THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Optimizing performance in Gaussian Processes

Michael Ciccotosto-Camp

Supervisor: Fred (Farbod) Roosta
Co-Supervisors: Andries Potgieter
Yan Zhao

Bachelor of Mathematics (Honours)
June 2022

The University of Queensland
School of Mathematics and Physics

Contents

Acknowledgements

I would like to deeply thank my supervisor Dr. Masoud Kamgarpour for his advice and all of his time spent with me. I consider myself lucky and am glad to have been his student for my honours year. I would also like to thank my co-supervisor Dr. Anna Puskás for the same reasons. A special thanks to Dr. Valentin Buciumas for his time spent teaching me while he was at The University of Queensland.

Introduction

Origins of group theory. The mathematical field of group theory has its origins in the early 19th century. At the time, mathematicians were investigating the solutions to polynomial equations. That is, solutions to equations of the form

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 = 0.$$

Full solutions to polynomial equations of low degrees (i.e. $n \leq 4$) had already been formulated [?Riggs96]. These include the familiar *quadratic formula*, which has been known since antiquity. The formula tells us that the solutions to a general quadratic equation $ax^2 + bx + c = 0$ are given by

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

The full solutions to any cubic ($n = 3$) or quartic ($n = 4$) polynomial equation were also known. These are given by the lesser-known Cardano's formula and Ferrari's method, respectively. We say that a polynomial is *solvable by radicals* if one can write all of its solutions in terms of its coefficients combined with the algebraic operations; addition, subtraction, multiplication, division, powers and radicals (i.e. k^{th} roots).

In the 1830s, the mathematician Évariste Galois provided an elegant method to prove that a general polynomial of degree $n \geq 5$ is not solvable by radicals. Galois understood that to every polynomial one could associate a *Galois group*, a new mathematical object at the time. The Galois group was the first object in a class of mathematical objects that we call *groups* today. We say that the pair $(G, \circ)$ is a group, where G is a set and $\circ \colon G \times G \to G$ is a binary operation on G, when three conditions are satisfied:

- Associativity: $g \circ (h \circ k) = (g \circ h) \circ k$ for all $g, h, k \in G$.
- Existence of an identity: there exists some $1_G \in G$ such that $1_G \circ g = g \circ 1_G = g$ for all $g \in G$.
- Existence of inverses: for every $g \in G$, there exists some $g^{-1} \in G$ such that $g \circ g^{-1} = g^{-1} \circ g = 1_G$.

Examples of groups that are likely familiar to the reader include $(\mathbb{Z}, +)$, the integers under addition, $(\mathbb{R}^+, \times)$, the positive real numbers under multiplication, and $(\mathbb{Z}/n\mathbb{Z}, +)$, the integers modulo n under addition. Some geometric examples of groups are the *dihedral groups*. These groups are generated by the m symmetries associated to the regular m-sided polygon (i.e. a polygon with all interior angles and all side lengths the same). Then each dihedral group contains $2m$ elements (m reflections and m rotations) with the group operation of composition of reflections and rotations.

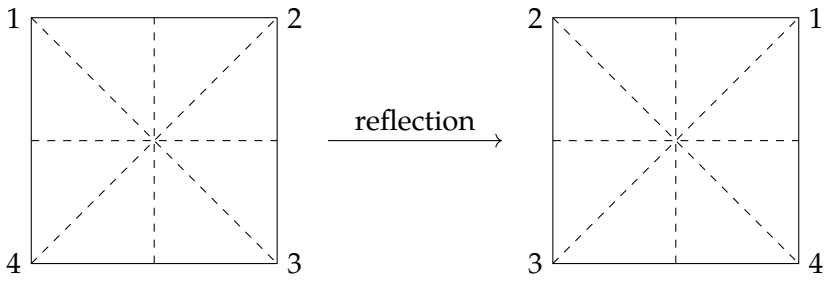


Figure 1. The symmetries of a square and a reflection about the vertical line of symmetry.

This gives us an intuitive understanding of groups: they encode the symmetries of mathematical objects.

What is representation theory? The study of groups yields insight into geometric objects. The action of the dihedral group on the m-gon serves as example of a group acting on a geometric object. More generally, we can consider the action of a group on some object. Specifically, we say that a group G *acts* on a set X if, for each $g \in G$, there is a map $\cdot\colon G \times X \to X$ satisfying $1_G \cdot x = x$ and $g \cdot (h \cdot x) = (gh) \cdot x$ for all $x \in X$. Alternatively, one can view this as a group homomorphism $\rho\colon G \to \mathrm{Sym}(X)$, where $\mathrm{Sym}(X)$ is the symmetric group associated to X, i.e. the group of permutations of elements of X.

Now we linearise the setting above by requiring that $X = V$ is a *vector space*. Then we say that G acts *linearly* on V if there exists a group homomorphism $\rho\colon G \to \mathrm{GL}(V)$. We call (V, ρ) a *representation* of G, and ρ is often suppresed from notation. We see that G acts on V in the sense that $\rho(g)\colon V \to V$ is a linear invertible map on V. We may denote $\rho(g)(v)$ by $g \cdot v$ as before.

Representation theory is concerned with understanding and classifying linear actions of groups. The general situation of representation theory is as follows. If the group G acts on a vector space V, then we say that a vector subspace $W \subseteq V$ is a *subrepresentation* of V if it is invariant under the action of G. A representation is called *irreducible* if its only proper subrepresentation is the trivial representation $W = \{0\}$. The primary goals of representation theory are finding all irreducible representations of G, and to decompose a given representation into its irreducible components.

We can think of irreducible representations as the building blocks of all other representations. This is a common idea in mathematics, seen in other areas. For instance, in number theory, the building blocks of integers are primes and, in group theory, the building blocks of groups are simple groups.

Writing a general representation in terms of irreducible components is not always possible. We call a representation *decomposable* if we can write it as the direct sum of irreducible representations. A lot can be said about the case where the representation of a finite group is over a field whose characteristic not dividing the order of the group. In this case, *Maschke's theorem* tells us that these representations are always decomposable [?Lang02]. In particular, complex representations of a finite group are always decomposable.

Gelfand Pairs. Henceforth, we assume some knowledge of abstract algebra from the reader. Let G be a finite group and $K \leq G$ a subgroup. The pair (G, K) is called a *Gelfand pair* if the induced representation $\mathrm{Ind}_K^G \mathbf{1}$ is multiplicity-free. Here $\mathbf{1}$ denotes the trivial (1-dimensional) complex representation of K, and multiplicity-free means that any irreducible representation appears in the decomposition of $\mathrm{Ind}_K^G \mathbf{1}$ at most once (up to isomorphism).

Gelfand pairs play an important role in representation theory [?Musili93], analysis [?Koranyi80, ?Morel18], combinatorics [?Bannai84], number theory [?Gross91, ?Terras99] and probability [?CSST20, ?Diaconis88]. One of our objectives is to give a detailed study of Gelfand pairs of finite groups. A main theorem of this thesis is the following:

Theorem 1. (*Gelfand's Trick*) *Let G be a finite group and K a subgroup of G. Suppose $\varphi\colon G \to G$ is an involutive anti-automorphism (i.e. a bijective anti-homomorphism) such that $K\varphi(x)K = KxK$ for all $x \in G$. Then (G, K) is a Gelfand pair.*

The theorem above is proved using the *Hecke algebra*. There are multiple constructions of Hecke algebras in the literature [?CMHL03, ?CSST20].

Types of Hecke algebras. Another objective of this thesis is to present these a priori different Hecke algebras and resolve their apparent discrepancies. For instance, one way to define the Hecke algebra is as a convolution algebra of K-bi-invariant complex-valued functions $f : G \to \mathbb{C}$ on a group. Another way to define the Hecke algebra is as the algebra generated by $n-1$ variables $T_1, \dots T_{n-1}$ subject to a *quadratic relation* $T_i^2 = (q-1)T_i + q$ and a *braid relation*

$$\underbrace{T_iT_jT_i\dots}_{m_{ij}\text{ terms}} = \underbrace{T_jT_iT_j\dots}_{m_{ij}\text{ terms}}.$$

Here m_{ij} is the ij^{th} entry in the *Coxeter matrix* associated to the *Weyl group* of G. The name 'braid relation' is due to a method of visualising the *symmetric group* S_n. If n is a positive integer then the group S_n is the collection of bijections on the set $\{1, 2, \dots, n\}$ to itself, with the group operation of composing functions. A natural method of visualising elements and multiplication in this group is via *braid diagrams*. For instance, if $\sigma = (1\ 2)(3\ 5\ 4)$ and $\pi = (1\ 2\ 4\ 6\ 5\ 3)$ are permutations in S_6 (written in cycle notation), then we may visualise these elements and their product $\pi\sigma = (1\ 4)(5\ 6)$ in the following manner:

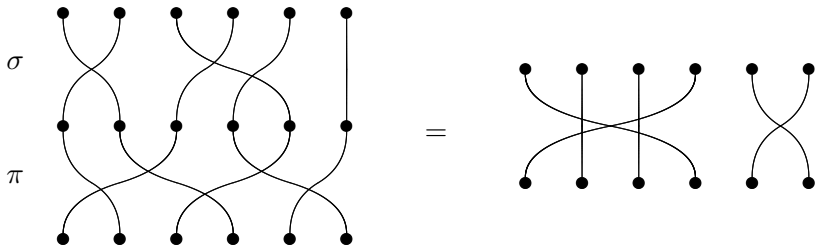


FIGURE 2. A braid diagram visualising the multiplication $\pi\sigma = (1\ 4)(5\ 6)$.

