Appendix

Appendix A

Gaussian Identities

In this appendix we provide basic background on each topic to make this thesis self-contained. Let $\boldsymbol{x} = \{x_1, \ldots, x_i, \ldots, x_N\}$ be a set of N scalar Gaussian random variables $x_i \in \mathcal{R}$. Such that:

$$x_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{A.1}$$

Here, μ_i represents the scalar mean, while σ_i^2 represents the variance. Their probability density can thus written as:

$$\begin{aligned} \Pr[x_i] &= \mathcal{N}(\mu_i, \sigma_i^2) && \text{(A.2)} \\ &= \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp^{-\frac{(x_i-\mu_i)^2}{2\sigma_n^2}} && \text{(A.3)} \end{aligned}$$

Let us partition the set $\boldsymbol{x}$ into two sets, $\boldsymbol{a}$ and $\boldsymbol{b}$ of size N_a and N_b respectively. Such that $\boldsymbol{a} \cup \boldsymbol{b} = \boldsymbol{x}$, while each set may contain one or more variables. We denote $\boldsymbol{x_a}$ as the ordered collection of random variables in set $\boldsymbol{a}$, thus $\boldsymbol{x_a} \in \mathcal{R}^{N_a}$ is a collection of random variables. Thereby, the probability distribution of random vector $\boldsymbol{x_a}$ can be denotes as equation A.4 and similarly for $\boldsymbol{x_b}$.

$$\begin{aligned} \Pr[\boldsymbol{x_a}] &= \mathcal{N}(\boldsymbol{\mu_a}, \boldsymbol{\Sigma_a}) && \text{(A.4)} \\ &= \frac{1}{\sqrt{(2\pi)^{N_a}\boldsymbol{\Sigma_a}}} exp\left[-\frac{1}{2}(\boldsymbol{x_a} - \boldsymbol{\mu_a})^T \boldsymbol{\Sigma_a}(\boldsymbol{x_a} - \boldsymbol{\mu_a})\right] && \text{(A.5)} \end{aligned}$$

Here, $\boldsymbol{\mu_a}$ is the ordered collection of means in the random variable set $\boldsymbol{a}$, while $\boldsymbol{\Sigma_a}$ represents the covariance matrix. We now describe the joint, marginal and conditional distributions which are also Gaussian distributions, for a detailed treatment refer to [Bishop 2006].

A.1 Joint distribution

If the probability densities of sets $\boldsymbol{a}$ and $\boldsymbol{b}$ can be written as, $\Pr[\boldsymbol{x_a}] = \mathcal{N}(\boldsymbol{\mu_a}, \boldsymbol{\Sigma_a})$ and $\Pr[\boldsymbol{x_b}] = \mathcal{N}(\boldsymbol{\mu_b}, \boldsymbol{\Sigma_b})$ respectively. Then the joint distribution between two random vectors $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$ is given by $\Pr[\boldsymbol{x_a}, \boldsymbol{x_b}]$ in equation A.6 ($\boldsymbol{a} \cup \boldsymbol{b} = \boldsymbol{x}$).

$$\Pr[\boldsymbol{x}] = \Pr[\boldsymbol{x_a}, \boldsymbol{x_b}] = \Pr \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x_b} \end{bmatrix} \tag{A.6}$$

$$= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu_b} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_a & \boldsymbol{\Sigma_{ab}} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma_b} \end{bmatrix} \right) \tag{A.7}$$

A.2 Sum of Gaussians

If the size of the random vectors $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$ is same, then if a new vector $\boldsymbol{x_c} = \boldsymbol{x_a} + \boldsymbol{x_b}$ is sum of two Gaussians, it is also a multi-variate Gaussian with probability density given as follows:

$$\Pr[\boldsymbol{x_a}] = \mathcal{N}(\boldsymbol{\mu_a} + \boldsymbol{\mu_b}, \boldsymbol{\Sigma_a} + \boldsymbol{\Sigma_b}) \tag{A.8}$$

