Fast Gaussian Process Emulation of Mars Global Climate Model

Marc Tunnell[1], **Nathaniel Bowman**[1], **Erin Carrier**[1]

[1]Allendale, Michigan, Grand Valley State University

Key Points:

- Gaussian Proccess emulation is effective, vastly decreasing the required runtime for sensitivity studies using the Mars Global Climate Model
- The use of Gaussian Process emulation is remarkably accurate for the Mars Global Climate Model
- Gaussian Process emulation has significant potential to support more refined studies that would otherwise be computationally unfeasible

Corresponding author: Erin Carrier, carrieer@gvsu.edu

Abstract

The NASA Ames Mars Global Climate Model (MGCM) software has been in steady use at NASA for decades and was recently released to the public. This model simulates the complex interactions of various weather cycles that exist on Mars, namely the Dust Cycle, the CO2 Cycle, and the Water cycle. Utilized by NASA, the MGCM is used to help understand their empirically observed data through the use of sensitivity studies. However, these sensitivity studies are computationally taxing, requiring weeks to run. To address this issue, we have developed a surrogate model using Gaussian processes (GP) that can emulate the output of this model with relatively small amounts of data in a reduced amount of time (on the order of minutes). We demonstrate the effectiveness of our emulator using backward error analysis.

Plain Language Summary

The NASA Ames Mars Global Climate Model (MGCM) is a computer model that simulates the weather on Mars. The MGCM is used by NASA to help understand weather data collected from satellites and other sources. The model has many inputs, the correct values of which are unknown, so it is often run many times with varying input values. Each run of the MGCM takes a long time, so running enough times to gather all of the desired outputs can take weeks. To address this issue, we have developed a method to approximate the output of the MGCM through the use of a machine learning model. This model requires a relatively small amount of data to train and once trained can approximate the MGCM output accurately and very quickly.

1 Introduction

Global Climate Models (GCM) are paramount to our understanding of the different processes affecting the weather on Mars. Recently, one such model, the Legacy NASA Ames Global Climate model (MGCM) was released to the public. This model has been in steady use in climate studies at NASA for decades (Haberle et al., 2019; Bertrand et al., 2020), aiding in our understanding of Martian weather. For many studies, such as the sensitivity study referenced in (Bertrand et al., 2020), numerous runs of the model are required to gather adequate results. The model is fairly long-running code, requiring in excess of 40 hours to simulate a single Martian year on an Intel Xeon Gold processor, making it incredibly time consuming to perform studies requiring many simulation runs.

In order to address this issue and aid in future sensitivity studies, we have developed an approach for the emulation of the MGCM. This approach utilizes a machine learning technique called Gaussian Process (GP) emulation. Within the geophysics research community, GP models have successfully been applied to regression tasks in papers such as (Domingo et al., 2020; Sarkar et al., 2019; Kleiber et al., 2012). The method utilized in this study offers linear scaling of both time and space as the simulated time span increases. This is a vast improvement over standard GP emulation, for which the time and space requirements would instead grow cubically and quadratically, respectively.

To be useful in facilitating sensitivity studies, an emulator must be accurate as well as fast. In order to evaluate the accuracy of our tool, we utilize both forward and backward error. We demonstrate that our emulator results in a small forward error, and, using backward error analysis, we show that a run of our emulator is similar to a simulation run with just a slight variation in parameters. This variation is generally much smaller than the parameter variations used in the sensitivity study that guided our data collection, indicating that the emulator would be an appropriate replacement for additional costly simulation runs in such a study.

This paper is organized as follows. Necessary technical background is provided in section 2. The data utilized is described in section 3. The methodology and evaluation metrics are described in section 4. Results are presented in section 5. Finally, a conclusion and discussion is given in the section 6.

2 Background

Understanding this work requires a high-level familiarity with the global climate model that is used as the subject of study, including the model outputs and the dust scenario maps utilized by the model (discussed in section 2.1). A mathematical background for Gaussian Processes and details about the covariance functions used by Gaussian Processes (discussed in section 2.2) is also helpful.

2.1 NASA Ames Mars Global Climate Model

The Legacy NASA Ames Mars Global Climate Model, recently released for public use, simulates Martian weather based on a set of input parameters. This model essentially amounts to a discretized mathematical representation of the complex interactions of the different weather cycles and geothermal properties of the Martian climate. This section serves as a short introduction, and more details about the model may be found in (Haberle et al., 2019).

2.1.1 Overview of Model and Outputs

The model makes use of physics packages that have been developed and thoroughly tested over decades of use (Haberle et al., 2019). In order to properly model the weather, the topography and soil properties of Mars are loaded at runtime. The topography is represented with data derived from the Mars Orbiter Laser Altimiter topography, documented in Smith et al. (1999). The soil properties are derived from research at Oregon State University, documented in Tyler and Barnes (2014). Each of these are smoothed to the model's nominal resolution (which will be explained in the following paragraph) of 5° latitude by 6° longitude (Haberle et al., 2019). For the adiabatic processes of the model, a slightly modified version of the dynamical core described in Suarez and Takacs (1995) is used. In this modified version, the transport scheme is replaced with a Van Leer scheme described in Hourdin and Armengaud (1999) (Haberle et al., 2019).

The MGCM has a total of 34 outputs relating to the different weather cycles on Mars, including surface temperature, surface pressure, dust aerosol mass mixing ratio, and diabatic heating rate, which are computed once per time-step per point in the model's resolution. The MGCM has a spatial resolution of 5° × 6°, meaning that the Martian globe is split into a 36×60 grid of rectangles in which each rectangle represents an area equating to 5° latitude by 6° longitude. The MGCM models the vertical axes with 24 layers extending into the atmosphere and 40 beneath the surface (Haberle et al., 2019). The outputs at all of these points are written to disk at a nominal rate of once per 1.5 simulated hours, making for approximately 16 points in time per solar day on Mars (sol) for each location on the grid. Each model output is a tensor of rank four or five. The dimensions of the tensors represent sols, time-steps, latitudes, longitudes, and, if applicable, vertical axis layers extending either into the atmosphere or subsurface (Haberle et al., 2019). The model has 22 outputs of rank four and 12 outputs of rank five.