Why study Hecke algebras? The Hecke algebra arises naturally when one wishes to compute certain irreducible representations of a group [?Williamson21, ?CMHL03]. Consider a finite group G and a normal subgroup $N \triangleleft G$. If G acts linearly on a vector space V (i.e. V is a representation of G), then there is a natural action of G on the subrepresentation V^N, the space of vectors in V that are fixed by N. Under this action, N will clearly act trivially on V^N. This yields a representation of the quotient group G/N. After some representation theoretic arguments, one arrives at the conclusion that

$$\left\{ \begin{array}{c} \text{Irreducible representations} \\ \text{of } G \text{ with } N\text{-fixed vectors} \end{array} \right\} \overset{1:1}{\longleftrightarrow} \left\{ \begin{array}{c} \text{Irreducible representations} \\ \text{of } G/N \end{array} \right\}.$$

It is a straightforward exercise that a complex representation of the group G/N is the same as a representation of the algebra $\mathbb{C}[G/N]$, the *group algebra* of G/N.

What happens when we do not require a normal subgroup of G? Consider an arbitrary subgroup K of a finite group G. Now G/K is no longer necessarily a group, so G/K and $\mathbb{C}[G/K]$ no longer necessarily make sense. We ask ourselves: what acts on V^K? The action of G on V is not well-defined on V^K since

K is no longer normal. It is not obvious how we could study irreducible G-representations with K-fixed vectors. We are able to salvage the situation with the help of the Hecke algebra.

For $g \in G$, define the Hecke operator $[KgK] := \frac{1}{|K|}\sum_{x\in KgK} x \in \mathbb{C}[G]$, which acts on V^K by

$$[KgK] \cdot v := \frac{1}{|K|}\sum_{x\in KgK} x \cdot v.$$

Now define the Hecke algebra $\mathcal{H}(G, K)$ to be the space of functions $f\colon G \to \mathbb{C}$ that are constant on K-double cosets. The indicator functions χ_{KgK} form a basis of this space and we can uniquely associate the indicator functions χ_{KgK} to the Hecke operators $[KgK]$. We see that, through the Hecke operators, we have defined an action of $\mathcal{H}(G, K)$ on V^K. This answers our question of what acts on V^K. Through another representation-theoretic exercise, one can conclude that

$$\left\{\begin{array}{c}\text{Irreducible representations}\\ \text{of } G \text{ with } K\text{-fixed vectors}\end{array}\right\} \overset{1:1}{\longleftrightarrow} \left\{\begin{array}{c}\text{Irreducible representations}\\ \text{of } \mathcal{H}(G, K)\end{array}\right\}.$$

An immediate example of the utility of this result is as follows. It is easy to show that if $\mathcal{H}(G, K)$ is commutative, then all of its irreducible finite-dimensional representations are one-dimensional [?Etingof11]. The commutativity of the Hecke algebra turns out to be an important property which will be investigated throughout this thesis.

Contents of this thesis. In Chapter 1, we begin our study of the Hecke algebra. First, we investigate the convolution algebra of all complex-valued functions on G and its ideal of K-right-invariant complex-valued functions. This is followed by results describing the relationship between the induced representation and its associated Hecke algebra. We use these results to prove Theorem 1. This allows us to write down simple proofs that $\mathrm{Ind}_K^G \mathbf{1}$ is multiplicity-free for certain choices of G and K. Namely, (G, K) with G commutative, (G, K) with $[G\colon K] = 2$, $(S_{n+m}, S_n \times S_m)$, and $(O_{n+1}(\mathbb{F}_q), O_n(\mathbb{F}_q))$ for q odd.

In Chapter **??**, we generalise the discussion of Chapter 1 to the case of a non-trivial *character* $\sigma\colon K \to \mathbb{C}^\times$. Here our goal is to obtain a twisted analogue of Theorem 1. To this end, we describe the basis of the Hecke algebra using the idea of *relevant orbits*. We state and prove the generalisation of Theorem 1. We apply the new theorem to a particular representation, the *Gelfand–Graev representation* of $\mathrm{GL}_n(\mathbb{F}_q)$, to show that it is multiplicity-free.

In Chapter 2, we investigate the Hecke algebra of Chapter 1 under the particular choice of $G = \mathrm{SL}_n(\mathbb{F}_q)$ and $K = B(\mathbb{F}_q)$, the *Borel subgroup* of G, i.e. the subgroup of upper-triangular matrices. The *Weyl group* associated to G is introduced and shown to be isomorphic to S_n. Next, we perform some elementary matrix calculations which yields the surprising result above: the Hecke algebra may be written in terms of $n-1$ generators subject to the quadratic relation and the braid relations associated to W. This leads to a concluding discussion of Hecke algebras generated by any finite *Coxeter group*.

In Chapter 3, we generalise the results of earlier chapters to the case where G is no longer finite, but instead a locally compact topological group. This allows for an extension of the theory we have developed to more general groups and their Hecke algebras. To do this, we discuss how one can impose a topological structure on a group and supply examples to give some intuition for these types of groups. To define Hecke algebras of these groups, we require some measure theory. In particular, the convolution product

on the Hecke algebra is defined in terms of an integral with respect to the *Haar measure*. We spend some time developing the theory of Haar measures for this purpose. We conclude with a discussion of how to recover the Hecke algebra of a finite group from this new definition. In this chapter, we shall denote the Hecke algebra by $C_c(K\backslash G/K)$ to emphasise the non-finiteness of G.

In Chapter **??**, we take a look at some specific Hecke algebras of locally compact topological groups. In particular, we restrict our attention to the general linear group over a non-archimedian local field k and its ring of integers $\mathcal{O}$. We look at the *Spherical Hecke algebra*, formed when one considers $G = \mathrm{GL}_n(k)$ and $K = K^\circ := \mathrm{GL}_n(\mathcal{O})$, and the *Iwahori–Hecke algebra*, formed when one considers $G = \mathrm{GL}_n(\mathcal{O})$ and $K = I$, the *Iwahori subgroup*. In order to investigate these algebras, we must develop an understanding of these fields. We detail their definition, classification and structure.

The contents of this thesis may be visualised with the following diagram.

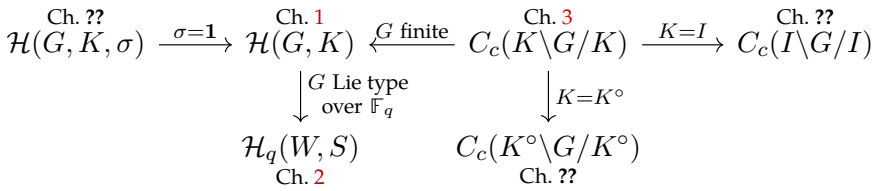


FIGURE 3. The relationship diagram of this thesis.

Directions for future research. We assume the reader is familiar with the contents of this thesis. The modern study of Hecke algebras is largely focused on the *Iwahori–Hecke algebra*, which is also known as the *affine Hecke algebra*. This algebra is central to the study of representations of *reductive groups* over non-archimedian local fields (e.g. groups such as GL_n, SL_n, Sp_{2n} over fields such as $\mathbb{Q}_p$ or $\mathbb{F}_q(\!(t)\!)$).

Some topics relevant to the Iwahori–Hecke algebra include *Bernstein's presentation*, the *Iwahori–Matsumoto presentation* and the *Satake isomorphism* [?HKP]. Properties of the Iwahori–Hecke algebra such as these presentations may be viewed as a consequence of the *universal unramified principal series module*, which we now describe.

Fix a "nice" (i.e. split and connected) reductive group G (e.g. SL_n) over a non-archimedian local field k with ring of integers $\mathcal{O}$. Then write A to mean a *split maximal torus* of G and write N to mean the *unipotent radical* of a Borel subgroup of G that contains A. Also recall I is the Iwahori subgroup of G given in Chapter **??**.

The universal unramified principal series module M is given by $C_c(A(\mathcal{O})N\backslash G/I)$. It is a right module over the Iwahori–Hecke algebra under convolution. Furthermore, a basis of the Iwahori–Hecke algebra is parameterised by the *affine Weyl group* $\widetilde{W}$. We may write $\widetilde{W} \cong W \ltimes \Lambda^\vee$, where $\Lambda^\vee$ is the *coroot lattice* of G. Then $\mathbb{C}[\Lambda^\vee]$ is the corresponding group algebra over $\mathbb{C}$. Then M is also a left module over $\mathbb{C}[\Lambda^\vee]$.

1. Gaussian Processes

The aim of this chapter is to review some essential mathematical machinery required for us to understand the core concepts of Gaussian Processes.

1.1. **Kernels.** Often in machine learning we are often met with the problem of how to best represent data instances as fixed size feature vectors $\boldsymbol{x}_i \in X$. For certain objects it might not be obvious at all on how to represent the data as a fixed length vector. Good examples of variable length data include textual documents and genomic data. For these types of data we can instead define a method of measuring similarity between object which does require them to be converted to a fixed length feature vector first [Mur12]. To do this we can begin by mapping the feature vectors into a Hilbert space H which provides us with an inner product $\langle \cdot, \cdot \rangle_H : H \times H \to \mathbb{R}$ and a norm $\| \cdot \|_H : H \to \mathbb{R}$. Input data is transformed into feature space vectors via a non-linear feature mapping $\Phi : X \to H$. The benefit of using a feature mapping in this way is that we can construct non-linear descision boundaries using linear models. In some instances a similarity measure can be computed directly using a function $k : X \times X$, instead of needing to construct a Φ and then computing the inner product of the transformed instances. We call such functions that act directly of our data instances kernel functions and using a kernel function to avoid computation associated with the underlying feature space is known as the kernel trick [Ste08]. These ideas are stated more formally in definition 2.

