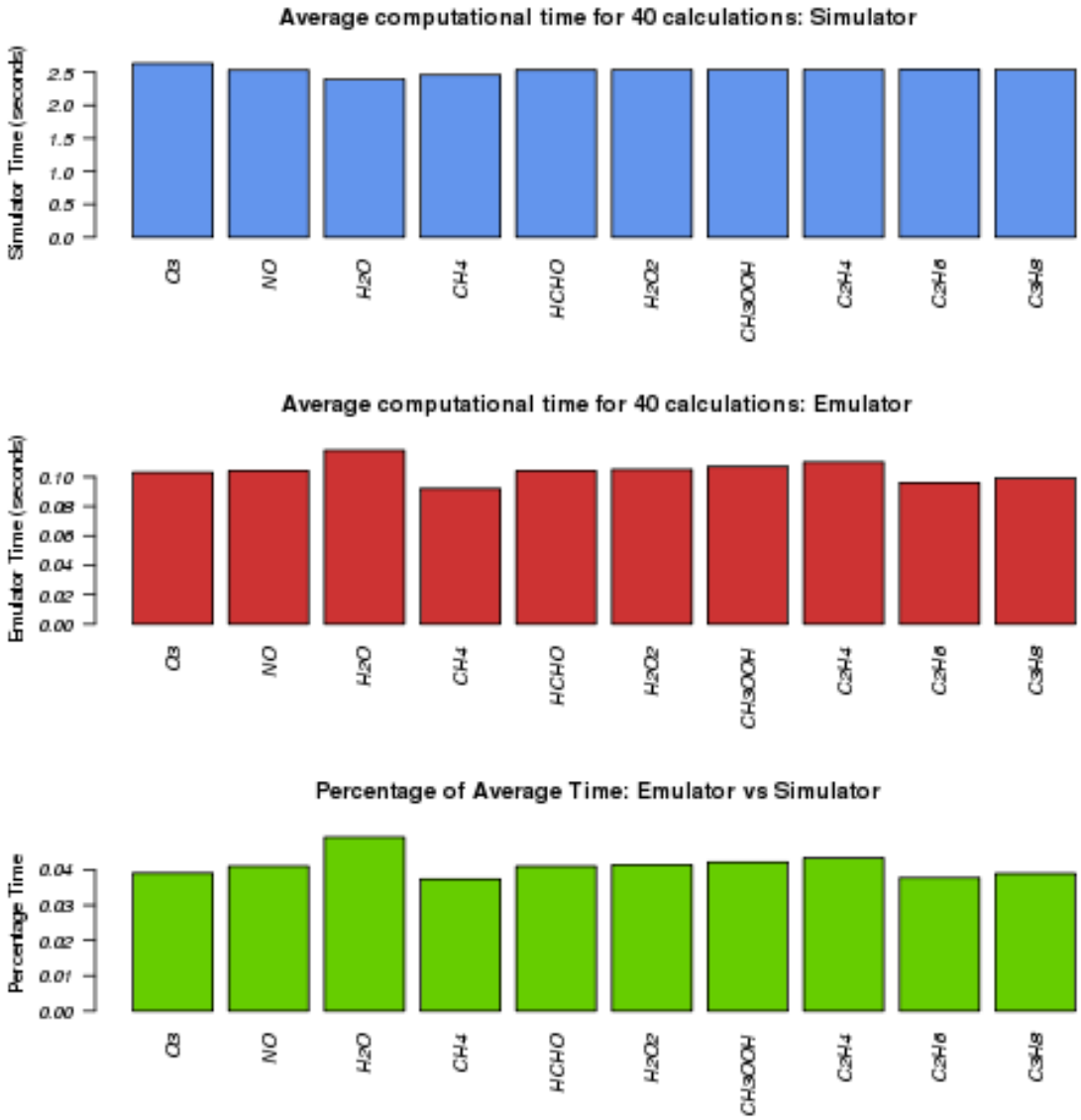


Figure 63: computation time for 1 point for a range of MCM schemes. This uses a 2D emulator