2.1.2 Model Parameters

The MGCM has numerous model input parameters that influence the model behavior. These parameters include toggleable options such as water latent heat effects, water cloud formation, and tracking of empirically observed dust opacities (explained

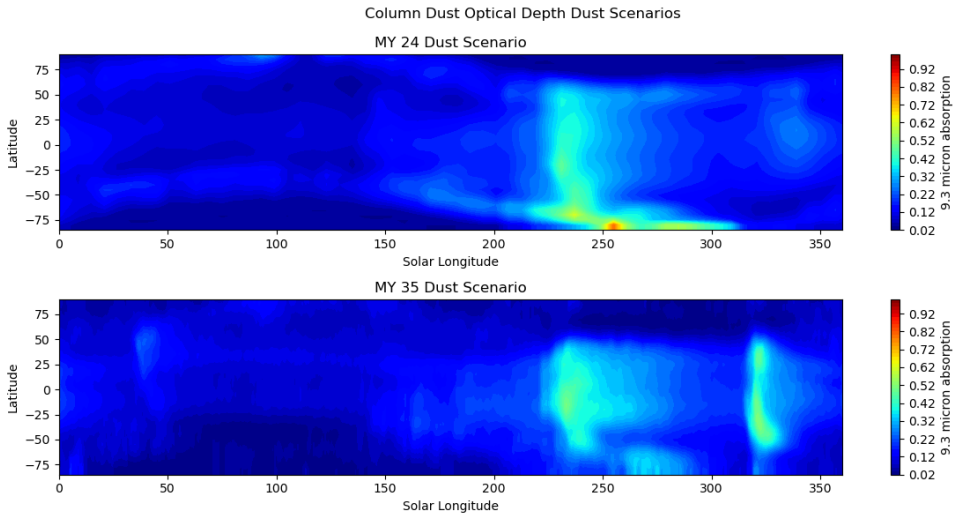

Figure 1. The MY 24 and MY 35 dust scenarios binned to a resolution of 5° latitude by 6° longitude by 6° Solar longitude. The graphs in this figure are averaged along the longitude dimension.

in-depth in the following section). Additionally, the model contains a range of floating point value attributes, including the albedo value of surface ice and other threshold values. Finally, there are a multitude of compile-time variables. For the purpose of our analysis, we vary the compile-time variable lifted dust effective radius, which controls the log-normal distribution of the lifted dust particle size in microns. We chose this variable to follow the sensitivity study performed in (Bertrand et al., 2020).

2.1.3 Dust Scenario Maps

Of interest to this study is the ability for the MGCM to be partially guided by empirical data (Haberle et al., 2019). The model can make use of daily gridded dust column opacity maps, which are based on empirical data collected from April 1999 to the present year. Montabone et al. (2015) documents the first eight; up to and including data collected in July 2013. These dust column opacity maps (dust scenarios) are denoted by the Martian year (MY) in which they occurred, beginning with MY 24 from data collected in 1999 (Montabone et al., 2015). The maps are linearly interpolated and re-binned to a temporal resolution of 6° Solar Longitude and the MGCM's spatial resolution (Haberle et al., 2019), as previously described. By tracking these dust scenarios over time, the MGCM is able to produce a more authentic dust lifting scheme (Haberle et al., 2019). These dust scenarios are toggled at runtime and specified as compile-time parameters. In this study, we use the MY 24 and MY 35 dust scenarios, which are depicted in fig. 1.

2.2 Gaussian Process Emulation

Gaussian processes (GPs) are general machine learning models that can be applied to a wide range of machine learning tasks. At a high level, a GP is the generalization of a multivariate Gaussian distribution into infinite dimensions (Seeger, 2004; Rasmussen & Williams, 2005). Similarly to a Gaussian distribution, which may be fully defined by its mean and a covariance matrix, a GP is fully defined by its mean function and covariance function (kernel) (Seeger, 2004). Although GPs can be applied to both regression and classification tasks with little modification, we solely utilize the regression function-

133 ality in this paper. Trained on a sample of data, a GP attempts to learn the underly-
134 ing distribution from which the data was generated. The ability for GPs to fit to a set
135 of data is influenced by the covariance function (explained in section 2.2.1), chosen based
136 on the known properties of the data, either *a priori* or empirically.

137 For all GPs in this work, the mean function is defined to be the mean of the train-
138 ing set, and the covariance function (described further in 2.2.1) is rational quadratic (RQ).
139 This can be stated mathematically as

$$f(\boldsymbol{X}) \sim \mathcal{GP}(\mu, RQ(\boldsymbol{r})),$$

140 where μ is the constant function whose value is the mean of the training data, $a \sim b$
141 denotes a is distributed according to b, $\boldsymbol{X}$ is the input matrix, $\boldsymbol{r}$ is the Euclidean dis-
142 tance of each pairwise combination of row vectors of $\boldsymbol{X}$, and f is the process that gen-
143 erated the training data.

144 ### *2.2.1 Covariance Functions*

145 The choice of covariance function, and the covariance matrix resulting from this
146 choice, has a considerable effect on the behavior of the resulting GP (Seeger, 2004; Ras-
147 mussen & Williams, 2005). Properties of the regression such as stationarity and peri-
148 odicity are dictated by the choice of covariance function (Seeger, 2004). For the purpose
149 of this study, we focus on the Rational Quadratic (RQ) covariance function (kernel), which
150 may be seen as an infinite sum of the Squared Exponential (SE) kernel, which itself may
151 be seen as an infinitely smooth Matérn kernel (Duvenaud, 2014; Rasmussen & Williams,
152 2005).