Definition 2 (Kernel)**.** *Let X be a non-empty set. Then a function $k : X \times X \to \mathbb{R}$ is called a kernel on X if there exists a Hilbert space and a map $\Phi : X \to H$ such that for all $\boldsymbol{x}, \boldsymbol{x}' \in X$ we have $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_H$. We call the Φ the feature map and H the feature space of k.*

It is worth noting that almost no conditions are placed on the set X, allowing it to accommodate virtually any form of data. It is of little surprise then that neither the feature map nor the feature space are uniquely determined by the kernel. As a quick example from Ingo Steinwart and Andreas Christmann book *Support Vector Machines* [Ste08], let $X = \mathbb{R}$ and $k(x, x') = x \cdot x'$ where $x, x' \in X$. We can see that k is a kernel using the feature map $\Phi(x) \triangleq x$ and $H = \mathbb{R}$. However another suitable feature map for this particular kernel is $\Phi'(x) \triangleq \left(x/\sqrt{2}, x/\sqrt{2}\right)$ with a corresponding feature space of $H = \mathbb{R}^2$ since

$$\langle \Phi'(x), \Phi'(x') \rangle_{\mathbb{R}^2} = \frac{x'}{\sqrt{2}} \cdot \frac{x}{\sqrt{2}} + \frac{x'}{\sqrt{2}} \cdot \frac{x}{\sqrt{2}} = x \cdot x'$$

for $x, x' \in X$. While their might be numerous functions that provide some notion of similarity between data entries, these functions might not be valid kernels. Instead of needing to construct a feature map and feature space to verify that a chosen function is a valid kernel using definition 2, we can make use of a much simpler set of criteria. First, we shall need the following definition.

Definition 3 (Positive Definite and Positive Semidefinite)**.** *A function $k : K \times K \to \mathbb{R}$ is positive semidefinite if for all $n \in \mathbb{N}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ and all $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in X$ we have*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) \geq 0. \tag{1}$$

Furthermore, k is said to be positive definite if for mutually distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in X$ equality 1 only holds for $\alpha_1 = \ldots = \alpha_n = 0$ [Ste08].

Definition 4 (Symmetric). *A function $k : K \times K \to \mathbb{R}$ is called symmetric if $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$ for any inputs $\boldsymbol{x}', \boldsymbol{x} \in X$* [Ste08].

Definition 5 (Gram Matrix). *For fixed $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in X$ the matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ where $\boldsymbol{K}_{i,j} \triangleq k(\boldsymbol{x}_j, \boldsymbol{x}_i)$ is the Gram matrix* [Ste08].

Note that checking if a function is positive (semi) definite is equivalent to checking that any Gram matrix produced by function is positive (semi) definite. If k is a real valued kernel corresponding to the feature map Φ, then k is symmertic by virtue of the fact that the inner product of a real Hilbert space is symmetric. Moreover k is positive definite since for $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in X$ we have

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle_H \\
&= \left\| \sum_{i}^{n} \alpha_i \Phi(\boldsymbol{x}_i) \right\|_H^2 \\
&\geq 0.
\end{aligned}$$

The following theorems tell us that it is not only necessary for a kernel to be positive semi definite but it is also a sufficient condition.

Theorem 6. *A function $k : K \times K \to \mathbb{R}$ is a kernel if and only if it is symmertic and positive semidefinite* [Ste08].

1.2. **Reproducing Kernel Hilbert Spaces.** We shall now shift our attention towards reproducing kernel Hilbert spaces (RKHS) and describe their relation to kernels. We shall see that in some sense the RKHS of a kernel k is the smallest feature space for a kernel. The formal definition of a RKHS is stated in definition 7.

Definition 7 (RKHS). *Let $X \neq$ and H be a real Hilbert space over X*

(1) *A function $k : X \times X \to \mathbb{R}$ is called a reproducing kernel if we have $k(\cdot, \boldsymbol{x}) \in H$ for all $\boldsymbol{x} \in X$ and the reproducing property*

$$f(\boldsymbol{x}) = \langle f, k(\cdot, \boldsymbol{x}) \rangle$$

holds for all $f \in H$ and $x \in X$.

(2) *The space H is called a reproducing kernel Hilbert space over X if for all $\boldsymbol{x} \in X$ the Dirac functional $\delta_{\boldsymbol{x}} : H \to \mathbb{R}$ defined by $\delta_{\boldsymbol{x}}(f) \triangleq f(x),\ f \in H$ is continuous.*

[Ste08]

An important property of the RKHS is that the convergence in the norm implies pointwise convergence. Specifically in a RKHS for any sequence of functions $\{f_n\} \subset H$ where $\|f_n - f\| \to 0$ we have $|\delta_{\boldsymbol{x}}(f_n) - \delta_{\boldsymbol{x}}(f)| = |f_n(x) - f(x)| \to 0$. Note that because the evaluation function is both linear and continuous then it is also bounded in the sense that there is an $c \in \mathbb{R},\ c > 0$ such that for every $f \in H$

and a fixed $\boldsymbol{x} \in X$ we have $|\delta_{\boldsymbol{x}}(f)| \leq c\|f\|_H$ [Ber96]. This property of uniform convergence implying pointwise convergence is important since it tells us that if functions $f, g \in H$ are close in norm then their evaluation at any point is also similar. The following lemma ties together the definition of an RKHS, reproducing kernel and a kernel.

Lemma 8. *Let H be a Hilbert function space over X that has a reproducing kernel k. Then H is a RKHS and H is also a feature space of k where the feature map $\Phi : X \to H$ is given by*

$$\Phi(\boldsymbol{x}) = k(\cdot, \boldsymbol{x})$$

for some $\boldsymbol{x} \in X$. We call Φ the canonical feature map.

Proof. Since the reproducing property tells us that any Dirac functional can be represented by the reproducing kernel this means

$$|\delta_{\boldsymbol{x}}(f)| = |f(\boldsymbol{x})| = |\langle f, k(\cdot, \boldsymbol{x})\rangle| \leq \|k(\cdot, \boldsymbol{x})\|_H \cdot \|f\|_H$$

for all $\boldsymbol{x} \in X,\ f \in H$. This shows continuity of $\delta_{\boldsymbol{x}}$ for $\boldsymbol{x} \in X$. In order to show that Φ is a feature map, fix an $\boldsymbol{x}' \in X$ and set $f = k(\cdot, \boldsymbol{x}')$. Then for $\boldsymbol{x} \in X$, the reproducing property yields

$$\langle \Phi(\boldsymbol{x}'), \Phi(\boldsymbol{x})\rangle_H = \langle k\left(\cdot, \boldsymbol{x}'\right), k(\cdot, \boldsymbol{x})\rangle_H = \langle f, k(\cdot, \boldsymbol{x})\rangle_H = f(\boldsymbol{x}) = k\left(\boldsymbol{x}', \boldsymbol{x}\right).$$

□

This tells us that every Hilbert space with a reproducing kernel is a RKHS. We can also show the converse, that is, every RKHS has a unique reproducing kernel seen in theorem 9.

Theorem 9. *Let H be a RKHS over X. Then $k : X \times X \to \mathbb{R}$ defined by $k(\boldsymbol{x}', \boldsymbol{x}) = \langle \delta_{\boldsymbol{x}}, \delta_{\boldsymbol{x}'}\rangle_H,\ \boldsymbol{x}, \boldsymbol{x}' \in X$ is the only reproducing kernel of H* [Ste08].

Theorem 9 shows that a RKHS is uniquely determined by its kernel. In fact the other direction can also be shown giving a one-to-one correspondence between kernels and RKHS. This is known as the Moore-Aronszajn theorem seen in thorem 10.

Theorem 10 (Moore-Aronszajn)**.** *Suppose k is a symmertic positive definite kernel on a set X. Then there is a unique Hilbert space of functions for which k is the reproducing kernel* [Ber03].

The elements of a RKHS will often inherit the analytical properties of its corresponding kernel. This means that kernels provide a mechanism for generating spaces of functions with useful analytical properties.

1.3. **Gaussian Radial Basis Kernel.** We shall now focus on a specific class of kernel that shall be used extensively in upcoming theory and experimentation.

Definition 11 (Gaussian Radial Basis Kernel)**.** *For $d \in \mathbb{N},\ \sigma \in \mathbb{R}_{>0}$ and $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^d$ we define*

$$k_\sigma\left(\boldsymbol{z}, \boldsymbol{z}'\right) \triangleq \exp\left(-\sigma^{-2}\sum_{j=1}^{d}\left(z_j - {z'}_j\right)^2\right).$$

Then k_σ is a real valued kernel called the Gaussian Radial Basis Kernel (RBF) kernel with bandwidth σ. Moreover k_σ can be computed as

$$\exp\left(\frac{-\|\boldsymbol{z}-\boldsymbol{z}'\|_2^2}{\sigma^2}\right)$$

[Ste08].

The Gaussian RBF kernel makes for a very simple an intuitive measurement of similarity between its inputs. One geometric interpretation of the Gaussian RBF kernel is that as the radius of the smallest d-sphere containing $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^d$ grows the corresponding measurement of similarity decays exponentially. A visual representation of this decay is shown in figure **??**.

This kernel is infinitely differentiable meaning it has mean square derivatives of all orders and is therefore very smooth. In fact, some argue that such strong smoothness makes it unrealistic for modelling natural phenomena [Ras06, Ste99]. Nontheless, Gaussian RBF kernelis remains the one of the most widely used kernels in literature.

1.4. **Kernel Machines.**

1.4.1. *Introduction to Support Vector Machines for Binary Classification.* In this section, we shall look at two different machine learning models that make use of kernels to perform classification and regression. The first kernel machine we shall look at are support vector machines (SVM). SVMs where originally designed for binary classification and as such we shall only present a model for binary classification, although extensions exist that allow regression and multi-class classification.