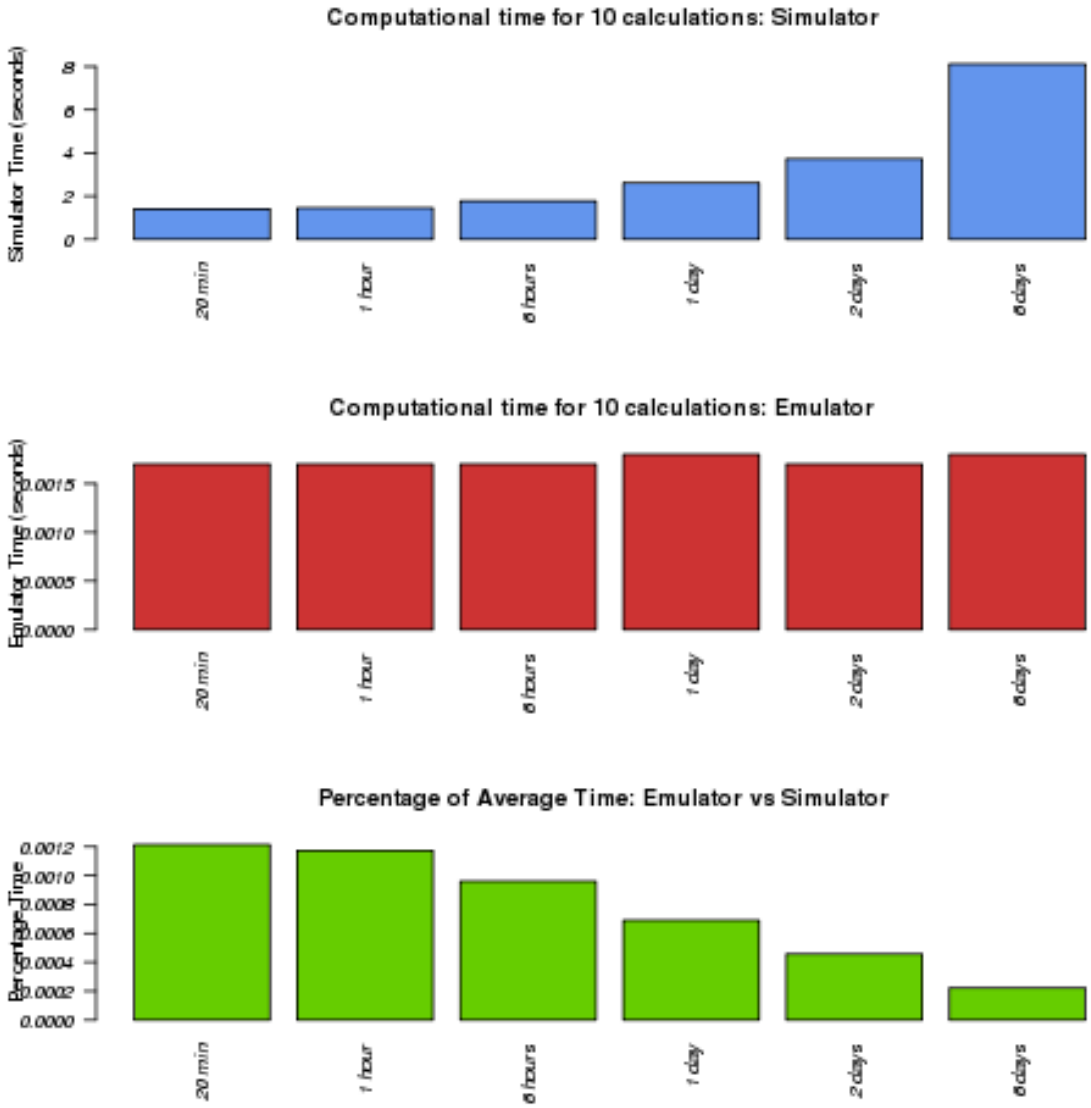


Figure 64: Comparing simulator and a 2D emulator whilst varying the simulation time for a 10 point system.

Appendices

A Program Versions and Installation

This section outlines the libraries and version numbers used in the study, and any installation instructions that may accompany these. This is to be used as a preface to the program documentation provided in Appendix

FDSJL:FAJKLJAF

A.1 The Gaussain Processes Emulator

The following Packages:

- Dice Optim v1.2
- Dice Kriging v1.5.4
- Plot3D v1.0-2
- ColorBrewer v1.1-2

and dependencies are to be installed for R version version *3.1.2 (2014-10-31) – "Pumpkin Helmet"*. This can be done using the CRAN [CITE] database and through the

```
R CMD INSTALL <package name>
```

command in terminal.

Note: Although the DiceKriging package can be installed locally, many of its functions will not run correctly if done this way. It is therefore important that this is installed globally on the cluster it will be used on.

A.2 DSMACC

This can be downloaded and installed using following the instructions on the git repository provided in `https://github.com/barronh/DSMACC/commit/404bc47563e635b3278ca00ec607dd7cd3de9d00`

For the project the Sep 9, 2014 commit was used. Intel's fortran compiler (*ifort*) is required for this.

A.2.1 MCM

Note:*During the project the MCM was updated to Version 3.3, the simple and complex rate coefficients in the file were required to be manually updated to those in [Rickard* et al.*, 2015*b*] and [Rickard* et al.*, 2015*a*] respectively.*

Once installed, a correct chemistry scheme needs to be downloaded for the program. This can be done through `http://mcm.leeds.ac.uk/MCM/home.htt`. From here the required species can be selected using the *browse* selection, and then downloaded in the KPP format using the *extract* menu.

Once this has been done the required species are to be placed in the *organic.kpp* file. Once this has been done the program can be compiled doing the following.

1. Generate the deposition file (*depos.dat*) by running the python file *deposition.py* **Note**: This does not come with the DSMACC download and was written by me. Information on it will therefore be included in the Documentation below.
2. Run the Kinetic PreProcessor:
   `kpp model.kpp`
3. Compile the Fortran files using the *make* command.

A.2.2 Final modifications

After ensuring the model provides correct results, the write statements in the program can be altered to allow the running of my code. As our primary concern is with the chemistry of the model, the line

```
! #INCLUDE ./depos.kpp
```

is to be either removed of commented out.

Simple Models: (e.g. just methane)

The print statements can be directed into the *Rates_1.dat* file, and computation outputs (usually in *Specs_1.dat*) are passed into *sdout*.

The locations of write statements in the **FORTRAN 90** files can found using *grep* in terminal:

```
grep 'write(6,*)' *
```

followed by a 'find and replace' function from any graphical editor.

It was found that the following changes had to be made:

Filename	Change
model_Util.f90	In subroutine **NewSaveData** replace the *write(10,*)* statement with a *write(6,*)* (Specs_1.dat) to write the model output to screen
model_Init.f90	All *write(6,*)* changed to *write(12,*)*
model_Main.f90	All *write(6,*)* changed to *write(12,*)*

Table 4: Changes made to the DSMACC fortran files

This means that the R program can read the output using the system command with the following argument:

```
cmd <- ( "..." ) # command to run the model
output <- system ( cmd , intern= TRUE )
```

More Complex Models:
(e.g. methane, ethane, propanol and isoprene)

It was found that these have a species output too large for reading in using the system command. For this reason the DSMACC files were altered to store the output in a file, and a separate header file was created. The separate file is placed within a separate directory in order to preserve the layout and cleanness of the DSMACC files.

Write statements in *model_Init* and *model_Main* are removed as before, however the only changes to *model_Util* are the creation of a new 'header' file for each run:

```
!model_Util.f90
! subroutine NewInitSaveData !!!!

! make header1.dat
filename = 'headers'//TRIM(filename1)//'.dat'
open(13, file=filename)
! Change write(10,*) statement to add specie
    names to header file

write (13,'(100000(a25,"!"))')
    'TIME','LAT','LON','PRESS','TEMP','H2O',
    'M', 'JNO2FACT','JO1DFACT','RO2', SPC_NAMES
```

The reading of DSMACC inputs/arguments:

Firstly the variable type needs to be defined within the global's file:

```
! model_globals.f90:
character(len=100)::args
```

Next a new file to create the initial conditions for each run needs to be created. The following changes need to be made to *model_init.f90*:

```
! model_Init.f90:
! subroutine InitVal
call get_command_argument(1,args) ! get arguments
```

Under the conditionals *if (counter .eq. 0)* and *if (counter .eq. 0)* change the *OPEN* statement to :

```
open(UNIT=21,FILE='initcons/Init_cons'//
                              trim(args)//'.dat')

!model_Util.f90
!  subroutine NewInitSaveData
call get_command_argument(1,args) ! get arguments

!spec file
filename='simresults/'//trim(args)//'.dat'
    open(10, file=filename)
!rate file
filename='output_'//TRIM(filename1)//'.dat'
       open(12, file=filename)
```

As we do not desire to store all the model results, we can opt to rewind the stored data (only write the most current). This decrees the file size for a day's simulation from 25MiB to 6kb. :

```
!model_Util.f90
!  subroutine NewSaveData
! Only record latest result to Spec file
  REWIND(10)
```

Important: DSMACC needs to be recompiled whenever there are any changes made to the fortran files. This can be done using the *make* command.

A.2.3 Increasing the Rate coefficient of CH4 (Optional)

The rate constants for all reactions are located within the *organic.kpp* file (the file containing the MCM).