The Matérn kernel is defined by

$$\text{Matérn}(r) = \frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2v}r}{l} \right)^{v} K_v \left(\frac{\sqrt{2v}r}{l} \right), \tag{1}$$

153 where K_v is a modified Bessel function, $v, w \in \mathbb{R}_{>0}$, and Γ is the gamma function (Rasmussen
154 & Williams, 2005; Duvenaud, 2014; Abramowitz & Stegun, 1964). Note that the param-
155 eter v controls the smoothness of the function (Duvenaud, 2014). By letting v tend to
156 infinity as shown in Rasmussen and Williams (2005), we obtain the SE kernel, defined
157 as

$$\text{SE}(r) = e^{-\frac{r^2}{2l^2}}. \tag{2}$$

158 The RQ kernel is defined as

$$\text{RQ}(r) = \left(1 + \frac{r^2}{2\alpha^2} \right)^{-\alpha}, \tag{3}$$

159 where $\alpha \in \mathbb{R}_{>0}$ determines the shape of the gamma distribution for the length-scale
160 mixture comprising the kernel (Rasmussen & Williams, 2005). For each of the functions
161 defined in Eqs. [1-3], the input parameter r is the Euclidean distance between variable
162 pairs (Rasmussen & Williams, 2005; Duvenaud, 2014). Figure 2 shows a sampling of the
163 prior distribution for each of these kernels in the order they were introduced.

164 We experimented with a wide range of kernels and kernel combinations on a sam-
165 ple of the MGCM output and ultimately found the RQ kernel to be the best choice in
166 the majority of situations and, as such, we limit our presentation to the RQ kernel for
167 the remainder of this work. The GP model algorithm and covariance functions used in
168 this paper are provided by Pedregosa et al. (2011).

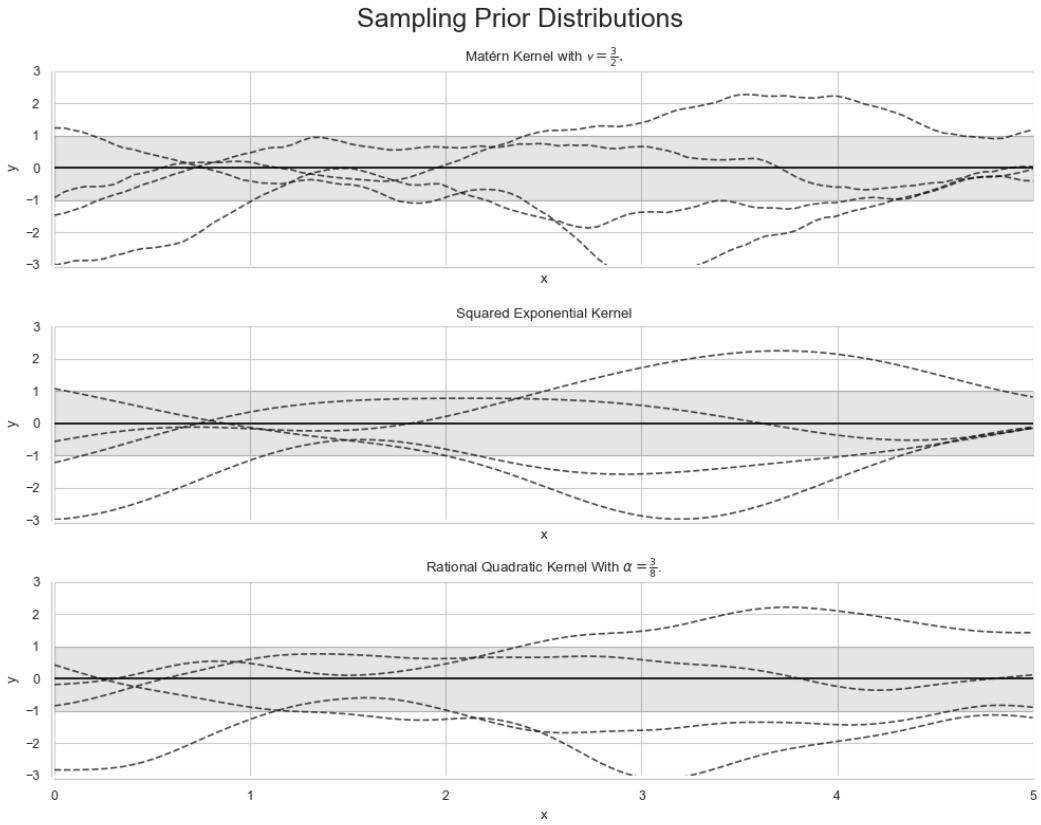

Figure 2. A sampling of five functions from the prior distributions of the Matérn, Squared Exponential, and Rational Quadratic kernels. The grey error bar shows the standard deviation with mean centered at zero. The $v = \frac{3}{2}$ value for the Matérn kernel was chosen for its relevance in practical ML applications whereas $\alpha = \frac{3}{8}$ for Rational Quadratic is arbitrarily chosen to display differences between itself and Squared Exponential. Shown together, the similarities and differences between the three kernels become more apparent.

3 Methodology

Creating an emulator for use in a sensitivity study requires forming a data set of simulation runs for the emulator to train on, training the emulator using this data, and inferring values of interest from the trained emulator. Here and elsewhere throughout this work, a "simulation" refers to a run of the actual MGCM code with a particular set of parameters, and an "emulation" is the output predicted by our GP emulator.

In general, a sensitivity study on a model such as the MGCM will vary one parameter while leaving others fixed in order to measure the effect of that parameter on the model output. The study we follow (Bertrand et al., 2020) varied the parameter lifted dust effective radius from 0.50 to 5.00 microns using a step-size of 0.5, requiring 10 simulations. For our emulator to be helpful, it must be able to train effectively on fewer runs than were required for the original study. We chose to train our emulator on data from simulation runs using 5 different values of lifted dust effective radius, or half of those required in the original study. Also, each dust scenario requires training a different emulator. We ran the simulations for 700 sols, which is slightly more than one Martian year.

The output of each simulation is originally 70 unformatted Fortran binary files. These files are then converted, with tools provided by the NASA Ames team (M. Kahre & Kling, 2021), to a single NetCDF file in a format adopted from the GFDL Finite-Volume Cubed-Sphere Dynamical Core (Harris et al., 2021).

3.1 Data Pre-processing

The data are first pre-processed by taking a 5-day average at each of the 16 daily time-steps of the model. Thus, for a simulation run of 700 Martian days (sol) (700×16 points in time), the resulting binned average will contain 140×16 points in time. Averaging the data in this manner was performed using publicly available tools provided by the NASA Ames team (M. Kahre & Kling, 2021) and has been utilized in papers such as (Bertrand et al., 2020). This has the added benefit of a reduction in size of each dataset by a factor of 5 from the original 3.3 terabytes.