A.3 Affine property of Gaussians

If $\boldsymbol{x_d}$ is a linear transformation of $\boldsymbol{x_a}$ such that $\boldsymbol{x_c} = \boldsymbol{l} + \boldsymbol{M}\boldsymbol{x_a}$. Here, $\boldsymbol{l}$ and $\boldsymbol{M}$ are a vector and a matrix of constants. Then $\boldsymbol{x_d}$ it is also a multi-variate Gaussian with probability density given as follows:

$$\Pr[\boldsymbol{x_d}] = \mathcal{N}(\boldsymbol{l} + \boldsymbol{M}\boldsymbol{\mu_a}, \boldsymbol{M}\boldsymbol{\Sigma}_a\boldsymbol{M}^T) \tag{A.9}$$

In fact the affine property is valid for any linear transformation of a Gaussian distribution. This is the same property that we use to enforce relationships defined by linear operations between multiple outputs.

A.4 Marginal Distribution

The marginal distribution between two probability distributions is given as:

$$\Pr[\boldsymbol{x_a}] = \int \Pr[\boldsymbol{x_a}, \boldsymbol{x_b}] d\boldsymbol{x_b} \tag{A.10}$$

We are effectively integrating out the random variable $\boldsymbol{x_b}$. This is the same principle while calculating the ‘marginal likelihood’ to choose hyper-parameters, we integrate out the latent functions.

For a Gaussian random vector (multi-variate Gaussian) as described in equation A.6, the marginal distribution is a Gaussian. This is also called the marginalization property of Gaussians.

$$\begin{aligned} \Pr[\boldsymbol{x_a}] &= \int \Pr[\boldsymbol{x_a}, \boldsymbol{x_b}] d\boldsymbol{x_b} && \text{(A.11)} \\ &= \mathcal{N}(\boldsymbol{\mu_a}, \boldsymbol{\Sigma_a}) && \text{(A.12)} \end{aligned}$$

A.5 Conditional Distribution

The probability of $\boldsymbol{x_a}$ given a value of $\boldsymbol{x_b} = \bar{\boldsymbol{x_b}}$, is called as conditional probability and is written as $\Pr[\boldsymbol{x_a} \mid \boldsymbol{x_b} = \bar{\boldsymbol{x_b}}]$.

$$\Pr[\boldsymbol{x_a} \mid \boldsymbol{x_b} = \bar{\boldsymbol{x_b}}] = \frac{\Pr[\boldsymbol{x_a}, \boldsymbol{x_b}]}{\Pr[\boldsymbol{x_b} = \bar{\boldsymbol{x_b}}]} \quad \text{(A.13)}$$

For a Gaussian random vector (multi-variate Gaussian) as described in equation A.6, the conditional distribution is a Gaussian.

$$\begin{aligned} \Pr[\boldsymbol{x_a} \mid \boldsymbol{x_b} = \bar{\boldsymbol{x_b}}] &= \mathcal{N}(\boldsymbol{\mu_{a|b}}, \boldsymbol{\Sigma_{a|b}}) && \text{(A.14)} \\ \boldsymbol{\mu_{a|b}} &= \boldsymbol{\mu_a} + \boldsymbol{\Sigma_{ab}} \boldsymbol{\Sigma_b}^{-1} (\bar{\boldsymbol{x_b}} - \boldsymbol{\mu_b}) && \text{(A.15)} \end{aligned}$$

This is the same principle while calculating the posterior distribution of a GP, there we are conditioning a GP given an observation dataset.

Appendix B

Proper Orthogonal Decomposition for pressure snapshots

This appendix demonstrates how to perform Proper Orthogonal Interpolation (POD) based interpolation using pressure snapshots. There exist two types of POD a classical POD [Gurvich 1969] and snapshot POD [Romanowski 1996]. We will use the snapshot POD in this thesis, since it is a highly popular in aerodynamic interpolation use cases.