These can be changed by doing changing the following lines:

```
! organic.kpp

! replace:
!OH + CH4 = CH3O2 :   1.85D-12*EXP(-1690/TEMP)

! with:
 OH + CH4 = CH3O2 :   2.7D-11*EXP(390/TEMP)
```

This increases the reaction rate of methane to one similar to that of isoporene.

Unfortunately in doing this, it is required that the model is once again recompiled using kpp and made, as one would using a new MCM scheme.

A.2.4 Running DSMACC

As inputs (initial conditions) require a specific layout and exact spacing, a python program was written to generate these. *makeics.py* and *makeics_shell* are included in the main program documentation. These can be used to generate the *Init_cons* file required by the model.

Once this has been done, the model can be executed in the usual manner:

```
./model
```

B Program Documentation

This section will provide the most useful parts of the programs used. It will be split into two sections, Programs,Libraries and Modules, and functions within them. First the background function files shall be described, then the other code, such as that used to make the emulators will be included.

Disclaimer: Code included is provided to aid someone in making an emulator suited to their needs, it will not necessarily work directly[18]

B.1 Initial conditions file

This is a file that reads in the concentrations using command line arguments, and creates the required init_cons file for the DSMACC program.

```
'''
A program to ensure the correct format of the Init_cons file for DSMACC. It should also allow an
    easier comparison between values used and the quick alterations of each.

Species and variables are defined by the conditions list, and simulation time is set by the seconds
    varaible.                                                              [Dan Ellis]
'''
import sys,os

seconds = 60#*60*24

elements =[  #number of spaces here does not, matter
'TEMP   ,    0',
'LAT    ,    0',
'LON    ,    0',
'JDAY   ,    0',
'H2O    ,    0',
'PRESS  ,    0',
'NOx    ,    1',
'NO     ,    1',
 "NO2   ,    0",
 "O3    ,    0",
 "H2O   ,    0",
 "CH4   ,    0",
 "CO    ,    0",
 "HCHO  ,    0",
 "H2O2  ,    0",
 "CH3OOH,    0",
 "C2H4  ,    0",
 "C2H6  ,    0",
 "C3H8  ,    0",
 "C5H8  ,    0"
 ]
```

[18] For instance the 1D emulator program has been adapted to be called from other modules rather than being stand-alone as it was initially.

```
#makes dictionary
if (len(elements) < 10): sys.exit('\n Wrong number of arguments supplied \n')

elements = elements[0:len(sys.argv[2:])]


concentrations= dict(zip(elements,sys.argv[2:]))

#write to the file
ic = open('initcons/Init_cons%s.dat'%sys.argv[1],'w')

ic.write('%s\n'%seconds) # write simulation time

# go through items and write to file.
for i in [0,1,2]:
    for item in [line.replace(' ','').split(',') for line in ['%s , %s'
     %(selection, concentrations[selection]) for selection in elements ]]:

        if i != 2 :
               ic.write('%15s!'%item[i])
        else:
          ic.write('%15s!'%('%.5e'%float(item[2])))


    ic.write('\n')

ic.close()
```

B.2 Deposition.py

Should deposition want to be used in the model, it needs to be taken from the kpp files and placed within a depos.kpp file. This is done here.

```
REACTION_RATE = '1.1D-5*EXP(0./TEMP)' #enter reaction rate here

#function to get the species,
def get_species (filename):
       global species

       for element in [line for line in open('%s.kpp'%filename)]:
     #checks each line of file
                if ('= IGNORE' in str(element)) == True : species.append( element.split(' ')[0])
                #searches for species, labeled with '= IGNORE'
       return None
       #do no return anything

global species
species = []
```

```
for scheme in ['organic', 'inorganic']: get_species(scheme)#forwards correct files to function and
    gets species

depos_file= open('depos.kpp', 'w')
depos_file.write('#EQUATIONS\n')

for num,line in enumerate(species): depos_file.write( '{%s.} %s = DUMMY : %s ;\n'
    %(num,line,REACTION_RATE) )
#correct fomatting and writes to file
```

B.3 functions.r

This is a file containing all the functions used by the emulators. It loads all required libraries, and gets the header file that is to be used when reading in the DSMACC data. All programs will be listed under the appropriate subheadings.

```
#libraries
library('DiceOptim')
library('plot3D')
source('functions1d.r')
source('functions3d.r')

# clean folders before beginning computation
system('cd DSMACC_Repository_new/initcons/ && rm *')
system('cd DSMACC_Repository_new/simresults/ && rm *')
```

B.3.1 Get header file

```
## generate a header string
get.header <- function( TEMP=2.9800e+02,LAT=5.3958e+01,LON=0.1080e+01,
    JDAY=155.409655,H2O=2e-2,PRESS=1013,NOx=0,NO2=0,
    NO=1e-9,O3=21.9813175914e-9,CH4=1e-6    ,
    CO=0,      H2O2=0, HCHO =0 ,CH3OOH =0 ,
    C2H4 =0 , C2H6 =0 , C3H8 =0 , C5H8 =0 )
#add new species here

{system(sprintf('cd DSMACC_Repository_new/ && python makeics_shell.py %s && ./model' ,
    paste(0,TEMP,LAT,LON,JDAY,H2O,PRESS,NOx,NO, NO2, O3 , H2O , CH4 , CO, HCHO , H2O2 , CH3OOH , C2H4
    , C2H6 , C3H8 , C5H8 )))

print('getting headers')

repeat{ #insture headers are read

    headings= unlist(strsplit(gsub(' ','',readLines('DSMACC_Repository_new
        /headers1.dat')), split="!"))
    if (length(headings)>1) break }

return (unlist(strsplit(gsub(' ','',readLines('DSMACC_Repository_new
```

```
       /headers1.dat')), split="!"))))}
```