For the binary classification problem we are tasked with labelling new samples with either one of two classes, -1 or 1. We shall assume our input space consists of vectors from $\mathbb{R}^d$ and that we provided with a labelled training set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. One simple method to classify samples is by creating an affine linear hyperplane satisfying

$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b > 0, \quad y_i = +1 \\ \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b < 0, \quad y_i = -1 \end{aligned} \tag{2}$$

for some $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ where $\|\boldsymbol{w}\|_2 = 1$. Moreover we would like $\boldsymbol{w}$ and b to maximise the margin, that is the maximal distance between the separating hyperplane and the points in D. The specific $\boldsymbol{w}$ and b obtained through the training set is denoted $\boldsymbol{w}_D$ and b_D and the resulting descision function is defined as

$$f_D(\boldsymbol{x}) \triangleq \operatorname{sign}\left(\langle \boldsymbol{w}_D, \boldsymbol{x} \rangle + b_D\right).$$

There are, however, a number of short comings to this model. The most obvious is that our training data may not be linearly separable in $\mathbb{R}^d$ meaning no such $\boldsymbol{w}_D$ and b_D exist. Moreover, when noise is introduced to the data set this model will prioritize finding a hyperplane that perfectly separates the two classes, making no comprises in misclassifying points therefore leaving it subject to overfitting. SVMs where introduced by Boser *et al.* [Bos92] to address the first issue of separability. Their approach was to lift the input vector into a more malleable Hilbert space H_0 using a feature map. The inputs are then classified within the new space. Unfortunately this method does nothing to address the second

issue of over fitting and, if anything, actually worsens it. Cortes and Vapnik [Cor95] attempted to address this second issue by introducing slack variables to equation 2 so that we instead need to satisfy $y_i\left(\langle \boldsymbol{w}, \Phi\left(\boldsymbol{x}_i\right)\rangle + b\right) \geq 1 - \xi_i$ for some $\xi_i \in \mathbb{R}_{>0}$. These constraints can be re-written as

$$\xi_i \geq 1 - y_i\left(\langle \boldsymbol{w}, \Phi\left(\boldsymbol{x}_i\right)\rangle + b\right)$$

and combining this with our slack constraints (that is $\xi_i \geq 0$) yields

$$\xi_i \geq \max\left\{0, 1 - y_i\left(\langle \boldsymbol{w}, \Phi\left(\boldsymbol{x}_i\right)\rangle + b\right)\right\} = L_{\text{hinge}}\left(y_i, \langle \boldsymbol{w}, \Phi\left(\boldsymbol{x}_i\right)\rangle + b\right)$$

where L_{hinge} is the hinge loss defined as

$$L_{\text{hinge}}\left(y, \eta\right) \triangleq \max\left\{0, 1 - y\eta\right\}.$$

This optimization problem can be re-written is the form

$$\min_{(\boldsymbol{w},b)\in H_0\times\mathbb{R}} \lambda\left\|\boldsymbol{w}\right\|_{H_0} + \frac{1}{n}\sum_{i=1}^{n} L_{\text{hinge}}\left(y_i, f_{(\boldsymbol{w},b)}\right)$$

where $f_{(\boldsymbol{w},b)} : X \to \mathbb{R}$ is defined as

$$f_{(\boldsymbol{w},b)} \triangleq \langle \boldsymbol{w}, \Phi\left(x_i\right)\rangle + b.$$

Unfortunately, this new embedding requires us to solve for optimal parameters in a very high, or even infinite, dimension vector space. To get around this, often the Lagrange approach is used to solve the corresponding dual problem. When the hinge loss is used the dual problem becomes

$$\begin{aligned} &\max_{\alpha\in[0,C]^n} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \Phi\left(\boldsymbol{x}_i\right), \Phi\left(\boldsymbol{x}_j\right)\rangle \\ &\text{subject to} \quad \sum_{i=1}^{n} y_i\alpha_i = 0 \end{aligned} \tag{3}$$

Notice that in the dual problem, we find that inner products are only taken with vectors that have the feature map applied to them allowing us to employ the kernel if the corresponding kernel trick described in section 1.1 is known for the feature map used so that 3 becomes

$$\begin{aligned} &\max_{\alpha\in[0,C]^n} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \\ &\text{subject to} \quad \sum_{i=1}^{n} y_i\alpha_i = 0. \end{aligned}$$

1.4.2. *Introduction to Gaussian Processes for Regression.* The next machine learning model of interest that uses kernels are gaussian processes. To motivate this model, consider the time series data in figure 4 (A).

In this diagram there is a number of observed values $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$ (blue labels) as well as a missing observation at time $x^\star$. This is a classic problem of time series prediction. What seems like a good prediction for the missing value at time $x^\star$? Perhaps something close to the red diamond seen in figure 4 (B). Why does this red diamond seem like a good choice? Because for known data for which the measurement of similarity is small, we expect the corresponding outputs should also be similar since most natural phenomena are continuous by nature. This reasoning is used as the founding ideas behind

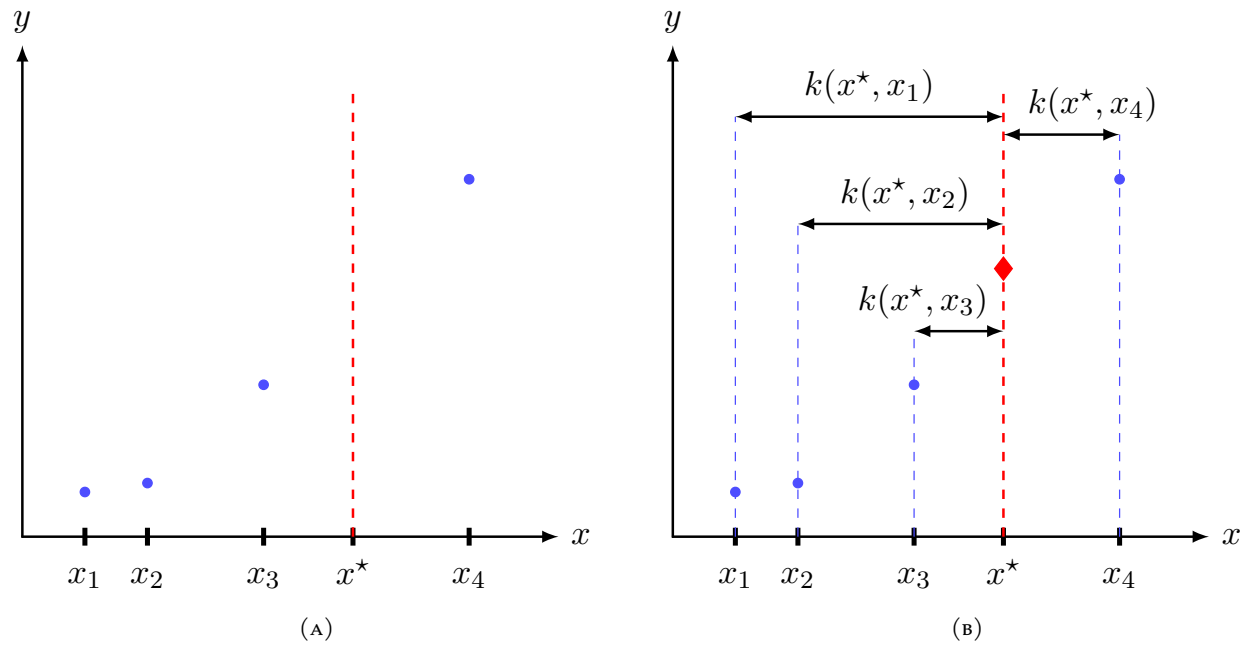


FIGURE 4. Panel (A) shows depicts the classical problem of time series prediction, guessing a value for $x^\star$ given values for surrounding times. Panel (B) shows a suitable choice for the value at time $x^\star$ with the reasoning that closely surrounding values should have greater influence over inference.

GPs, that is, training inputs that are more similar value we would like to make predictions for should have a greater influence over the prediction.

Similar to SVMs we can motivate the mathematical model of a GP through a linear model. Since GPs are designed for regression tasks, we shall only focus on motivating GP regression although we will see later on that GPs can be extended to perform binary classification and even multi-class classification. To begin, consider the following linear regression model

$$f(\boldsymbol{x}) \triangleq \langle \boldsymbol{w}, \boldsymbol{x} \rangle \tag{4}$$

where we are again assuming that $\boldsymbol{x}$ belongs to $\mathbb{R}^d$ and that $\boldsymbol{w} \in \mathbb{R}^d$ is a weight vector. Notice the striking resemblances to the linear classifier used to motivate SVMs, although this time we are using the value computed by the inner product directly to infer instead of fitting it over a sign function to force it into a binary class. Suppose we have independently sampled observations $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ to a noisy version of f

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i$$

where $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_n^2)$. Together the assumption of noise and the base linear model give rise to a likelihood, or more specifically, a probability density over the observations given the inputs and weight parameters. Due to the assumption of independence in our observations

$$p(y \mid \boldsymbol{X}, \boldsymbol{w}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \tag{5}$$

$$
\begin{aligned}
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)^2}{2\sigma_n^2}\right) \\
&= \frac{1}{(2\pi\sigma_n^2)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma_n^2}\left(\sum_{i=1}^{n} (y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)^2\right)\right) \\
&= \frac{1}{(2\pi\sigma_n^2)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma_n^2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2\right) \\
&= \mathcal{N}\left(\boldsymbol{X}\boldsymbol{w}, \sigma_n^2\, \mathbb{1}_{n\times n}\right)
\end{aligned}
$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^\intercal \in \mathbb{R}^n$ and $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^\intercal \in \mathbb{R}^{n\times d}$. Within the Bayesian paradigm, a prior is required to represent our beliefs about the parameters in the absence of any information. Typically, the following prior is used for the weight vector

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p)$$

where $\boldsymbol{\Sigma}_p$ is an appropriate covariance matrix. We would like to know the posterior pdf $p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})$ which refines our choices of $\boldsymbol{w}$ by taking into account our observations. The posterior can be computed using Bayes rule