B.1 Pressure Snapshot

Let us first start by defining a pressure snapshot. There exists a 3 dimensional spatial vector $\omega_i \in \mathbb{R}^3$ such that $\omega_i = \{(\omega_i^1, \omega_i^2, \omega_i^3)\}$. Here, $i \in [1, N_{nodes}]$ are the spatial coordinates of the i^{th} pressure node in a mesh containing N_{nodes} pressure nodes. Similarly there exists a D dimensional parameter vector $d_j \in \mathbb{R}^D$, for $d_j = \{(d_j^1, d_j^2, \ldots, d_j^D)\}$. Here, $j \in [1, N_{parameter}]$ correspond to the j^{th} parameter set. The parameters can be any non-spatial parameter which are desired to be interpolated, some common examples include Mach, Angle of Attack for steady aerodynamics and time or frequency for unsteady aerodynamics. Since, this thesis relates to interpolating steady aerodynamics d_j will correspond to the j^{th} run in a total of $N_{parameter}$ simulations or experiments.

The pressure measured on the i^{th} pressure node for the j^{th} parameter set will be denoted as $p_j(\omega_i)$ defined by the equation B.1. We next define the matrix $\Omega = \{\omega_1; \omega_2; \ldots; \omega_{N_{nodes}}\}$ for $\Omega \in \mathbb{R}^{N_{nodes} \times 3}$ containing the full spatial information of the CFD mesh. Finally, the pressure snapshot for the CFD run j will be denoted as $P_j(\Omega) = \{p_j(\omega_1); p_j(\omega_2); \ldots; p_j(\omega_{N_{nodes}})\}$ for $P_j(\Omega) \in \mathbb{R}^{N_{nodes}}$ defined by the equation B.2.

$$p_j(\omega_i) = f_{pressure}(\omega_i, d_j) \tag{B.1}$$

$$P_j(\Omega) = f_{pressure}(\Omega, d_j) \tag{B.2}$$

B.2 POD for aerodynamic snapshots

The optimal POD basis vectors $\phi(\Omega)$ are chosen so that they maximize the cost described in equation B.3. $(\cdot,\cdot)$ is an inner product and $\langle\cdot,\cdot\rangle$ is the parameter-averaging operation [Berkooz 1993, Epureanuj 1999]. Solving this optimization problem leads to an eigen value problem, where $\phi(\Omega)$ are the eigen-vectors.

$$max_{\phi} \frac{\langle (P_j(\Omega), \phi(\Omega))^2 \rangle}{(\phi(\Omega), \phi(\Omega))} \tag{B.3}$$

The idea of snapshot POD is to write the eigen-vector $\phi(\Omega)$ in terms of the pressure snapshots (equation B.4) and not in terms of all inputs.

$$\phi^l(\Omega) = \sum_{j=1}^{N_{parameter}} \beta_j^l P_j(\Omega) \tag{B.4}$$

Here, $\phi^l(\Omega)$ is the l^{th} eigen-vector and the coefficients β_j^l can be shown to satisfy the eigen problem in equation B.5 and B.6.

$$R\beta = \Lambda\beta \tag{B.5}$$

here,

$$\beta = \begin{pmatrix} \beta^1 \\ \beta^2 \\ \dots \\ \beta^{N_{parameter}} \end{pmatrix} \quad and \quad R_{lm} = \frac{1}{N_{parameter}} (P_l(\Omega), P_m(\Omega)) \tag{B.6}$$

The matrix R is called the correlation matrix for $R \in \mathbb{R}^{N_{parameter} \times N_{parameter}}$. The eigen-vector $\beta^l \in \mathbb{R}^{1 \times N_{parameter}}$ represents the participation factors for equation B.5. We have hence calculated the eigen-vectors for our matrix of pressure snapshots.

The original pressure snapshots can be now be written as a linear combination of the eigen-vectors equation B.7.

$$P_j(\Omega) = \sum_{l=1}^{p} a^l(d_j)\phi^l(\Omega) \tag{B.7}$$

Here, $a^l(d_j)$ denotes the participation factor or amplitude for the mode l at a point d_j in the parameter space. Once we evaluate the eigen-vectors calculating the amplitudes become a task of solving equation B.8.

$$a^l(d_j) = (\phi_l(\Omega), P_j(\Omega)) \tag{B.8}$$

Note, further speed up in reconstruction of pressure snapshots can be gained by taking $p < N_{parameter}$. Only, taking into account the modes which correspond to the highest participation can significantly improve the reconstruction times [Tan 2003, Allemang 2011].