B.3.2 Function to gather simulator values

```
## get the model output
get.value <- function (
#default parameters
head=header,
z='M', #output value (usually O3)
TEMP=2.9800e+02,
LAT=5.3958e+01,
LON=0.1080e+01,
JDAY=155.4495655,
H2O= 1e-2,
PRESS=1013,
NOx=0,
NO=1e-7,
NO2=0.0,
O3=21.9813175914e-9,
CH4=1.854e-8,
CO=0,
H2O2=0,  #1e-10,
HCHO =0 ,
CH3OOH =0 ,
C2H4 =0 ,
C2H6 =0 ,
C3H8 =0 ,
C5H8 =0 ,
xspec = 'none',
x= 0,
yspec = 'none',
y= 0,
zspec = 'none',
z1= 0
       )
{   # make ICS and run model

       assign(xspec,x)
       assign(yspec,y)
       assign(zspec,z1)


       results <- call_number<<- call_number+1

       inputs <<- paste(results,TEMP,LAT,LON,JDAY
         ,H2O,PRESS,NOx,NO,NO2,O3 , H2O ,     CH4 ,  CO,
           HCHO ,     H2O2 ,     CH3OOH ,     C2H4 ,   C2H6 ,    C3H8 ,    C5H8 )

       #run command as seperate process
```

```
        cmd <<- sprintf('cd DSMACC_Repository_new/ && python makeics_shell.py %s && ./model
            %s',inputs,results)
        sdout = system(cmd,ignore.stdout = TRUE, ignore.stderr = FALSE, intern=TRUE)

        # make a dataframe from the model output
        d <- (t(as.double(unlist(strsplit(readLines(
          sprintf('DSMACC_Repository_new/simresults
          /%s.dat',results)),split=' '))[-1]))

        dsmaccdata <- as.data.frame(d)

        names(dsmaccdata) <- head # apply headers to dataframe.

        # test global dataframe that eliminates all columns with value 0
        df1 <<-dsmaccdata[ , dsmaccdata != 0]
        return (dsmaccdata[z]/M)}     # convert number density to mixing ratio
```

B.3.3 Get the unknown data for a 3D grid

```
###3D fn: get ozone data
#finds uncomputed 3D values and gets them. parallel.

fill.3d <- function(grid, species= 'O3', out.3d=rep(NA,nrow(grid)), ncores =16,rep=3){
        repeat{

                failed <- which(is.na(
                suppressWarnings(as.double(out.3d))))
                lf <- length(failed)
                if (lf == 0 ) break

                print(c('redo',lf))
                new.v <<- grid[failed,]

                if (lf > 33) {#parallel

                        correct <- unlist(mclapply(
                        as.matrix(1:nrow(new.v)), function(n){

                                NOx <- new.v[n,1] # need to convert out of log scale.
                                CH4 <- new.v[n,2]
                  H2O <- new.v[n,3]

                                #ensure consistent values. xspec=xspec, x=x,zspec=zspec,
                                    z1=z,yspec=yspec, y=y,
                                values <- as.double(get.value(NO=NOx
                                , CH4=CH4, H2O=H2O,z=species, head= header))
                                repeat{

                                        new <- as.double(get.value(NO=NOx, CH4=CH4, H2O=H2O,z=species,
                                            head= header))
```

```
                                if (sum(new==values)> rep){return(new);break}
                                values <- c(values,new)

                        } } , mc.cores=ncores))        }


                else { #if less than 32, do in serial

                        correct <- unlist(lapply(as.matrix(1:nrow(new.v)), function(n){

                                NOx <- new.v[n,1] # need to convert out of log scale.
                                CH4 <- new.v[n,2]
                  H2O <- new.v[n,3]

                                return (as.double(get.value(NO=NOx, CH4=CH4, H2O=H2O,z=species, head=
                                    header)))

                        } ))            }


                #update newly calculated values
                for (i in 1:lf) { out.3d[failed[i]] = (correct[i]) }  }
        print(correct)

 return(out.3d)  }
```

B.3.4 Kriging calculation in 3D

Calculates the GPE using two second order polynomials and one third.

```
 #computes the kriging model.
model.3D <- function(grid,grid.z){

        formula <- y ~ poly(X2,2,raw=TRUE)+ poly(X1,2,raw=TRUE)+ poly(X3,3,raw=TRUE)

        suppressWarnings(repeat{   #ensure varience != 0

                model <- km(formula=formula , design=data.frame(X1= grid[,1], X2= grid[,2],X3=grid[,3])
                    , response=data.frame(y=grid.z),covtype="gauss", nugget.estim=TRUE)

                if (model@covariance@sd2 > 0) break         })

        return(model)
}
```

B.3.5 Fill the 2D data grid

```
## 2D fn: get ozone data
```

```
#finds uncomputed 2D values and gets them. parallel.

fill.2d <- function(grid, species= 'O3', out.2d=rep(NA,nrow(grid)), ncores =16,rep=3,yspec='CH4'){
       repeat{

              failed <- which(is.na(suppressWarnings(as.double(out.2d))))
              lf <- length(failed)
              if (lf == 0 ) break

              print(c('redo',lf))
              new.v <- grid[failed,]

              if (lf > 33) {#parallel

                     correct <- unlist(mclapply(as.matrix(1:nrow(new.v)), function(n){

                            NOx <- new.v[n,1] # need to convert out of log scale.

                            #ensure consistent values.
                            values <- as.double(get.value(xspec=yspec,x=new.v[n,2], NO=NOx
                                ,z=species))
                            repeat{

                                   new <- as.double(get.value(xspec=yspec,x=new.v[n,2], NO=NOx
                                       ,z=species))

                                   if (sum(new==values)> rep){return(new);break}
                                   values <- c(values,new)

                     } } , mc.cores=ncores))       }


              else { #if less than 33, do in serial

                     correct <- unlist(lapply(as.matrix(1:nrow(new.v)), function(n){

                            NOx <- new.v[n,1] # need to convert out of log scale.

                            return (as.double(get.value(xspec=yspec,x=new.v[n,2],NO=NOx ,z=species)))

                     } ))           }


              #update new.vly calculated values
              for (i in 1:lf){ out.2d[failed[i]] = (correct[i]) }  }

 return(out.2d)  }
```

B.3.6 Altered q-q function and verification test whether to keep a kriging model

```
##QPLOT CODE TO GIVE SLOPE
qqline <- function (y, datax = FALSE, distribution = qnorm, probs = c(0.25,
    0.75), qtype = 7, ...)
    {
        stopifnot(length(probs) == 2, is.function(distribution))
        y <- quantile(y, probs, names = FALSE, type = qtype, na.rm = TRUE)
        x <- distribution(probs)

        if (datax) {
            slope <- diff(x)/diff(y)
            int <- x[1L] - slope * y[1L]
        }
        else {
            slope <<- diff(y)/diff(x)
            int <- y[1L] - slope * x[1L]
        }
    }


##TEST if keep model
accept <- function(model){

    trend.reestim = FALSE
    pred <- leaveOneOut.km(model, type = 'SK', trend.reestim = trend.reestim)
    y <<- as.matrix(model@y)
    yhat <<- pred$mean
    sigma <- pred$sd
    resid <- (y - yhat)/sigma
    xmin <- min(min(yhat), min(y))
    xmax <- max(max(yhat), max(y))

    qqline(resid)

    #conditions
    print(slope)
    if ((slope > .6) & (slope < 1.4)){
        if (mean(abs(yhat-y)) < 0.6) {
            if ( abs(mean(yhat-y)) < 0.1 ) {
                return(TRUE)}}}

    return(FALSE)

    }
```