3.2 Gaussian Process Splitting

One major limitation of GP emulation is the cubic-scaling time complexity. Numerous approximation methods have been proposed in the literature which aim to solve this issue, such as the methods discussed in (Lawrence et al., 2002; Csató & Opper, 2002; Tipping, 2001). Terry and Choe (2021) proposed a non-approximate method to alleviate the time complexity problem by recursively splitting the data using principal component analysis, which has shown success within the domains it was tested on.

In this work, we solve the time-complexity issue in a simple but effective manner: naively splitting the data and training a group of GPs whose output is then stitched together. This data splitting is performed such that each GP in the group learns the entire longitude dimension. For each of the remaining dimensions relating to sols, time-step, and latitude, the GP trains on a single value.

For the purpose of this study, each GP is also trained on the 5 equidistant values of lifted dust effective radius referenced above. Thus, each GP is trained on 300 points of data (60 longitude values × 5 lifted dust effective radius values) *irrespective of the length of the simulation*. Longer simulations simply require more GPs. By increasing the number of GPs rather than the size of any individual GP, the cost of training the emulator grows in a linear rather than cubic fashion as the duration of the simulation increases. The linear, $O(n)$, versus cubic, $O(n^3)$, time complexity difference is visualized in fig. 3.

It is worthwhile to contrast the training data input to each GP in the group against the training data that would be input to a single GP emulator responsible for the entire simulation. We demonstrate by a limited example. For the sake of brevity, suppose that we have two reference points of lifted dust effective radius that we wish to train on: 0.5 and 1.5 microns. Also, suppose that we are modeling 700 Martian sols (which is 140 time bins). A single GP training on all of the data in this scenario would have the input training matrix

$$\begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5 & 0 & 0 & 0 & 59 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5 & 139 & 15 & 35 & 59 \\ 1.5 & 0 & 0 & 0 & 0 \\ 1.5 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1.5 & 139 & 15 & 35 & 59 \end{bmatrix},$$

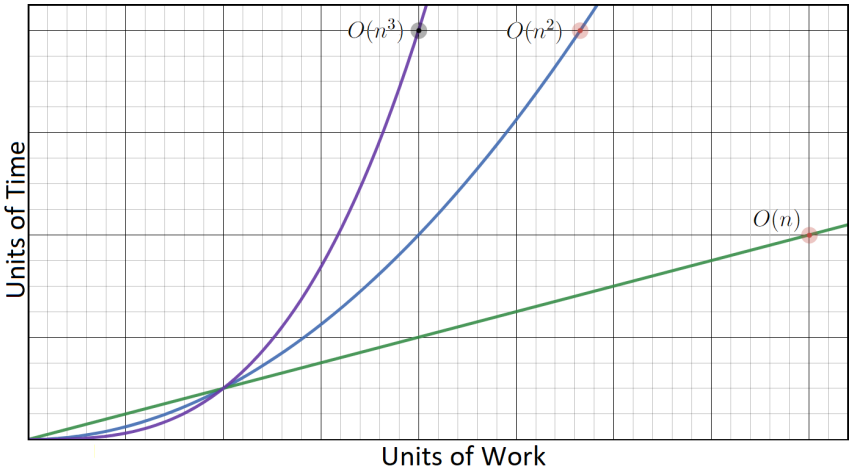

Figure 3. Illustration of growth of differing orders of time complexity. In general, the time complexity scaling of Gaussian Process models during training is cubic due to the matrix inversion used during training. During inferring, the time complexity scales quadratically. Due to our splitting, each Gaussian Process trains on a constant amount of data, so the scaling of the emulator as a whole is linear.

where the columns represent the lifted dust effective radius, index of the day (or binned day), index of the time-step, latitude, and longitude, respectively. This matrix contains 9,676,800 rows, which makes it far too large to use to train a GP. We thus see that without splitting, the training data consists of the Cartesian product of each dimension used, so adding training dimensions can quickly increase the amount of training data to an unreasonable size.

In the case of using a group, all columns but the first and last would be redundant since each individual GP is training on only a single sol, time-step, and latitude. The input training matrix of the individual GPs is reduced to

$$\begin{bmatrix} 0.5 & 0 \\ 0.5 & 1 \\ 0.5 & 2 \\ \vdots & \vdots \\ 0.5 & 59 \\ 1.5 & 0 \\ 1.5 & 1 \\ 1.5 & 2 \\ \vdots & \vdots \\ 1.5 & 59 \end{bmatrix}.$$

This matrix would have just 120 rows. The resulting emulator would consist of 80,640 (= 140 bins $\times$16 time steps $\times$36 latitudes) such GPs, each with a training matrix of 120 rows. As expected, this results in the same total amount of training data, but using many smaller matrices is far more manageable due to the fact that training time scales cubically with the size of the training matrix.

4 Evaluation Metrics

To evaluate the accuracy of our emulator, we utilize two types of evaluation metrics: forward error and backward error. Below, we first discuss the forward error met-

rics used to compare the simulator and emulator, which should be familiar to most readers. We then describe the general concept of backward error, how it is being applied in the context of this paper, and how to interpret our results using backward error analysis. Although it is likely less familiar to some readers, we believe backward error to be the more important metric for gauging the usefulness of our emulator for our expected use cases.

4.1 Forward Error

Forward error directly compares the output of the emulator at a particular parameter value against the output of the simulator at that same parameter value. For forward error, both the mean squared error (MSE) and R^2 values are presented. The MSE quantifies how far the approximate values are from the true values. Specifically, it quantifies the error by computing the average of the squares of the forward error of a regression. This weights large outliers more heavily due to the presence of squaring in the calculation. The MSE is calculated as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

where n is the number of data points, Y_i are the true values (i.e., those given by the MGCM), and $\hat{Y}_i$ are the emulated values.

The coefficient of determination, denoted as R^2, is a statistical measure of the amount of explained variance from the true values compared to the estimated values. At a high level, this value provides an easily understandable metric for the goodness of fit of a regression. This value can range from between 0 and 1, where an R^2 of 1 means that 100% of the variance of the true values are captured by the regression. The R^2 value is calculated as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2},$$

where $\bar{Y}$ is the arithmetic mean of the true values and the other variables are defined as above for MSE.