B.3 Interpolation

The above section describes how to calculate eigen-vectors from a set of pressure snapshots. Furthermore we can use equation B.7 and B.8 to reconstruct the initial snapshots. This section describes how to interpolate or predict the snapshots at an unknown point d_{new}in the parameter space.

If the participation factors $a^l(d_j)$ are smooth functions of parameters $d \in \mathbb{R}^D$, interpolation can be used to determine the participation factors at desired point d_{new}. In this thesis we use cubic spline [Bartels 1987] to interpolate the participation factors. Once the new participation factors are obtained a new snapshot can be constructed using the equation B.9. A pseudo-code is given in algorithm 1 of the full process.

$$P_{new}(\Omega) = \sum_{l=1}^{p} a^l(d_{new})\phi^l(\Omega) \tag{B.9}$$

Due to the reduced order modelling we are interpolating only the $N_{parameter}$ of the order of 10^2 parameters instead of N_{nodes} of the order of 10^4 pressure nodes. Note, for the reminder of this thesis we will use all the available modes for interpolating the snapshots i.e. $p = N_{parameter}$.

Algorithm 1: Algorithm for POD + Interpolation.

```
Data: $P_j(\Omega)$ (pressure snapshots),
      $d_j$ for $j \in [1 : N_{parameter}]$ (set of parameters),
      $d_{new}$ (desired point)
Result: $P_{new}(\Omega)$ (Interpolated pressure snapshot at point $d_{new}$)
pod()
    // Calculating eigen-vectors using equations B.6, B.5 and B.4
    $R_{lm} = \frac{1}{N_{parameter}}(P_l, P_m^*)$
    $R\beta = \Lambda\beta$
    $\phi^l(\Omega) = \sum_{j=1}^{N_{parameter}} \beta_j^l P_j(\Omega)$

    // Calculating participation factors of eigen-vector $l$ at point $d_j$
    $a_l(d_j) = (\phi_l(\Omega), P_j(\Omega))$
    return $\phi^l, \quad a^l(d_j) \quad for \quad l \in [1 : N_{parameter}]$
interpolate()
    // Interpolating the participation factor of the $l^{th}$ eigen-vector
    foreach $l \in [1 : N_{parameter}]$ do
        $a^l(d_{new}) = \textbf{spline}(\alpha_l(d_j), d_{new})$
    end
    // Reconstructing snapshot using the interpolated participation
       factors and original eigen-vectors
    $P_{new}(\Omega) = \sum_{l=1}^{p} a^l(d_{new})\phi^l()\Omega$
    return $P_{new}(\Omega)$
```

Appendix C

Gaussian Process Regression enforcing Non-linear operators

This chapter reproduces the proof of approximation performed when enforcing non-linear operators in an MTGP framework as provided by [Constantinescu 2013].

For simplicity let us take the case of an explicit relationship between two outputs y^1 and y^2. Suppose we measure the two outputs with some error (ϵ_{n1} and ϵ_{n2}), while the true physical process is defined by latent variables (f^1 and f^2). Then the relation between the output function, measurement error, and true physical process can be written as follows.

$$y^1 = f^1 + \epsilon_{n1} \tag{C.1}$$

$$y^2 = f^2 + \epsilon_{n2} \tag{C.2}$$

Here, ϵ_{n1} and ϵ_{n2} are measurement errors sampled from a white noise Gaussian $\mathcal{N}(0, \sigma_{n1}^2)$ and $\mathcal{N}(0, \sigma_{n2}^2)$ respectively. While, the relation between the latent function can be expressed as follows:

$$f^1(z) = \mathcal{L}\left(f^2(x), z\right) \tag{C.3}$$

Here $\mathcal{L}\left(.\right) \in \mathcal{C}^2$ is an operator defining the relation between f^1 and f^2. We can write a zero mean GP prior for the full set of inputs and outputs $\{\boldsymbol{X_{joint}}, \boldsymbol{y_{joint}}\}$, as equation 6.5.