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X})\, p(\boldsymbol{w}).$$

Equation 5 gives us a probability for $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X})$ and since $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p)$ then

$$p(\boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^d\, |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\boldsymbol{w}^\intercal \boldsymbol{\Sigma}_p^{-1} \boldsymbol{w}\right)$$

[Kro14]. This means, up to proportionality

$$
\begin{aligned}
p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) &\propto \exp\left(-\frac{1}{2\sigma_n^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^\intercal (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})\right) \exp\left(-\frac{1}{2}\boldsymbol{w}^\intercal \boldsymbol{\Sigma}_p^{-1}\boldsymbol{w}\right) \\
&\propto \exp\left(-\frac{1}{2}(\boldsymbol{w} - \bar{\boldsymbol{w}})^\intercal \left(\frac{1}{\sigma_n^2}\boldsymbol{X}^\intercal \boldsymbol{X} + \boldsymbol{\Sigma}_p^{-1}\right)(\boldsymbol{w} - \bar{\boldsymbol{w}})\right)
\end{aligned}
$$

where $\bar{\boldsymbol{w}} \triangleq \sigma_n^{-2}\left(\sigma_n^{-2}\boldsymbol{X}^\intercal \boldsymbol{X} + \boldsymbol{\Sigma}_p^{-1}\right)^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$. Notice that this again is a multivariate Gaussian distribution with mean $\bar{\boldsymbol{w}}$ and covariance $\boldsymbol{A}^{-1}$ where $\boldsymbol{A} \triangleq \sigma_n^{-2}\boldsymbol{X}^\intercal \boldsymbol{X} + \boldsymbol{\Sigma}_p^{-1}$ so that

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{N}\left(\boldsymbol{w}, \boldsymbol{A}^{-1}\right)$$

To make a prediction of our target function for an input, $\boldsymbol{x}^\star$, outside our observed values we can take the average over all possible parameter values weighted by the posterior to predict $f^\star \triangleq f(\boldsymbol{x}^\star)$ which yields

$$p(f^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y}) = \int_{\mathbb{R}^d} p(f^\star \mid \boldsymbol{x}^\star, \boldsymbol{w})\, p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y})\; d\boldsymbol{w} = \mathcal{N}\left(\boldsymbol{x}^{\star\intercal}\bar{\boldsymbol{w}}, \boldsymbol{x}^{\star\intercal}\boldsymbol{A}^{-1}\boldsymbol{x}^\star\right).$$

This gives another Gaussian distribution whose means is the mean of the posterior distribution of the weight vectors multiplied by the input vector, and whose covariance in the quadratic form of the covariance of the weight vectors again with the input vectors. This makes sense since it tells us that the uncertainty of the model grows quadratically with the magnitude of the input.

We can now use the kernel trick in the exact same manner in the derivation of the SVM model, that is, by using a feature mapping Φ to lift the inputs of our linear regression model from equation 4 into a higher

dimension and more workable Hilbert space so that our model now becomes.

$$f(\boldsymbol{x}) \triangleq \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle.$$

The derivation for the new model is identical with the only difference being that $\boldsymbol{x}^\star$ is replaced with $\Phi(\boldsymbol{x}^\star)$ and $\boldsymbol{X}$ is replaced with $\Phi(\boldsymbol{X}) \triangleq [\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2), \ldots, \Phi(\boldsymbol{x}_n)]^\intercal \in \mathbb{R}^{n \times N}$ where N is the dimension of the Hilbert space. The new predictive distribution becomes

$$f^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\Phi(\boldsymbol{x}^\star)^\intercal \boldsymbol{A}^{-1}\Phi(\boldsymbol{X})^\intercal \boldsymbol{y}, \Phi(\boldsymbol{x}^\star)^\intercal \boldsymbol{A}^{-1}\Phi(\boldsymbol{x}^\star)\right) \tag{6}$$

where $\boldsymbol{A}$ is now $\boldsymbol{A} \triangleq \frac{1}{\sigma_n^2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X}) + \boldsymbol{\Sigma}_p^{-1} \in \mathbb{R}^{N \times N}$. From this, it becomes evident that the inverse of $\boldsymbol{A}$ is required to compute both the mean and the covariance. This is not favourable since this would require knowledge of the Hilbert space into which the feature map send inputs. Moreover, computing $\boldsymbol{A}^{-1}$ may become impractical in the dimension of the Hilbert space is incredibly large. Remember, the whole point of the kernel trick is to avoid any computation that involves direct knowledge of H and to instead use a kernel k to bypass these obstacles and indirectly produce inner products of the data applied to the feature map. With this in mind, let us try and find different expressions for the mean and the covariance of equation 6 the will enable us to apply the kernel trick. To start we can find a suitable expression for the mean. First define the notation

$$\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'} \triangleq \Phi(\boldsymbol{W})\,\boldsymbol{\Sigma}_p\Phi(\boldsymbol{W}')^\intercal \in \mathbb{R}^{n \times n'}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{W}' \in \mathbb{R}^{n' \times d}$ are two data matrices. Consider the following

$$\begin{aligned}
&\boldsymbol{A}\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal \\
&= \left(\sigma_n^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X}) + \boldsymbol{\Sigma}_p^{-1}\right)\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal \\
&= \sigma_n^{-2}\,\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\,\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal + \Phi(\boldsymbol{X})^\intercal \\
&= \sigma_n^{-2}\,\Phi(\boldsymbol{X})^\intercal\left(\Phi(\boldsymbol{X})\,\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal + \sigma_n^2\mathbb{1}_{n\times n}\right) \\
&= \sigma_n^{-2}\,\Phi(\boldsymbol{X})^\intercal\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right)
\end{aligned}$$

meaning

$$\begin{aligned}
\sigma_n^{-2}\,\Phi(\boldsymbol{X})^\intercal\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right) &= \boldsymbol{A}\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal \\
\sigma_n^{-2}\,\boldsymbol{A}^{-1}\Phi(\boldsymbol{X})^\intercal\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right) &= \boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal \\
\sigma_n^{-2}\,\boldsymbol{A}^{-1}\Phi(\boldsymbol{X})^\intercal &= \boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right)^{-1}
\end{aligned}$$

so that the current mean of

$$\frac{1}{\sigma_n^2}\Phi(\boldsymbol{x}^\star)^\intercal\,\boldsymbol{A}^{-1}\Phi(\boldsymbol{X})^\intercal\boldsymbol{y}$$

in equation 6 can be replaced with

$$\Phi(\boldsymbol{x}^\star)^\intercal\,\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right)^{-1}\boldsymbol{y}.$$

To find a more suitable expression for the covariance matrix, we will need the assistance of the matrix inversion lemma stated without proof in lemma 12.

Lemma 12 (Matrix Inversion Lemma)**.** *For* $\boldsymbol{Z} \in \mathbb{K}^{n\times m}, \boldsymbol{W} \in \mathbb{K}^{m\times m}$ *and* $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{n\times m}$ *then*

$$(\boldsymbol{Z} + \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\intercal)^{-1} = \boldsymbol{Z}^{-1} - \boldsymbol{Z}^{-1}\boldsymbol{U}\left(\boldsymbol{W}^{-1} + \boldsymbol{V}^\intercal\boldsymbol{Z}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{V}^\intercal\boldsymbol{Z}^{-1}$$

assuming the relevant inverses exist [Pre92][*page 75*].

Consider

$$\boldsymbol{A} = \boldsymbol{\Sigma}_p^{-1} + \Phi(\boldsymbol{X})^\intercal\left(\sigma_n^{-2}\mathbb{1}_{n\times n}\right)\Phi(\boldsymbol{X}) \tag{7}$$

then setting $\boldsymbol{Z}^{-1} = \boldsymbol{\Sigma}_p, \boldsymbol{W}^{-1} = \sigma_n^2\mathbb{1}_{n\times n}$ and $\boldsymbol{V} = \boldsymbol{U} = \Phi(\boldsymbol{X})$ then equation 7 becomes

$$\boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\left(\sigma_n^2\mathbb{1}_{n\times n} + \Phi(\boldsymbol{X})\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\right)^{-1}\Phi(\boldsymbol{X})\boldsymbol{\Sigma}_p$$

via the matrix inversion lemma. Thus equation 6 can be equivalently formulated as

$$\begin{aligned} f^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(&\Phi(\boldsymbol{x}^\star)^\intercal\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right)^{-1}\boldsymbol{y}, \\ &\Phi(\boldsymbol{x}^\star)^\intercal\boldsymbol{\Sigma}_p\Phi(\boldsymbol{x}^\star) - \Phi(\boldsymbol{x}^\star)^\intercal\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\left(\sigma_n^2\mathbb{1}_{n\times n} + \Phi(\boldsymbol{X})\boldsymbol{\Sigma}_p\Phi(\boldsymbol{X})^\intercal\right)^{-1}\Phi(\boldsymbol{X})\boldsymbol{\Sigma}_p\Phi(\boldsymbol{x}^\star)) \end{aligned} \tag{8}$$

The astute reader may have noticed the very suggestive notation of labelling matrices of the form $\Phi(\boldsymbol{W})\boldsymbol{\Sigma}_p\Phi(\boldsymbol{W}')^\intercal$ as $\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'}$ as though it may have some sort of connection to a kernel. To make this even more obvious, notice that each occurance of the feature map in both expressions for the mean and covariance in equation 8 can be replaced with a $\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'}$ for some appropriate choice of $\boldsymbol{W}$ and $\boldsymbol{W}'$ giving a more notationally cleaner expression

$$f^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{K}_{\boldsymbol{x}^\star\boldsymbol{X}}\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2\mathbb{1}_{n\times n}\right)^{-1}\boldsymbol{y}, \boldsymbol{K}_{\boldsymbol{x}^\star\boldsymbol{x}^\star} - \boldsymbol{K}_{\boldsymbol{x}^\star\boldsymbol{X}}\left(\sigma_n^2\mathbb{1}_{n\times n} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}\right)^{-1}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{x}^\star}) \tag{9}$$

. To get a better idea of the connection to kernels, since $\boldsymbol{\Sigma}_p$ is a symmetric positive semi definite matrix, it defines an inner product

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\boldsymbol{\Sigma}_p} = \boldsymbol{y}^*\boldsymbol{\Sigma}_p\boldsymbol{x}, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}^N$$

[Wan][page 34] so that

$$(\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'})_{ij} = \langle\Phi(\boldsymbol{w}_i), \Phi(\boldsymbol{w}_i)\rangle_{\boldsymbol{\Sigma}_p} = k(\boldsymbol{w}_i, \boldsymbol{w}_j) \tag{10}$$

where k is the kernel with feature map Φ and inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{\Sigma}_p}$. In fact when $\boldsymbol{W} = \boldsymbol{W}'$ equation 10 is exactly the Gram matrix with said kernel. Thus GPs are another great example of models that take advantage of the kernel trick. We shall see in the coming chapters on how exactly we can compute predictions using observed values.