4.2 Backward Error

Backward error analysis is another approach to quantifying the error of approximation methods and utilizes the assumption that the computed approximate solution is the exact solution to a nearby problem. Backward error analysis can be helpful when there is uncertainty in the inputs to a model. In our case, the true lifted dust effective radius on Mars is unknown. If a researcher is interested in the simulation output of the MGCM at a particular lifted dust effective radius, an emulator that is "correct" for a similar lifted dust effective radius could be in effect just as useful as the simulation at that exact value. In general, if an emulated output is closer to the solution than a simulated output that is some ϵ away, the backward error is less than ϵ.

We give a visual demonstration of backward error in fig. 4. In that figure, the blue points represent the forward error between the simulated ("true solution") computed at a lifted dust effective radius value of 2.65 microns against the simulated solution computed at corresponding lifted dust effective radius value indicated on the x-axis. The red line represents the forward error between the emulated solution at a lifted dust effective radius of 2.65 microns against the simulated solution at a lifted dust effective radius of 2.65 microns. Note that the vertical axis is logarithmic. Continuing with fig. 4, in graphs such as the left one, the blue dots do not cross the red line, indicating that the backward error is some amount less than 0.05 (we would need finer granularity to set a lower bound)

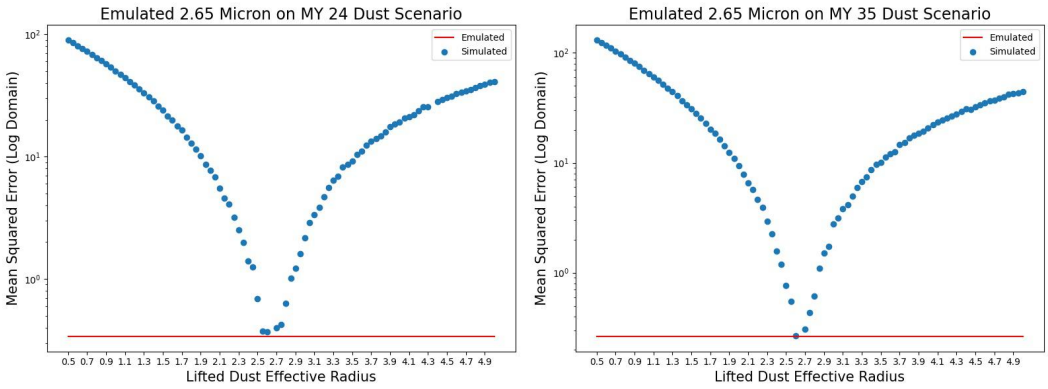

Figure 4. Example illustration of backward error analysis. For a given simulation with input lifted dust effective radius parameter value (2.65 microns), we compute the difference between the output result (in this case surface temperature) for a simulation with an input value of 2.65 microns and simulations with nearby input values, represented by the blue dots. The red line represents the difference between the emulated output value and simulated output value, both for the input parameter of 2.65 microns. The right graph demonstrates a backward error of 0.05 as the nearest input value above the red line was 0.05 away from 2.65. The left graph, with no points below the red line, indicates the backward error is smaller than granularity of this analysis.

as the difference between the emulated and simulated outputs are less than the difference in simulation outputs for an extremely close input parameter. In the right graph in fig. 4, we see that the blue dots cross the red line. In this case, the backward error is given by the difference between the input value in question (2.65) and the nearest input value that produced a blue dot above the line (2.7 in this example). Thus, we can state that the backward error is 0.05.

For the remainder of this paper, the backward error plots will be condensed in order to display as much information as possible. These condensed plots will show the emulated lifted dust effective radius value along the horizontal axis while the vertical axis represents the backward error at that value. Note that a backward error of less than our granularity of 0.05 is represented on the graph as 0.05.

5 Results

In this section, we first present the emulator results when evaluated with forward error, then results using backward error analysis. Finally, the timing results of the emulator are compared against the MGCM simulator.

For the purpose of this comparison, we have created a large set of data consisting of simulation runs, far more than would be required simply to create the emulators. We created this data set by varying the lifted dust effective radius from 0.50 to 5.00 microns using a step-size of 0.05 and running the simulation for 700 sols for each value of the lifted dust effective radius parameter. This was done separately for both the MY 24 and MY 35 dust scenarios. It should be noted that the MGCM was unable to perform the entire simulation when the lifted dust effective radius was set to 4.35 microns on the MY 24 dust scenario; this value has been omitted from the MY 24 dataset.

5.1 Forward Error Results

Figure 5 shows a representative slice at a specific longitude of the accuracy of the GP group when applied to the MY 24 map temperature output. The emulated output is shown in the top pane, the MGCM output in the middle pane, and the forward error in the bottom pane. Looking at the emulator and MGCM output, it is difficult to discern the difference between the MGCM output and the emulated output, though the emulated output is marginally more smoothed in the areas corresponding to the error in the third pane. The MSE is 3.97 while R^2 is 99.80%. Examining the third pane on its own, we see that the vast majority of the domain is white (indicating effectively zero error), and the error, indicated by the shaded spots, is extremely localized to a few spots along the edge of the curve present in the solution.