$$\begin{aligned}\Pr[\boldsymbol{y_{joint}}] &= \Pr\begin{bmatrix}\boldsymbol{y_{joint}}(x,1)\\ \boldsymbol{y_{joint}}(x,2)\end{bmatrix}\\ &= GP\left[\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}Cov(\boldsymbol{y_{joint}}(x,1),\boldsymbol{y_{joint}}(x,1)) & Cov(\boldsymbol{y_{joint}}(x,1),\boldsymbol{y_{joint}}(x,2))\\ Cov(\boldsymbol{y_{joint}}(x,2),\boldsymbol{y_{joint}}(x,1)) & Cov(\boldsymbol{y_{joint}}(x,2),\boldsymbol{y_{joint}}(x,2))\end{pmatrix}\right]\\ &= GP\left[\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}Cov(\boldsymbol{f^1}(x),\boldsymbol{f^1}(x))+\sigma_{n1}^2 & Cov(\boldsymbol{f^1}(x),\boldsymbol{f^2}(x))\\ Cov(\boldsymbol{f^2}(x),\boldsymbol{f^1}(x)) & Cov(\boldsymbol{f^2}(x),\boldsymbol{f^2}(x))+\sigma_{n2}^2\end{pmatrix}\right]\end{aligned} \tag{C.4}$$

Here, $Cov(\boldsymbol{y_{joint}}(x,j),\boldsymbol{y_{joint}}(x,j))$ is called the auto-covariance between observations of the j^{th} output, while $Cov(\boldsymbol{y_{joint}}(x,2),\boldsymbol{y_{joint}}(x,1))$ is called cross-covariance between the 2^{nd} and the 1^{st} outputs. Similarly, $Cov(\boldsymbol{f^j}(x),\boldsymbol{f^j}(x))$ is the auto-covariance function between j^{th} latent function, while $Cov(\boldsymbol{f^1}(x),\boldsymbol{f^2}(x))$ is the cross-covariance function between the 2^{nd} and the 1^{st} latent outputs. We are thus, interested in evaluating the Gram matrix $\boldsymbol{K_{XX}}$ between the full dataset.

$$\boldsymbol{K_{XX}} = \begin{pmatrix}Cov(f^1(x),f^1(x))+\sigma_{n1}^2 & Cov(f^1(x),f^2(x))\\ Cov(f^2(x),f^1(x)) & Cov(f^2(x),f^2(x))+\sigma_{n2}^2\end{pmatrix} \tag{C.5}$$

$$= \begin{pmatrix}\boldsymbol{K_{11}}+\mathbb{I}\sigma_{n1}^2 & \boldsymbol{K_{12}}\\ \boldsymbol{K_{21}} & \boldsymbol{K_{22}}+\mathbb{I}\sigma_{n2}^2\end{pmatrix} \tag{C.6}$$

Proof The *Taylor* series expansion of $\mathcal{L}\left(f^2\right)$ around $E[f^2]$ gives:

$$\begin{aligned}\mathcal{L}\left(f^2\right) &= \mathcal{L}\left(E[f^2]\right) + L\delta_2 + \frac{1}{2}\delta_2^T H\delta_2 + \mathcal{O}(\delta_2^3)\\ L &= \tfrac{\partial\mathcal{L}}{\partial x}|_{f^2=E[f^2]}\\ H &= \frac{\partial^2\mathcal{L}(f^2)}{\partial^2 x}|_{f^2=E[f^2]}\end{aligned} \tag{C.7}$$

Where L is the Jacobian matrix of $\mathcal{L}\left(.\right)$ evaluated at the mean $(E[f^2])$ of latent output (f^2), while H is the derivative of L evaluated at the mean of f^2. δ_2 is the amplitude of small variations of f^2, introduced by the Taylor series expansion of $\mathcal{L}(f^2)$ with respect to $E[f^2]$.