B.4 1D Program

A program for generating a 1D GPE. Can be used from the 2D program with global variables.

```
source('functions.r')
ncores=16
```

```
out='O3'
#specie='NO'
#minin=12
#maxin=7
npts= (minin-maxin) *5
if (npts < 10) {npts=10}

#if (exists('CH4')==FALSE) {CH4 <<- 1e-8}
#if (is.na(CH4)){CH4 <<- 1e-8}

#################LHS inputs#################################
inarr <- (10**( (minin-maxin)* randomLHS(npts, 1) - minin ))

################populate ozone#############################
df.1d <- suppressWarnings(fill.data.1d(inarr, ncores=ncores,species=out ,xspec=specie) )

data = df.1d
names(data)<- c('input','output')

#par(mfrow = c(3, 1))

#
#png(paste(specie,'1d.png'))

plot((data$input), (data$output), log='xy',xlab=paste(specie , 'input [mixing ratio]'), ylab='Ozone
    output [mixing ratio]', main= paste('1D simulation of', specie,'and Ozone'))


################Kriging of log data ########################################
df.1d=log10(data)

       formula <- y ~ poly(x,2,raw=TRUE)

       suppressWarnings(repeat{
              model.1d <- km(formula=formula , design= data.frame(x=df.1d$input),
                  response=data.frame(y=df.1d$output), covtype="gauss", nugget.estim=TRUE)

              if (model.1d@covariance@sd2 > 0) break  })#ensure varience != 0
       model.1d

############predict ########################################
       t <- seq(from = min(df.1d$input), to = max(df.1d$input) , length = length(inputs)*1000)
       p <- predict(model.1d, newdata = data.frame(x=t), type = "SK")

       lines( 10**t, 10**p$mean, col='black', lty=1)
       lines( 10**t, 10**p$lower95, col='#6495ED', lty=2)
       lines( 10**t, 10**p$upper95, col='darkorchid3', lty=2)

loc<<-'topleft'
legend(loc, c("Kriging Mean","Upper 95", 'Lower 95') ,
```

```
  lty=1, col=c('black', '#6495ED','darkorchid3'), bty='n', cex=1)
```

B.5 2D

2D GPE Program

```
source('functions.r')
ncores=16
minxin=10      ;       maxxin=5
minyin=10  ;   maxyin=5

redocounter=0
ppo=1

repeat{

    npts=(minxin-maxxin)*(minyin-maxyin) * ppo
    if (npts < 20) { npts=20 }

    #LHS
    LHSinputs = randomLHS(npts, 2)
    LHSinputs[,1] <- (10**( (minxin-maxxin)* LHSinputs[,1] - minxin ))
    LHSinputs[,2] <- (10**( (minyin-maxyin)* LHSinputs[,2] - minyin ))
    grid <- LHSinputs
    names(grid) <- c('X1','X2')
    # Populate LHSmk.
    popval <- log10(as.double((fill.2d(grid,ncores=ncores))))
    grid.z <- suppressWarnings(popval)

    #empty grid
    #change to deal with log 10


    len=10
    Xin <- -1*rev(seq(maxxin,minxin,length=len*abs(maxxin-minxin)))
    Yin <- -1*rev(seq(maxyin,minyin,length=len*abs(maxyin-minyin)))
    np.grid <- expand.grid(Xin,Yin)
    names(np.grid) <- c("X1","X2")

    # get the kriging model
    model <- get.model(log10(grid),grid.z)
    model


    #predict ## ensure np.grid is in logarithmic terms.
    pred.grid <- predict.km(model, newdata=((np.grid)), type="SK")
    mk.grid = 10**pred.grid$m
    mk.grid
```

```
    contour2D(x=1:length(Xin),y= 1:length(Yin),z= matrix(mk.grid, length(Xin), length(Yin)),
        nlevels=40,xaxt='n',yaxt='n',colkey = list(plot = FALSE, side = 4),xlab='x \n')

    #verification conditions.
    if (accept(model)){break}
    else {redocounter = redocounter+1
        print(paste('redoing for the', redocounter))}

    if (redocounter>2){
        ppo=ppo+1
        redocounter=0
        if (ppo > 5){break}}#every 3 reestimates increase number of points.

}


#Validate
    addmodel=data.frame()
    addvalue=numeric()

    nvalues=(minxin-maxxin)*(minyin-maxyin)
    nvalues=10
    LHSval = randomLHS(nvalues, 2)
    LHSval[,1] <- (10**( (minxin-maxxin)* LHSval[,1] - minxin ))
    LHSval[,2] <- (10**( (minyin-maxyin)* LHSval[,2] - minyin ))
    valGrid <- LHSval

    valZ <- log10(as.double((fill.2d(valGrid,ncores=ncores))))
    valPred <- predict.km(model, newdata=(valGrid), type="SK")
    lower <- valPred$lower95
    upper <- valPred$upper95

    for (i in 1:length(valZ)){
        print(c(valZ[i],upper[i],lower[i]))
        if ((valZ[i] < upper[i]) & (valZ[i] > lower[i])){print('pass')}
        else { addmodel= rbind(addmodel,valGrid[i,])
               addvalue=c(addvalue,valZ[i])} }

colnames(grid) <- c('X1','X2')
colnames(addmodel) <- c('X1','X2')
grid = rbind(grid,addmodel)
grid.z= c(grid.z,addvalue)



    # get the kriging model
    model <- get.model(log10(grid),grid.z)
    model
```

```
    #predict ## ensure np.grid is in logarithmic terms.
    pred.grid <- predict.km(model, newdata=((np.grid)), type="SK")
    mk.grid = 10**pred.grid$m
    upper=10**pred.grid$upper95
    lower=10**pred.grid$lower95
    mk.grid

    contour2D(x=1:length(Xin),y= 1:length(Yin),z= matrix(mk.grid, length(Xin), length(Yin)),
        nlevels=40,xaxt='n',yaxt='n',colkey = list(plot = FALSE, side = 4),xlab='x \n')



    contour2D(x=1:length(Xin),y= 1:length(Yin),z= matrix(mk.grid, length(Xin), length(Yin)),
        nlevels=40,xaxt='n',yaxt='n',colkey = list(plot = FALSE, side = 4),xlab='x \n')


ntick = 6
a= c(seq(1,length(Xin),length(Xin)/ntick),length(Xin))
b= c(seq(1,length(Yin),length(Yin)/ntick),length(Yin))


axis(1, at=a, labels=signif(10**Xin[a],1))
axis(2, at=b, labels=signif(10**Yin[b],1))
```