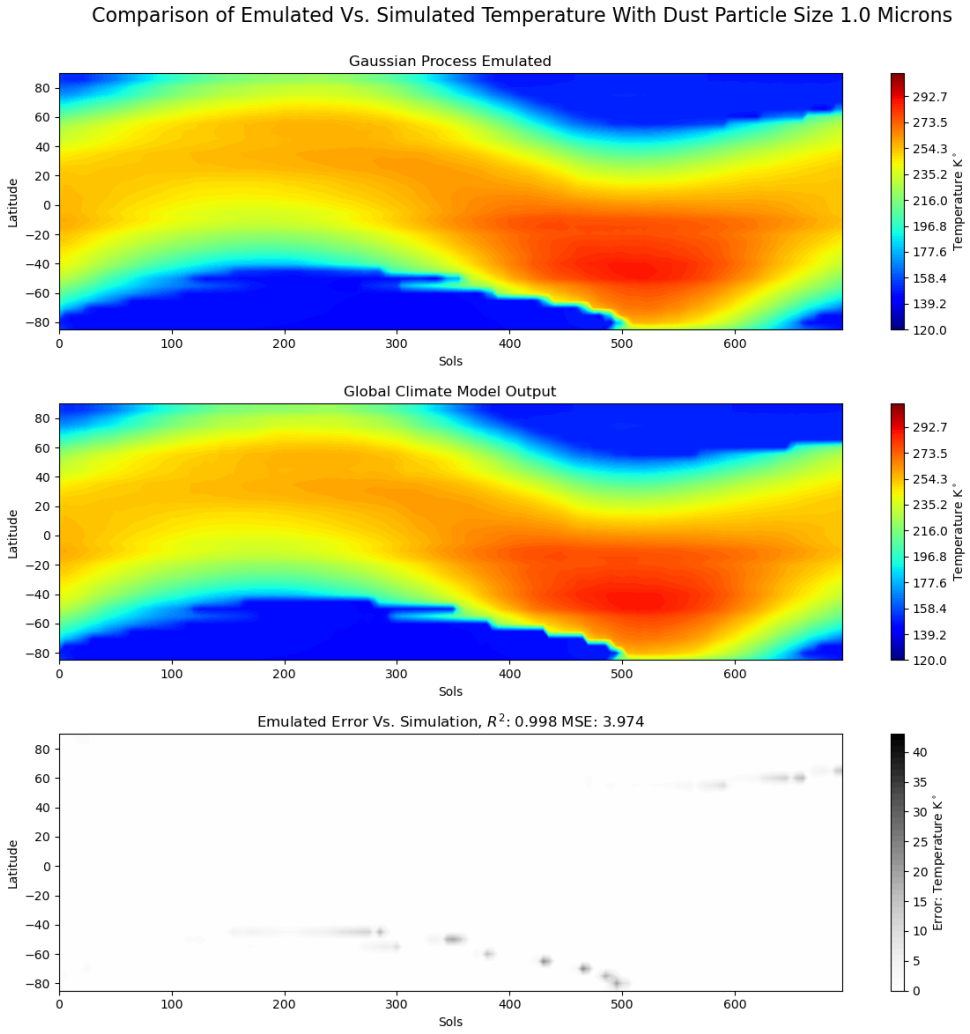

Figure 5. Performed on the temperature model output for the MY24 dust scenario. The forward error of the Gaussian Process group compared against the model output at 78° longitude when lifted dust effective radius is set to 1.0 micron. The top pane shows the GP emulated output, the middle pane shows the MGCM output, and the bottom plane shows only the forward error between the previous two plots.

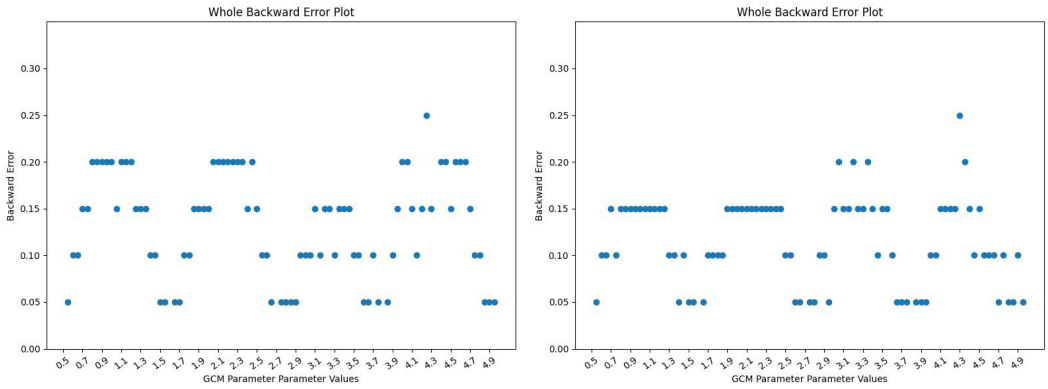

Figure 6. The backward error of the emulation when applied to the surface temperature MGCM output. MY 24 and MY 35 are shown on the left and right respectively.

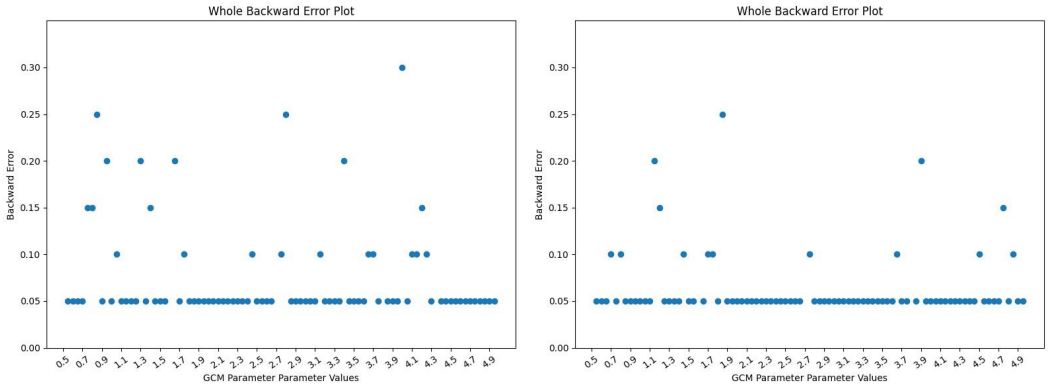

Figure 7. The backward error of the emulation when applied to the surface pressure MGCM output. MY 24 and MY 35 are shown on the left and right respectively.

5.2 Backward Error Results

Figure 6, which displays the backward error for surface temperature, is representative of the error on the majority of MGCM outputs. For both the MY 24 and MY 35 dust scenarios, the worst backward error was 0.30 microns. Figure 7 shows the backward error for the surface temperature output. Similarly to surface pressure, the worst backward error was 0.30 microns, and in this case, most of the values were significantly lower.