1.5. **Gaussian Processes for Regression.** We saw in section 1.4 that, unlike most other machine learning models, GPs infer over a distribution of functions $p(f \mid D)$ instead of a vector of parameteric values $p(\boldsymbol{\theta} \mid D)$. Naively, one may attempt to find a suitable f by fixing a class of functions $\mathcal{F}$ and then search over this class to find a function that best represents the data. However, this may not work well if there is not enough richness in $\mathcal{F}$ to represent the data. Instead we choose a suitable f by assigning a prior probability to every possible function using the training data and then to select the function with the highest probability. To keep this computation tractable we only evalute our predicted function at a finite number of points. The prediction itself is found by taking the mean over all functions with respect to the posterior conditioned on the observed data which is assumed to be jointly Gaussian with the input value. This gives rise to Gaussian Process more formally stated in definition 13.

Definition 13 (Gaussian Process)**.** *A Gaussian Process (GP) is a collection of random variables with index set I, such that every finite subset of random variables has a joint Gaussian distribution* [Ras06, Mur12].

A GP is completely characterised by a mean function $m(\boldsymbol{x})$ and a kernel, which in the context of GPs is sometimes called a covariance function, $k(\boldsymbol{x}, \boldsymbol{x'})$ on a real process as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}\left[f(\boldsymbol{x})\right] \\ k(\boldsymbol{x}, \boldsymbol{x'}) &= \mathbb{E}\left[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'}))\right] \end{aligned}$$

GPs define a prior over all possible functions which can be used to create a posterior once enough data has been observed. The prior is used to represent the functions we expect to see before any observations are made. Although defining a prior over all possible function may seem computationally intractable, we actually only need to define a distribution over a finite number of points. Before any observations are made, we typically assume that the mean function is the constant zero function, that is $m\left(\boldsymbol{x}\right) = 0$. A function $f(\boldsymbol{x})$ sampled from a GP with mean $m(\boldsymbol{x})$ and covariance $k(\boldsymbol{x}, \boldsymbol{x'})$ is written as

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'})\right)$$

Since a GP is a collection of random variables it must satisfy the consistency requirement, that is, an observation of a set of variables should not the distribution of any small sub set of the observed values. More specifically if

$$(\boldsymbol{y_1}, \boldsymbol{y_2}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

then

$$\begin{aligned} \boldsymbol{y_1} &\sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_{1,1}}) \\ \boldsymbol{y_2} &\sim \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_{2,2}}) \end{aligned}$$

where $\boldsymbol{\Sigma_{1,1}}$ and $\boldsymbol{\Sigma_{2,2}}$ are the relevant sub matrices. Again, we shall us the notation that for set of data $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n]^\intercal \in \mathbb{R}^{n \times d}$ and $\boldsymbol{W'} = [\boldsymbol{w}'_1, \boldsymbol{w}'_2, \ldots, \boldsymbol{w}'_m]^\intercal \in \mathbb{R}^{n' \times d}$ we use the notation

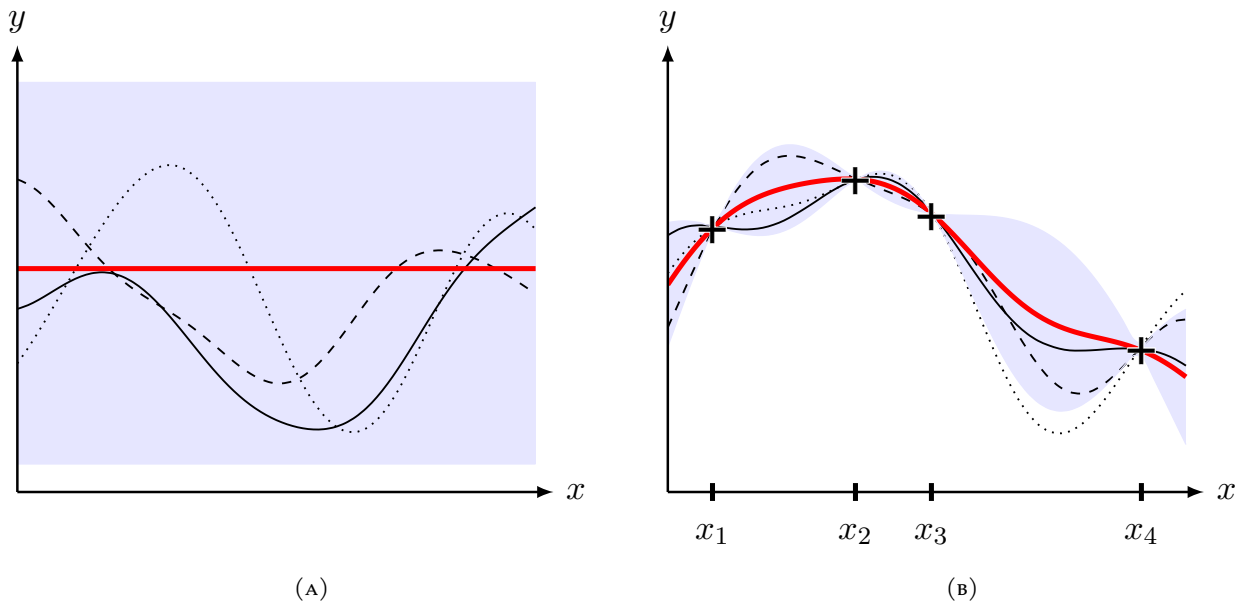
$$\left(\boldsymbol{K_{WW'}}\right)_{i,j} \triangleq k\left(\boldsymbol{w}_i, \boldsymbol{w}'_j\right)$$

where $\boldsymbol{K_{WW'}} \in \mathbb{R}^{n \times n'}$. The convariance function completely characterized by its kernel. Unless otherwise stated, the kernel or covariance function used in examples and experimentation in the Gaussian RBF kernel, definition 11. To understand this better, as a small exercise we can select a number of inputs $\boldsymbol{X^\star} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n^\star}]^\intercal \in \mathbb{R}^{n^\star \times d}$ of compute the corresponding covariance matrix using the Gaussian RBF kernel. Gaussian vectors can then be sampled using a joint Gaussian distribution using the covariance matrix from the distribution

$$\boldsymbol{f^\star} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{K_{X^\star X^\star}}\right)$$

graph its values as a function of its inputs.

Figure 5 (A) shows three samples drawn from the prior before any observations are made. GPs also allow us to compute the pointwise variance which can provide some measure of variability for predicted values. The blue shaded area of figure 5 (A) represents twice the standard deviation about the mean.



(A) (B)

FIGURE 5. Panel (A) shows three function drawn from the prior distribution. Panel (B) shows three function drawn from the prior distribution after four observations have been made. In both panels the mean function is drawn in red, sampled functions in black and twice the standard deviation shaded in light blue.

1.5.1. *Noise-free observations.* Typically when using GP we would like to incorporate data from observations, or training data, into our predictions on unobserved values. Let us suppose there is some obsevered data $D = \{(\boldsymbol{x}_i, \boldsymbol{f}_i) \mid i \in \{1, 2, \dots, n\}\}$ which is (unrealistically) noise-free that we would like to model as a GP. In other words, for any sample in our dataset we can be certain that the observed value is the true value of the underlying function we wish to model. Then for the observed data

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}).$$

We would then like to make a prediction for unobserved values say $X^\star = \left[\boldsymbol{x}_1^\star, \boldsymbol{x}_2^\star, \dots, \boldsymbol{x}_{n_\star}^\star\right]$ with value $f_\star$ as has a distribution of

$$\boldsymbol{f}_\star \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star}).$$

where $\boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} = k(\boldsymbol{X}^\star, \boldsymbol{X}^\star) \in \mathbb{R}^{n_\star \times n_\star}$. Here $\boldsymbol{f}$ and $\boldsymbol{f}_\star$ are independent but we would like to give them some sort of correlation. We can do this by having them originate from the same joint distribution. According to the prior, we can write the joint distribution of the training points $\boldsymbol{f}$ and the test points $\boldsymbol{f}_\star$ as

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} & \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star} \\ \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal & \boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} \end{bmatrix}\right)$$

where $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star} = k(\boldsymbol{X}, \boldsymbol{X}^\star) \in \mathbb{R}^{n \times n_\star}$.