B.6 3D Program

```
source('functions.r')
ncores=8
minxin=10     ;      maxxin=5
minyin=10  ;   maxyin=5
minzin=7   ;   maxzin=2


npts=(minxin-maxxin)*(minyin-maxyin)*(minzin-maxzin) * 30
if (npts < 30) { npts=30 }

#LHS
inputs = randomLHS(npts, 3)
inputs[,1] <- (10**( (minxin-maxxin)* inputs[,1] - minxin ))
inputs[,2] <- (10**( (minyin-maxyin)* inputs[,2] - minyin ))
inputs[,3] <- (10**( (minzin-maxzin)* inputs[,3] - minzin ))
grid=as.data.frame(inputs)

# Populate LHSmk.
popval <- log10(as.double((fill.3d((grid),ncores=ncores))))
grid.out <- suppressWarnings(popval)



# get the kriging model
```

```
model <- model.3D( log10(grid), grid.out)
model

save(model,file='3ddata')



load('z3ddata')

scale=1e1
Xin <- -1e-1*seq( minxin*scale, maxxin*scale, -1)
Yin <- -1e-1*seq( minyin*scale, maxyin*scale, -1)
Zin <- -1e-1*seq( minzin*scale, maxzin*scale, -1)


#np.grid <- expand.grid(Xin,Yin,rep(-6,length(Xin)))

P1=Xin
P2=Yin
water= 1e-2


#predict ## ensure np.grid is in logarithmic terms.
X<-expand.grid(P1,P2)
X$X3 = log10(water)

newdata <- as.data.frame(X) #data.frame(X1=X[,3],X2=X[,2],X3=X[,1]) #as df x
colnames(newdata) <- colnames(model@X)


pred.grid <- predict.km(object=model, newdata=newdata[60,], type="SK")


mk.grid<- unlist(mclapply(as.matrix(1:nrow(newdata)), function(x){ 10**predict.km(object=model,
    newdata=newdata[x,], type="SK")$m },mc.cores=8))

sd <- unlist(mclapply(as.matrix(1:nrow(newdata)), function(x){ 10**predict.km(object=model,
    newdata=newdata[x,], type="SK")$sd },mc.cores=8))



#mk.grid = 10**pred.grid$m
#mk.grid


contour2D(x=1:length(P1),y= 1:length(P2),z= matrix(mk.grid, length(Xin), length(Yin)),
    nlevels=20,xaxt='n',yaxt='n',colkey = list(plot = FALSE, side = 4),xlab= 'NOx
    [log10(ppb)]',ylab='Water [log10(ppb)]')
```

```
ntick = 6
a= c(seq(1,length(P1),length(P1)/ntick),length(P1))
b= c(seq(1,length(P2),length(P2)/ntick),length(P2))
axis(1, at=a, labels=signif(10**P1[a],1))
axis(2, at=b, labels=signif(10**P2[b],1))



newdata <- as.data.frame(expand.grid(Xin,Yin,Zin))
colnames(newdata) <- colnames(model@X)

new=predict.km(object=model, newdata=newdata, type="SK")



persp3D(facets=FALSE,x=(P1+9),y=(P2+9),z= matrix(log10(mk.grid)+9, length(P1), length(P2)),
    nlevels=40,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [log10(ppbv)]',ylab='\n\n Methane [log10(ppbv)]',zlab='\n\n Ozone [log10(ppbv)]',main='2D plot of
    the 3D GPE')
```

B.7 Useful plotting commands

These the Plot3D library.

B.7.1 Overlay Contour

```
#overlay contours
contour(x=1:length(Xin),y= 1:length(Yin),z= matrix(10**mk, length(Xin), length(Yin)),
    nlevels=80,xaxt='n',yaxt='n',col='azure4',xlab='\n\n NOx inputs [ppbv] \n',ylab='Methane inputs
    [ppbv]', main = 'Ozone output for 3D Emulator and Simulator Computation',xlim=c(0,48),
    ylim=c(20,49) )

contour(x=1:length(Xin),y= 1:length(Yin),z= matrix(mk.grid, length(Xin), length(Yin)),
    nlevels=30,xaxt='n',yaxt='n',col='darkmagenta',xlab='x \n',add=TRUE)



legend('bottomright', c("Actual Simulation","Kriging Mean") ,
   lty=1, col=c('azure4', 'darkmagenta'), bty='n', cex=1)

ntick = 6
a= c(seq(1,length(Xin),length(Xin)/ntick),length(Xin))
b= c(seq(1,length(Yin),length(Yin)/ntick),length(Yin))
axis(1, at=a, labels=format(signif(10**(Xin[a]+9),1),scientific=TRUE))
axis(2, at=b, labels=format(signif(10**(Yin[b]+9),1),scientific=TRUE))
```

B.7.2 Different plot styles when comparing the difference between simulator and emulator

```
#difference
```

```
diff= mk - log10(mk.grid)



persp(x=(Xin+9),y=(Yin+9),z= matrix(((diff)), length(Xin), length(Yin)),
    nlevels=40,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [log10(ppb)]',ylab='\n\n Methane [log10(ppb)]',zlab='\n\n Ozone difference [Mixing Ratio] ',main='
    Simulation - 3D Emulation Ozone')



persp(x=(Xin+9),y=(Yin+9),z= matrix(abs((diff/mk)*100), length(Xin), length(Yin)),
    nlevels=40,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [log10(ppb)]',ylab='\n\n Methane [log10(ppb)]',zlab='\n\n Percentage difference ',main='Absolute
    Percentage difference: Simulation - 3D Emulation Ozone')



   contour2D(x=1:length(Xin),y= 1:length(Yin),z= matrix(diff, length(Xin),
       length(Yin)),xaxt='n',yaxt='n', nlevels=30,colkey = list(plot = FALSE, side = 4),xlab='\n\n
       NOx input [Mixing Ratio]',ylab='\n\n Methane input [Mixing Ratio]',main=' Simulation - 3D
       Emulation Ozone')

ntick = 6
a= c(seq(1,length(Xin),length(Xin)/ntick),length(Xin))
b= c(seq(1,length(Yin),length(Yin)/ntick),length(Yin))
axis(1, at=a, labels=format(signif(10**(Xin[a]),1),scientific=TRUE))
axis(2, at=b, labels=format(signif(10**(Yin[b]),1),scientific=TRUE))



persp(x=(Xin+9),y=(Yin+9),z= matrix(10**(abs(diff)), length(Xin), length(Yin)),
    nlevels=40,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [log10(ppb)]',ylab='\n\n Methane [log10(ppb)]',zlab='\n\n Absolute error between simulator and
    emulator',main='A 3D representation of simulator results')
```