Note that the worst-case backward error observed is 0.20 microns less than the step-size used in the sensitivity study referenced earlier. This demonstrates that the emulator is effective for our stated goal of use in sensitivity studies. In such studies, the purpose is to find the overall behavior of the MGCM simulator for a range of parameter values. With the backward error of the emulator being less than the granularity used in the studies, the emulator outputs capture the overall behavior as effectively as the simulations that are being replaced.

For all tested outputs except for surface opacity, these graphs are highly representative of the results. For the sake of completeness, fig. 8 depicts the backward error for surface opacity (note the different vertical axis from the previous graphs). This graph shows that the maximum backward error was 0.60 and 0.40 for the MY 24 and MY 35 dust scenarios respectively. This amounts to a backward error that is marginally worse

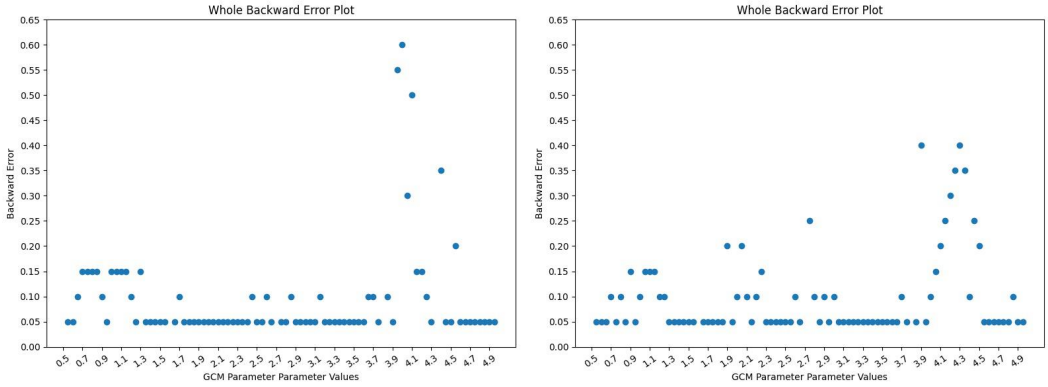

Figure 8. The backward error of the emulation when applied to the surface opacity MGCM output. MY 24 and MY 35 are shown on the left and right respectively.

Table 1. Computation Times for Gaussian Process Emulator

Emulator Training Time	Emulator Inference Time	MGCM Simulation Time
87.63 (minutes)	0.116 (minutes)	2404.8 (minutes)

[a]MGCM simulation time is included for comparison purposes.

than the step-size used in the Bertrand et al. (2020) paper, but only only for very few parameter values for the surface opacity output.

Across all figs. 6 to 8, the emulator error displays a sinusoidal-like pattern. This behavior is expected and stems from the points at which the emulator is trained – the emulator error is lowest near the training points. Thus, if a researcher is particularly interested in reducing the error of the emulator along a specific interval, the number of nearby training points can be increased as needed.

5.3 Timing Results

Finally, we compare the timing results of the emulator against the simulation. The execution time required for both the existing simulator and both training and inference of our Gaussian Process emulator are shown in table 1. It should be noted that after training the emulator ($\approx$ 87 minutes), any number of points within the input range of lifted dust effective radius can be inferred with no further training required. As an example, if the model is trained on 0.5 and 5.0 microns, the trained emulator can infer the model output given any value between 0.5 and 5.0, with each inference taking only seconds. For studies, such as sensitivity studies, that require the output value at a wide variety of lifted dust effective radius inputs, this results in an emulator that is a significant speedup over the MGCM simulation. For instance, the MGCM sensitivity study we reference uses 10 evenly-spaced lifted dust effective radius values which would take on average

$$2404.8 * 10/60 \approx 400 \text{ hours}$$

to run. In contrast, the emulator, including the time to run the original simulation to generate the 5 training points, train the emulator, and infer on the 10 lifted dust effective radius values, would require

$$(2404.8 * 5 + 87.63 + .116 * 10)/60 \approx 202 \text{ hours},$$

cutting the required time approximately in half.

6 Conclusion and Discussion

In this work, we presented a method of fast Gaussian-process emulation and applied it to the NASA Ames MGCM. For the vast majority of simulation outputs, the emulator produced a result with more fine-grained accuracy than was utilized in past sensitivity studies performed on the MGCM. This shows that GP emulation can be as effective as the current simulation-only approach when used for sensitivity studies. Furthermore, the use of GP emulation substantially decreased runtimes. The demonstrated combination of speed and accuracy shows that emulating the model output in this manner can be an effective tool for future sensitivity studies.

One consideration with the presented GP approach is that a different emulator must be trained for each output. However, the most time-consuming component of the emulator training is generating the simulation data for the training points, which does not need to be repeated for each output. Furthermore, the trained state of the emulators can be saved, so future users interested in different points in the same range would require only the inference time, taking just seconds.

Regarding future work, we note that a new version of the MGCM, which has not been released for public use, is under development at the NASA Ames Research Center (Haberle et al., 2019; M. A. Kahre et al., 2018). This new model uses many of the same physics packages that are used in the MGCM, though they have been modularized (Haberle et al., 2019). Based on the publicly released details of the new model (Haberle et al., 2019; M. A. Kahre et al., 2018), we expect the emulator described in this work to apply equally as well to this new model with little or no change to the emulator code. Although the new model is stated to be considerably faster than the MGCM, our emulation process is expected to still be highly advantageous because any speed increases in the MGCM would reduce the time to generate training data for the emulator, which is the most costly part of emulation. Regardless of how fast the new model is, it is unlikely to match the extremely fast time of GP training and inference once the training data has been generated. Once the new MGCM is publicly released by the NASA Ames team, we plan to validate the efficacy of our findings on the new model.

Finally, while only applied here to the MGCM, this idea should be applicable to many other global climate models that use the same output format. For both this case and the case of the new MGCM, the emulator code as released for this paper should nearly be a drop-in replacement.

7 Open Research

The MY24 and MY35 map data used for the simulation run inputs and the Python scripts used in this research are available on Zenodo (DOI: 10.5281/zenodo.7295469, https://zenodo.org/badge/latestdoi/562292845) (Tunnell et al., 2022). Specifically, this includes the versions of the scripts used to 1) run the simulations that generate the data the Gaussian Processes are trained on and 2) train the actual Gaussian Processes emulator and infer output, which are available open-source under the MIT License. In addition to being preserved on Zenodo, ongoing development of these scripts is available openly at https://github.com/tunnellm/MGCM_Public.