While the above does give us some information on $\boldsymbol{f}_\star$ is related to the observed data and the test inputs, it does not provide any method to evalute $\boldsymbol{f}_\star$. To do this we shall need the assistance of the following lemma

Theorem 14. (*Marginals and conditionals of an MVN* [Mur12]) *Suppose* $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ *is jointly Gaussian with parameters*

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2} \end{bmatrix}$$

then the posterior conditional is given by

$$\begin{aligned} \boldsymbol{x}_2 \mid \boldsymbol{x}_1 &\sim \mathcal{N}\left(\boldsymbol{x}_2 \mid \boldsymbol{\mu}_{2|1}, \boldsymbol{\Sigma}_{2|1}\right) \\ \boldsymbol{\mu}_{2|1} &= \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}\left(\boldsymbol{x}_1 - \boldsymbol{\mu}_1\right) \\ \boldsymbol{\Sigma}_{2|1} &= \boldsymbol{\Sigma}_{2,2} - \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}\boldsymbol{\Sigma}_{1,2} \end{aligned}$$

Thus finding a mean an covariance for $\boldsymbol{f}_\star$ requires a direct application of Theorem 14 which gives

$$\boldsymbol{f}_\star \mid \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}, \boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star\right)$$

where

$$\begin{aligned} \boldsymbol{\mu}^\star &= \boldsymbol{0} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1}\left(\boldsymbol{f} - \boldsymbol{0}\right) \\ &= \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1}\boldsymbol{f} \end{aligned}$$

and

$$\boldsymbol{\Sigma}^\star = \boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} - \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}$$

meaning we can write a distribution for $\boldsymbol{f}_\star$ as

$$\boldsymbol{f}_\star \mid \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}, \boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1}\boldsymbol{f}, \boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} - \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}\right) \tag{11}$$

Function values from the unobserved inputs $\boldsymbol{X}^\star$, that is $\boldsymbol{f}_\star$, can be estimated using the joint posterior distribution by evaulting the mean of 11. Figure 5 (B) shows these computations given a data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$. Notice that the variance tightens around the observed values since (assuming no noise in our data is present) we know for certain this is how our target function should behave at x_1, x_2, x_3 and x_4. Specifying the properties of the prior is important since it fixes the properties of the functions considered during inference.

1.5.2. *Prediction with Noisy observations.* When attempting to model our value function we usually do not have access to the value function itself but a noisy version thereof, $y = f(\boldsymbol{x}) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ meaning the prior on the noisy values becomes

$$\operatorname{cov}(\boldsymbol{y}) = \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \boldsymbol{I}$$

The reason why noise is only added along the diagonal follows from the assumption of independence in our data. We can write out the new distribution of the observed noisy values along the points at which we wish to test the underlying function as

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \mathbb{1}_{n\times n} & \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star} \\ \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal & \boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} \end{bmatrix}\right)$$

Using a similar we arrive at a similar condition distribution of $\boldsymbol{f}_\star \mid \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}, \boldsymbol{f}$ we arrive at one of the most fundamental equations for GP regression tasks

$$\boldsymbol{f}_\star \mid \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}, \boldsymbol{y} \sim \mathcal{N}\left(\overline{\boldsymbol{f}_\star}, \operatorname{cov}(\boldsymbol{f}_\star)\right) \tag{12}$$

where

$$\overline{\boldsymbol{f}_\star} \triangleq \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \left[\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \mathbb{1}_{n\times n}\right]^{-1} \boldsymbol{y} \tag{13}$$

$$\operatorname{cov}(\boldsymbol{f}_\star) = \boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star} - \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \left[\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \mathbb{1}_{n\times n}\right]^{-1} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star} \tag{14}$$

Remarkably, this gives us the the exact same posterior distribution ascertained from the weight space derivation in equation 9. Notice that the prediction of the mean in equation 13 is a linear combination of the observations, somtimes referred to as a *linear predictor*. Another way of looking at the prediction is seeing it as a linear combination of n kernel evaluations centered at the input $\boldsymbol{x}^\star$

$$\boldsymbol{f}_\star = \sum_{i=1}^{n} \alpha_i k\left(\boldsymbol{x}_i, \boldsymbol{x}^*\right)$$

where $\boldsymbol{\alpha} = \left[\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \mathbb{1}_{n\times n}\right]^{-1} \boldsymbol{y}$. Intuitively, this expression can be understood by realising that despite defining the GP defining a joint Gaussian distribution over the observations when making predictions GPs only require the $(n+1)$-dimension distribution defined by the n observations and the single test point. Marginalising by taking the relevant submatrix block of the covariance matrix yields our desired 1-dimensional prediction.

Also notice that the covariance does not depend on observations but scales quadratically to the norm of the testing inputs. This is a key feature of GPs. The variance is comprised of the difference between the prior covariance, $\boldsymbol{K}_{\boldsymbol{X}^\star\boldsymbol{X}^\star}$, and positive term $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}^\intercal \left[\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \mathbb{1}_{n\times n}\right]^{-1} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}^\star}$ which represents knowledge given by the observations about the underlying function.

Algorithm 1 shows one possible implementation for computing the mean and covariance of a single test input.

Algorithm 1: Unoptimized GPR

input : Observations $\boldsymbol{X}, \boldsymbol{y}$ and a test input $\boldsymbol{x}^\star$.
output: A prediction $\overline{f_\star}$ with its corresponding variance $\mathbb{V}[f_\star]$.

$\boldsymbol{L} = \text{cholesky}\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_n^2 \mathbb{1}_{n\times n}\right)$
$\boldsymbol{\alpha} = \text{lin-solve}\left(\boldsymbol{L}^\intercal, \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{y}\right)\right)$
$\overline{f_\star} = \boldsymbol{K}_{\boldsymbol{x}^\star\boldsymbol{X}}\boldsymbol{\alpha}$
$\boldsymbol{v} = \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{K}_{\boldsymbol{x}^\star\boldsymbol{X}}\right)$
$\mathbb{V}[f_\star] = \boldsymbol{K}_{\boldsymbol{x}^\star\boldsymbol{x}^\star} - \boldsymbol{v}^\intercal\boldsymbol{v}$
return $\overline{f_\star}, \mathbb{V}[f_\star]$

A cholesky decomposition is typically used since $\boldsymbol{L}$ can be used twice to assist in solving both the linear systems in the mean and covariance. Unfortunately, a cholesky decomposition incurres a runtime of $\mathcal{O}\left(n^3\right)$ where n is the number of samples making it impractical for large data sets. In the later chapters we shall consider other methods for solving these linear systems.

1.6. **Gaussian Processes for Classification.** As with most classifications model, the Guassian processes classifier (GPC) seeks an estimate for the joint probability $p(y, \boldsymbol{x})$ where $\boldsymbol{x} \in \mathbb{R}^d$ is an input, as in the regression case, but y is now a class taking on a discrete and fintie number of values $\{\mathcal{C}_i\}_{i=1}^C$. Using Baye's theorem the joint probability density can be decomposed into either $p(y)\, p(\boldsymbol{x} \mid y)$ or $p(\boldsymbol{x})\, p(\boldsymbol{y} \mid \boldsymbol{x})$ giving rise to the *generative* and *discriminative* approaches respectively [Ras06, page 34]. The generative approach models the prior probabilities of each class, $p(\mathcal{C}_i)$, as well as the class conditional probabilities for each input $p(\boldsymbol{x} \mid \mathcal{C}_i)$ and computes the posterior as

$$p(y \mid \boldsymbol{x}) = \frac{p(y)\, p(\boldsymbol{x} \mid y)}{\sum_{i=1}^C p(\mathcal{C}_i)\, p(\boldsymbol{x} \mid \mathcal{C}_i)}.$$

On the other hand, the discriminative method focuses on modelling $p(y \mid \boldsymbol{x})$ directly. With both these paradigms at our disposal, which one should we prefer for our GPC? While there are strengths and weaknesses associated with both models, the discriminative approach is usually chosen as it has a rather attractive property of directly modeling what we want, that is, $p(y \mid \boldsymbol{x})$. Also the density estimation of $p(\boldsymbol{x} \mid \mathcal{C}_i)$ using in the generative model presents a number of difficulties, especially for larger values of d. If we are only focused on classifying inputs, the generative approach may mean we are trying to solve a harder problem than what we need to. For this reason we focus on GPCs that adopt the discriminative approach.

1.6.1. *Linear Models for Classification.* We can start by reviewing linear models for the simplist form of classification, that is binary classification. Adopting the notation from SVM (see section 1.4.1) literature the binary classification problem involves assigning an input $\boldsymbol{x}$ to a class of either -1 or $+1$. For a linear model likelihood can be formulated as

$$p(y = +1 \mid \boldsymbol{x}, \boldsymbol{w}) = \sigma(\langle \boldsymbol{x}, \boldsymbol{w} \rangle) \tag{15}$$

given a weight vector $\boldsymbol{w}$ and where $\sigma(\boldsymbol{z})$ is chosen to be any sigmoid function, see definition 15.

Definition 15 (Sigmoid Function)**.** *A sigmoid function is a monotonically increasing function mapping from* $\mathbb{R}$ *to* $[0, 1]$ [Ras06, page 35].

Typically the logistic function

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{16}$$

is used to take the role of the sigmoid function in equation 15. This type of model is aptly named the logistic regression. Unlike GPR, the likelihood is no longer a Gaussian distribution. Instead it follows the Bernoulli distribution

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \sigma(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)^y \left(1 - \sigma(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)\right)^{\frac{1-y}{2}}$$

which for symmeteric likelihood functions can be written more concisely as

$$p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) = \sigma(y_i f_i)$$

where

$$f_i \triangleq f(\boldsymbol{x}_i) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle. \tag{17}$$

Thus the logistic regression model can be written as the log ratio of the likelihoods of the input belonging to either class, that is

$$\operatorname{logit}(\boldsymbol{x}) \triangleq \langle \boldsymbol{x}, \boldsymbol{w} \rangle = \log\left(\frac{p(y=+1)}{p(y=-1)}\right)$$

where logit is commonly referred to as the logit transformation. For a given dataset $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^\intercal \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^\intercal \in \{-1, +1\}^n$ we assume each observation is independently generated conditioned over $f(\boldsymbol{x})$. Similar to GPR, a Gaussian prior is used for the weights so that $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma_p)$ giving an un-normalised log posterior of

$$\log p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) \propto -\frac{1}{2}\boldsymbol{w}^\intercal \Sigma_p^{-1} \boldsymbol{w} + \sum_{i=1}^{n} \log \sigma(y_i f_i).$$

However, unlike GPR an analytic form for the mean and variance for the posterior is not available due to the non-Gaussian nature of the likelihood. However, when using the logistic function it is easy enough to show that the log likelihood is concave as a function of $\boldsymbol{w}$ for a fixed dataset. This means a number of numerical optimization techniques, such as Newton's method or the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [Fle00].