B.7.3 Expected Improvement

```
#EI

EI.grid <- apply(np.grid, 1, EI, model)

contour(x=1:length(Xin),y= 1:length(Yin),z= matrix(sd, length(Xin), length(Yin)),
    nlevels=30,xaxt='n',yaxt='n',col='darkmagenta',xlab='x \n',add=FALSE)

contour2D(x=1:length(Xin),y= 1:length(Yin),z= matrix((EI.grid), length(Xin), length(Yin)),
    nlevels=20,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [log10(ppb)]',ylab='\n\n Methane [log10(ppb)]',zlab='\n\n EI',main='A 3D representation of
    simulator results',phi=20,add=TRUE)
```

```
image2D(x=(Xin),y=(Yin),z= matrix((EI.grid), length(Xin), length(Yin)),
    nlevels=20,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [Mixing Ratio]',ylab='\n\n Methane [Mixing Ratio]',zlab='\n\n EI',main='Expected Improvement
    Criterion',contour=TRUE,rasterImage = TRUE)
```

B.7.4 Error between confidence levels (same as the standard deviation)

```
#upper and lower
   upper=10**pred.grid$upper95
   lower=10**pred.grid$lower95
   bounds=upper-lower


persp3D(x=(Xin),y=(Yin),z= matrix((bounds/2)/lower, length(Xin), length(Yin)),
    nlevels=40,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    input [Mixing Ratio]',ylab='\n\n\n Methane input [Mixing Ratio]\n\n',zlab='\n (upper 95 -lower
    95)/2 ',main='95% Confidence level: Width of Uncertainty')

#sdev

sd=pred.grid$sd


persp3D(x=(Xin),y=(Yin),z= matrix(sd, length(Xin), length(Yin)),
    nlevels=40,ticktype="detailed",axes=TRUE, zlim=(c(.17,.1)),colkey = list(plot = FALSE, side =
    4),xlab='\n\n NOx input [Mixing Ratio]',ylab='\n\n\n Methane input [Mixing Ratio]\n\n',zlab='\n
    Standard Deviation ',main='Standard Deviation of Kriging Model',contour = list(nlevels = 20, col =
    "blue"))


persp(x=(Xin+9),y=(Yin+9),z= matrix((sd), length(Xin), length(Yin)),
    nlevels=40,ticktype="detailed",axes=TRUE,colkey = list(plot = FALSE, side = 4),xlab='\n\n NOx
    [log10(ppb)]',ylab='\n\n Methane [log10(ppb)]',zlab='\n\n sd',main='sdev 95')
```

B.7.5 3D glasses plot

```
#3D glases plot
anaglyph.plot(x=1:length(Xin),y= 1:length(Yin),z= matrix(10**mk.grid, length(Xin),length(Yin)), left =
    "red", right = "cyan", depth = "med", style = "pop-out", type = "p", ...)
```

References

[Andrianakis and Challenor, 2011] Andrianakis, Y. and Challenor, P. (2011). Super-parameterisation of ocean deep convection. Technical report University of Sheffield.

[Bastos, 2010] Bastos, L. (2010). Validating Gaussian Process Models in Computer Experiments. PhD thesis, The University of Sheffield.

[Bastos and O'Hagan, 2009] Bastos, L. S. and O'Hagan, A. (2009). Diagnostics for Gaussian Process Emulators. Technometrics *51*, 425 – 439.

[BIPM, 2015] BIPM (2015). Online. `http://www.bipm.org/jsp/en/ViewCGPMResolution.jsp?CGPM=10&RES=4` (Viewed: April 2015).

[Bohling, 2005] Bohling, G. (2005). KRIGING. `http://people.ku.edu/~gbohling/cpe940/Kriging.pdf` (Visited April 2015).

[Carnell, 2007] Carnell, R. (2007). An Example of Augmenting a Latin Hypercube. `http://cran.r-project.org/web/packages/lhs/vignettes/augmentLHS_Example.pdf`(Viewed: April 2015).

[Carnell, 2013] Carnell, R. (2013). lhs: Latin Hypercube Samples. `http://cran.r-project.org/web/packages/lhs/index.html`(Viewed: April 2015).

[Cazenave and Nerem, 2004] Cazenave, A. and Nerem, R. S. (2004). Present-day sea level change: Observations and causes. Rev. Geophys *42*.

[Chapman et al., 1994] Chapman, W. L., Welch, W. J., Bowman, K. P., Sacks, J. and Walsh, J. E. (1994). Arctic sea ice variability: Model sensitivities and a multidecadal simulation. Journal of Geophysical Research: Oceans *99*, 919–935.

[Crutzen, 1973] Crutzen, P. (1973). A discussion of the chemistry of some minor constituents in the stratosphere and troposphere. pure and applied geophysics *106-108*, 1385–1399.

[Emmerson and Evans, 2009] Emmerson, K. and Evans, M. (2009). Comparison of tropospheric gas-phase chemistry schemes for use within global models. Atmospheric Chemistry and Physics *9*, 1831 – 1845.

[Evans, 2014] Evans, M. (2014). Private Correspondance.

[Fang et al., 2005] Fang, K., Li, R. and Sudjianto, A. (2005). Design and Modeling for Computer Experiments. Chapman & Hall/CRC Computer Science & Data Analysis, CRC Press.

[Finlayson-Pitts and Pitts, 2000] Finlayson-Pitts, B. and Pitts, J. (2000). Chemistry of the Upper and Lower Atmosphere: Theory, Experiments, and Applications. Academic Press.

[Ginsbourger et al., 2013] Ginsbourger, D., Picheny, V., Roustant, O., Chevalier, C. and Wagner, T. (2013). Package 'DiceOptim'. CRAN.

[Henderson-Sellers, 2015] Henderson-Sellers (2015). Online. `http://d32ogoqmya1dw8.cloudfront.net/images/eet/envisioningclimatechange/gcm_grid_graphic.jpg` (Visited Feb 2015).

[IPCC, 2007] IPCC (2007). Climate Change: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report. IPCC *4*.

[IPCC, 2013] IPCC (2013). Climate Change: The Physical Science Basis: Working Group I Contribution to the Fifth Assessment Report. IPCC *5*.

[Jamieson, 2014] Jamieson, D. (2014). Reason in a Dark Time: Why the Struggle Against Climate Change Failed – and What It Means for Our Future. Oxford University Press.

[Jones, 2013] Jones, C. (2013). UKESM: A Met Office-NERC JWCRP collaboration on Earth System Modelling. In MOSAC and SRG Meetings 2013 vol. MOSAC PAPER 18.10, Met Office.

[Kimeldorf et al., 1970] Kimeldorf, S., G. and Wahba, G. (1970). A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines. The Annals of Mathematical Statistics *41*, pp. 495–502.

[Krige, 1951] Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on

the Witwatersrand. Journal of Chemical, Metallurgical, and Mining Society of South Africa *1*.

[Kuß, 2006] Kuß, M. (2006). Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning. PhD thesis, TU Darmstadt.

[Lee et al., 2011] Lee, L. A., Carslaw, K. S., J.Pringle, K., Mann, G. W. and Spracklen, D. V. (2011). Emulation of a complex global aerosol model to quantify sensitivity to uncertain parameters. Atmospheric Chemistry and Physics *11*, 12253–12273.