Acknowledgments

We would like to express our deepest appreciation to Dr. Kahre and the rest of the NASA Ames team for their patience and help with setting up and working with their climate model. This research made use of Clipper, the high-performance computing cluster at Grand Valley State University. This work was funded in part by the Kindschi Research Fellowship at Grand Valley State University.

References

Abramowitz, M., & Stegun, I. A. (1964). *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* New York: Dover Publications.

Bertrand, T., Wilson, R. J., Kahre, M. A., Urata, R., & Kling, A. (2020, jun). Simulation of the 2018 global dust storm on mars using the NASA ames mars GCM: A multitracer approach. *Journal of Geophysical Research: Planets*, *125*(7). Retrieved from https://doi.org/10.1029%2F2019je006122 doi: 10.1029/2019je006122

Csató, L., & Opper, M. (2002). Sparse on-line gaussian processes. *Neural computation*, *14*(3), 641–668.

Domingo, D., Malmierca-Vallet, I., Sime, L., Voss, J., & Capron, E. (2020). Using ice cores and gaussian process emulation to recover changes in the greenland ice sheet during the last interglacial. *Journal of Geophysical Research: Earth Surface*, *125*(5), e2019JF005237. Retrieved from https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019JF005237 (e2019JF005237 10.1029/2019JF005237) doi: https://doi.org/10.1029/2019JF005237

Duvenaud, D. (2014). *Automatic model construction with gaussian processes* (Doctoral dissertation). doi: https://doi.org/10.17863/CAM.14087

Haberle, R. M., Kahre, M. A., Hollingsworth, J. L., Montmessin, F., Wilson, R. J., Urata, R. A., ... Schaeffer, J. R. (2019). Documentation of the nasa/ames legacy mars global climate model: Simulations of the present seasonal water cycle. *Icarus*, *333*, 130-164. Retrieved from https://www.sciencedirect.com/science/article/pii/S0019103518305761 doi: https://doi.org/10.1016/j.icarus.2019.03.026

Harris, L., Chen, X., Putman, W., Zhou, L., & Chen, J.-H. (2021). *A scientific description of the gfdl finite-volume cubed-sphere dynamical core.* Retrieved from https://repository.library.noaa.gov/view/noaa/30725 (Technical Memorandum)

Hourdin, F., & Armengaud, A. (1999). The use of finite-volume methods for atmospheric advection of trace species. part i: Test of various formulations in a general circulation model. *Monthly Weather Review*, *127*(5), 822 - 837. Retrieved from https://journals.ametsoc.org/view/journals/mwre/127/5/1520-0493_1999_127_0822_tuofvm_2.0.co_2.xml doi: 10.1175/1520-0493(1999)127⟨0822:TUOFVM⟩2.0.CO;2

Kahre, M., & Kling, A. (2021, Aug). *Mars global climate model (gcm) tutorial.* NASA. Retrieved from https://www.nasa.gov/mars-climate-modeling-center-ames/MarsGlobalClimateModelTutorial

Kahre, M. A., Wilson, R. J., Hollingsworth, J. L., andA. S. Brecht, R. M. H., & Urata, R. A. (2018). *High resolution modeling of the dust and water cycles with the nasa ames mars global climate model* (Tech. Rep.).

Kleiber, W., Katz, R. W., & Rajagopalan, B. (2012). Daily spatiotemporal precipitation simulation using latent and transformed gaussian processes. *Water Resources Research*, *48*(1). Retrieved from https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011WR011105 doi: https://doi.org/10.1029/2011WR011105

Lawrence, N., Seeger, M., & Herbrich, R. (2002). Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, *15*.

Montabone, L., Forget, F., Millour, E., Wilson, R., Lewis, S., Cantor, B., ... Wolff, M. (2015). Eight-year climatology of dust optical depth on mars. *Icarus*, *251*, 65-95. Retrieved from https://www.sciencedirect.com/science/article/pii/S0019103515000044 (Dynamic Mars) doi: https://doi.org/10.1016/j.icarus.2014.12.034

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ...

Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning (adaptive computation and machine learning)*. The MIT Press.

Sarkar, D., Osborne, M. A., & Adcock, T. A. A. (2019). Spatiotemporal prediction of tidal currents using gaussian processes. *Journal of Geophysical Research: Oceans*, *124*(4), 2697-2715. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018JC014471` doi: https://doi.org/10.1029/2018JC014471

Seeger, M. (2004). Gaussian processes for machine learning. *International Journal of Neural Systems*, *14*(02), 69-106. Retrieved from `https://doi.org/10.1142/S0129065704001899` (PMID: 15112367) doi: 10.1142/S0129065704001899

Smith, D. E., Zuber, M. T., Solomon, S. C., Phillips, R. J., Head, J. W., Garvin, J. B., ... Duxbury, T. C. (1999). The global topography of mars and implications for surface evolution. *Science*, *284*(5419), 1495-1503. Retrieved from `https://www.science.org/doi/abs/10.1126/science.284.5419.1495` doi: 10.1126/science.284.5419.1495

Suarez, M. J., & Takacs, L. L. (1995). *Technical report series on global modeling and data assimilation. volume 5: Documentation of the aires/geos dynamical core, version 2* (Tech. Rep.).

Terry, N., & Choe, Y. (2021, 08). Splitting gaussian processes for computationally-efficient regression. *PLOS ONE*, *16*(8), 1-17. Retrieved from `https://doi.org/10.1371/journal.pone.0256470` doi: 10.1371/journal.pone.0256470

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, *1*(Jun), 211–244.

Tunnell, M., Bowman, N., & Carrier, E. (2022, November). *tunnellm/mgcm_public: Public [software]*. Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.7295469` doi: 10.5281/zenodo.7295469

Tyler, D., & Barnes, J. R. (2014). Atmospheric mesoscale modeling of water and clouds during northern summer on mars. *Icarus*, *237*, 388-414. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0019103514002127` doi: https://doi.org/10.1016/j.icarus.2014.04.020