On taking expectation of equation C.3:

$$\begin{aligned}E[f^1] &= E[\mathcal{L}(f^2)]\\ &= \mathcal{L}(E[f^2]) + \frac{1}{2}E[\delta_2^T H\delta_2] + E[\mathcal{O}(\delta_2^3)]\end{aligned} \tag{C.8}$$

Subtracting the above equation C.8 from equation C.3 gives:

$$f^1 - E[f^1] = \mathcal{L}(f^2) - \mathcal{L}(E[f^2]) - \frac{1}{2}E[\delta_2^T H \delta_2] - E[\mathcal{O}(\delta_2^3)] \tag{C.9}$$

Expanding $\mathcal{L}(f^2)$ in the above equation:

$$f^1 - E[f^1] = \mathcal{L}\left(E[f^2]\right) + L\delta_2 + \frac{1}{2}\delta_2^T H \delta_2 + \mathcal{O}(\delta_2^3) - \mathcal{L}(E[f^2]) - \frac{1}{2}E[\delta_2^T H \delta_2] - E[\mathcal{O}(\delta_2^3)] \tag{C.10}$$

$$\delta_1 = L\delta_2 + \left[\frac{1}{2}\delta_2^T H \delta_2 - \frac{1}{2}E[\delta_2^T H \delta_2]\right] + \left[\mathcal{O}(\delta_2^3) - E[\mathcal{O}(\delta_2^3)]\right] \tag{C.11}$$

Here, δ_1 is the amplitude of small variations of f^1. Multiplying the above equation by δ_2^T and taking the expectation gives:

$$\begin{aligned} E[\delta_1\delta_2^T] &= LE[\delta_2\delta_2^T] + E\left[\left[\frac{1}{2}\delta_2^T H \delta_2 - \frac{1}{2}E[\delta_2^T H \delta_2]\right]\delta_2^T\right] + E\left[\left[\mathcal{O}(\delta_2^3) - E[\mathcal{O}(\delta_2^3)]\right]\delta_2^T\right] \\ Cov(f^1, f^2) &= LCov(f^2, f^2) + E\left[\left[\frac{1}{2}\delta_2^T H \delta_2 - \frac{1}{2}E[\delta_2^T H \delta_2]\right]\delta_2^T\right] + E\left[\left[\mathcal{O}(\delta_2^3) - E[\mathcal{O}(\delta_2^3)]\right]\delta_2^T\right] \end{aligned} \tag{C.12}$$

If we eliminate the terms that are or order greater than $\mathcal{O}(\delta_2^3)$ then we get the following equation for $Cov(f^1, f^1)$:

$$Cov(f^1, f^2) = LCov(f^2, f^2) \tag{C.13}$$

Similarly by multiplying the equation C.10 by δ_1^T, taking the expectation and eliminating higher order terms will give:

$$\begin{aligned} E[\delta_1\delta_1^T] &= LE[\delta_2\delta_1^T] \\ Cov(f^1, f^1) &= LCov(f^2, f^1) \\ &= LCov(f^1, f^2)^T \\ &= L[LCov(f^2, f^2)]^T \\ &= LCov(f^2, f^2)L^T \end{aligned} \tag{C.14}$$

The last equation comes because $Cov(f^2, f^2)$ is symmetric, hence $Cov(f^2, f^2)^T = Cov(f^2, f^2)$. We can thus write the Gram matrix $\boldsymbol{K_{XX}}$ as:

$$\boldsymbol{K_{XX}} = \begin{pmatrix} L\boldsymbol{K_{22}}L^T + \mathbb{I}\sigma_{n1}^2 & L\boldsymbol{K_{22}} \\ \boldsymbol{K_{22}}L^T & \boldsymbol{K_{22}} + \mathbb{I}\sigma_{n2}^2 \end{pmatrix} \tag{C.15}$$

The proof for enforcing non-linear relationships in a multi-task GP regression framework is thus complete.