[Lee et al., 2013] Lee, L. A., Pringle, K. J., Reddington, C. L., Mann, G. W., Stier, P., Spracklen, D. V., Pierce, J. R. and Carslaw, K. S. (2013). The magnitude and causes of uncertainty in global model simulations of cloud condensation nuclei. Atmospheric Chemistry and Physics *13*, 8879–8914.

[Los-Alamis-National-Labratory, 1944] Los-Alamis-National-Labratory (1944). Chapter IV: Technical Review to August 1944. Technical report FAS. `http://www.fas.org/sgp/othergov/doe/lanl/00795708.pdf`.

[Lynch, 2008] Lynch, P. (2008). The origins of computer weather prediction and climate modeling. Journal of Computational Physics *227*, 3431 – 3444.

[Madronich, 2014] Madronich, S. (2014). Atmospheric chemistry: Ethanol and ozone. Nature Geosci *7*, 395–397.

[Mathez and Webster, 2013] Mathez, E. and Webster, J. (2013). The Earth Machine: The Science of a Dynamic Planet. Columbia University Press.

[McKay et al., 2000] McKay, M. D., Beckman, R. J. and Conover., W. J. (2000). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics *42*, 55–61.

[McKee, 1993] McKee, D. (1993). Tropospheric Ozone: Human Health and Agricultural Impacts. Taylor & Francis.

[NCADAC, 2014] NCADAC (2014). Climate Change Impacts in the United States.

[NIPCC, 2010] NIPCC (2010). Climate Change Reconsidered II: Physical Science. NIPCC *1*.

[O'Hagan, 2014] O'Hagan, A. (2014). GP-modelling.ppt. Online Powerpoint. urlwww.tonyohagan.co.uk/academic/GEM/GP-modelling.ppt(Viewed: October 2014).

[O'Hagan and Kingman, 1978] O'Hagan, A. and Kingman, J. F. C. (1978). Curve Fitting and Optimal Design for Prediction. Journal of the Royal Statistical Society. Series B (Methodological) *40*, pp. 1–42.

[Phillips, 1956] Phillips, N. A. (1956). The general circulation of the atmosphere: A numerical experiment. Quarterly Journal of the Royal Meteorological Society *82*, 123–164.

[Picheny et al., 2013] Picheny, V., Wagner, T. and Ginsbourger, D. (2013). A benchmark of kriging-based infill criteria for noisy optimization. Structural and Multidisciplinary Optimization *48*, 607–626.

[Platt and Stutz, 2008] Platt, U. and Stutz, J. (2008). Differential Optical Absorption Spectroscopy: Principles and Applications. Physics of Earth and Space Environments, Springer Berlin Heidelberg.

[Rasmussen and Williams, 2006] Rasmussen, C. and Williams, C. (2006). Gaussian Processes for Machine Learning. Adaptative computation and machine learning series, University Press Group Limited.

[Richet, 2015] Richet, Y. (2015). Online. `http://glimmer.rstudio.com/richetyann/DiceKriging/`
(Viewed: October 2014).

[Rickard et al., 2015a] Rickard, A., Young, J. and Pascoe, S. (2015a). Online. `http://mcm.leeds.ac.uk/MCM/parameters/complex.htt`
(Viewed: April 2015).

[Rickard et al., 2015b] Rickard, A., Young, J. and Pascoe, S. (2015b). Online. `http://mcm.leeds.ac.uk/MCM/parameters/simple.htt`
(Viewed: April 2015).

[Rougier et al., 2006] Rougier, J., Sexton, D. M., Murphy, J. M. and Stainforth, D. (2006). Emulating the sensitivity of the hadsm3 climate model

using ensembles from different but related experiments. http://www.maths.bristol.ac.uk/research/stats/reports/2007/0707.pdf.

[Roustant et al., 2012] Roustant, O., Ginsbourger, D. and Deville, Y. (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. Journal of Statistical Software *51*, 1–55.

[Roustant et al., 2014] Roustant, O., Ginsbourger, D., Deville, Y., Chevalier, C. and Richet, Y. (2014). Package 'DiceKriging'. CRAN.

[Sacks et al., 1989] Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). Design and Analysis of Computer Experiments. JSTOR *4*, 409–423.

[Sanderson et al., 2008] Sanderson, B. M., Knutti, R., Aina, T., Christensen, C., Faull, N., Frame, D. J., Ingram, W. J., Piani, C., Stainforth, D. A., Stone, D. A. and Allen, M. R. (2008). Constraints on Model Response to Greenhouse Gas Forcing and the Role of Subgrid-Scale Processes. Journal of Climate *21*, 2384–2400.

[Sandu and Sander, 2006] Sandu, A. and Sander, R. (2006). Technical note: Simulating chemical systems in Fortran90 and Matlab with the Kinetic PreProcessor KPP-2.1. Atmospheric Chemistry and Physics *6*, 187–195.

[Shindell et al., 2012] Shindell, D., Kuylenstierna, J. C. I., Vignati, E., van Dingenen, R., Amann, M., Klimont, Z., Anenberg, S. C., Muller, N., Janssens-Maenhout, G., Raes, F., Schwartz, J., Faluvegi, G., Pozzoli, L., Kupiainen, K., Höglund-Isaksson, L., Emberson, L., Streets, D., Ramanathan, V., Hicks, K., Oanh, N. T. K., Milly, G., Williams, M., Demkine, V. and Fowler, D. (2012). Simultaneously Mitigating Near-Term Climate Change and Improving Human Health and Food Security. Science *335*, 183–189.

[Thomas et al., 2004] Thomas, C. D., Cameron, A., Green, R. E., Bakkenes, M., Beaumont, L. J., Collingham, Y. C., Erasmus, B. F. N., de Siqueira, M. F., Grainger, A., Hannah, L., Hughes, L., Huntley, B., van Jaarsveld, A. S., Midgley, G. F., Miles, L., Ortega-Huerta, M. A., Townsend Peterson, A., Phillips, O. L. and Williams, S. E. (2004). Extinction risk from climate change. Nature *427*, 145–148.

[Vallero, 2014] Vallero, D. (2014). Fundamentals of Air Pollution. Elsevier Science.

[von Neumann, 1995] von Neumann, J. (1995). Can we survive technology. Forbes.

[Wilk and Gnanadesikan, 1968] Wilk, M. and Gnanadesikan, R. (1968). Probability plotting methods for the analysis for the analysis of data. Biometrika *55*, 1–17.

[Willis and Hooke, 2006] Willis, E. P. and Hooke, W. H. (2006). Cleveland Abbe and American Meteorology, 1871–1901. Bulletin of the American Meteorological Society *87*, 315–326.