Gaussian processes classification on binary classes, the basic idea is that a Gaussian process regression model is place over a latent function $f(\boldsymbol{x})$ with the output being "squashed" through a sigmoid function to obtain a prior on

$$\pi(\boldsymbol{x}) \triangleq p(y=+1 \mid \boldsymbol{x}) = \sigma(f(\boldsymbol{x})).$$

This constructsion is illustrated in figure and provides a natural extension to the linear logistic regression model. Specifically, the linear model from equation 17 is replaced with a GPR model and the Gaussian prior on the weights with a GPR weight prior.

»» ***RE WORD***
We have tacitly assumed that the latent Gaussian process is noise-free, and combined it with smooth likelihood functions, such as the logistic or probit. However, one can equivalently think of adding independent noise to the latent process in combination with a step-function likelihood. In particular, assuming Gaussian noise and a step-function likelihood is exactly equivalent to a noisefree latent process and probit likelihood.

The latent function f plays the role of a *nuisance function* we do not observe values of f and we are not particularly interested in the values of f, but rather in π, in particular for test cases $\pi(\boldsymbol{x}_\star)$. The purpose of f is solely to allow a convenient formulation of the model, and the computational goal to pursued in the coming sections will be to remove (integrate out) f.
»» ***RE WORD***
Subsequently, predictions for $\pi_\star = \pi(\boldsymbol{x}_\star)$ are made by average over all possible latent functions weighted by the posterior giving the prediction

$$\overline{\pi_\star} \triangleq p(y_\star = +1 \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_\star) = \int \sigma(f_\star)\, p(f_\star \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_\star)\ df_\star \tag{18}$$

While this is a sound model, computing predictions is not so straight forward since the integral in 18 is not analytically tractable for the same reason as the linear binary classifier. In this coming sections we

will see how we can make use of our numerical toolbox to instead come up with a good approximation for $\overline{\pi_\star}$.

1.6.2. *Lapace Approximation for Posterior.* We saw that the integral in 18 used to make predictions for $\overline{\pi_\star}$ could not be done analytically. In this section we shall address how the distribution for the latent process, $p\left(f_\star \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_\star\right)$, can be numerically approximated to provide a numerically tractable succedaneum. Using Baye's theorem

$$\begin{aligned} p\left(f_\star \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_\star\right) &= \int p\left(f_\star, \boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_\star\right)\ d\boldsymbol{f} \\ &= \frac{1}{p\left(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{x}_\star\right)} \int p\left(f_\star \mid \boldsymbol{X}, \boldsymbol{x}_\star, \boldsymbol{f}\right) p\left(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{x}_\star\right) p\left(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{x}_\star, \boldsymbol{f}\right)\ d\boldsymbol{f} \\ &= \int p\left(f_\star \mid \boldsymbol{X}, \boldsymbol{x}_\star, \boldsymbol{f}\right) p\left(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{x}_\star, \boldsymbol{y}\right)\ d\boldsymbol{f} \end{aligned}$$

using the fact that $p\left(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{x}_\star, \boldsymbol{f}, f_\star\right) = p\left(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{x}_\star, \boldsymbol{f}\right)$ [Bis06, page 315].

2. Random Fourier Features

We saw in section 1 that GPs relied heavily on the Gram matrix (see definition 5) to create predictions based on training data $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^\intercal \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^\intercal \in \mathbb{R}^n$. Unfortunately, the size of the Gram matrix scales quadratically with the number of samples making it difficult to train using data sets with more than 10^5 samples. Instead the kernel function itself can be factorized allowing one to convert training and kernel evaluation into the corresponding operations of a linear machine by mapping data into a relatively low-dimensional randomized feature space. This idea was first introduced by Rahimi and Recht [Rah08] where they propose that instead of using a kernel function to implicitly lift data into a higher dimensional feature space, an explicit feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^D$ can be used to approximate k as $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle_{\mathbb{R}^N} \approx \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle_{\mathbb{R}^D}$ where D is chosen so that $n \gg D$. Thus once $\varphi(\boldsymbol{x}_i)$ has been computed for each $\boldsymbol{x}_i$, each entry of the Gram matrix can be swiftly approximated as

$$\boldsymbol{K}_{ij} = \boldsymbol{K}_{ji} \approx \langle \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{y}_j) \rangle_{\mathbb{R}^D}.$$

Already there have been numerous applications of this technique in GPs that have seen improved time performance with little loss in prediction accuracy [Pot21].

2.1. Theory and Computation.

Contrary to the kernel trick the Random Fouier Features (RFF) technique approximates $\langle \Phi(\cdot), \Phi(\cdot) \rangle_{\mathbb{R}^N}$ through an explicit feature mapping φ. The RFF techniques hinges on Bochners theorem stated without proof in theorem 16 which characterises positive definite functions.

Theorem 16 (Bochner's)**.** *A continuous and shift-invariant function $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = k(\Delta)$ is positive definite (see definition 3) if and only if it can be represented as*

$$k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i \langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y} \rangle\right) \mu_k(d\boldsymbol{\omega})$$

where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$ [Hah33, Liu21].

The spectral distribution μ_k can be represented as finite measure induced by the Fourier transformation. Choosing a kernel for which $k(\boldsymbol{0}) = 1$ normalizes μ_k to a probability distribution $p(\cdot)$. For instance, the spectral distribution of the Gaussian RBF kernel is

$$p(\boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^D \left| \frac{\sigma^2}{2} \mathbb{1}_{D \times D} \right|}} \exp\left(-\frac{1}{2} \boldsymbol{w}^\intercal \left(\frac{\sigma^2}{2} \mathbb{1}_{D \times D} \right)^{-1} \boldsymbol{w} \right) \tag{19}$$

[Rah08, page 3]. One caveat in Bochner's theorem is that it requires our kernel to be shift-invariant (sometimes also referred to as stationary) stated in definition 17.

Definition 17 (Shift-Invariant)**.** *A kernel $k : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{C}$ is called shift-invariant if $k(\boldsymbol{x}, \boldsymbol{y}) = g(\boldsymbol{x} - \boldsymbol{y})$ for some positive definite function $g : \mathbb{R}^N \to \mathbb{C}$* [HAe16, page 3].

Clearly the Gaussian RBF kernel is shift-invariant since it only relies on the bounding radius of $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus from Bochners theorem a positive definite shift-invariant kernel with $k(0) = 1$ can be computed as

$$k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i \langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y} \rangle\right) p(\boldsymbol{\omega}) \, d\boldsymbol{\omega}. \tag{20}$$

The main of idea of RFF is to approximate the integral in 20 using the following Monte-Carlo estimate

$$\begin{aligned} k(\boldsymbol{x}-\boldsymbol{y}) &= \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle\right) p(\boldsymbol{\omega})\, d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega}\sim p(\cdot)}\left(\exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle\right)\right) \\ &\approx \frac{1}{D}\sum_{j=1}^{D} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x}-\boldsymbol{y}\rangle\right) \\ &= \sum_{j=1}^{D}\left(\frac{1}{\sqrt{D}}\exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x}\rangle\right)\right)\overline{\left(\frac{1}{\sqrt{D}}\exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{y}\rangle\right)\right)} \\ &= \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y})\rangle_{\mathbb{C}^D} \end{aligned}$$

where $\boldsymbol{\omega}_i \overset{\text{iid}}{\sim} p(\cdot)$ using the feature map

$$\varphi(\boldsymbol{x}) = \frac{1}{\sqrt{D}}\left[z\left(\boldsymbol{\omega}_1, \boldsymbol{x}\right), z\left(\boldsymbol{\omega}_2, \boldsymbol{x}\right), \ldots, z\left(\boldsymbol{\omega}_D, \boldsymbol{x}\right)\right]^{\mathsf{T}} \tag{21}$$

where for convenience of notation $z(\boldsymbol{\omega}, \boldsymbol{x}) = \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right)$. This allows the Gram matrix to be estimated as $\boldsymbol{K} \approx \widetilde{\boldsymbol{K}} = \boldsymbol{Z}\boldsymbol{Z}^{\mathsf{T}}$ where $\boldsymbol{Z} = [\varphi(\boldsymbol{x}_1), \varphi(\boldsymbol{x}_2), \ldots \varphi(\boldsymbol{x}_D)] \in \mathbb{C}^{n\times D}$ [Rah08, Liu21, HAe16]. To simplify computation, in most settings both $p(\cdot)$ and $k(\Delta)$ are real valued functions meaning $\exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle\right)$ can replaced with its real component $\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle\right)$. The vast majority of literature uses the embeddings Rahimi and Recht provide for $\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle\right)$ where $z(\boldsymbol{\omega}, \boldsymbol{x})$ satisfies equation 20. The first embedding takes the form

$$z(\boldsymbol{\omega}, \boldsymbol{x}) = \left[\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right), \sin\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right)\right]^{\mathsf{T}} \tag{22}$$

which satisfies 20 since

$$\begin{aligned} & z(\boldsymbol{\omega}, \boldsymbol{x})^{\mathsf{T}} z(\boldsymbol{\omega}, \boldsymbol{y}) \\ &= \left[\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right), \sin\left(\langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right)\right]\begin{bmatrix}\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right) \\ \sin\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right)\end{bmatrix} \\ &= \cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right)\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right) + \sin\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle\right)\sin\left(\langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right) \\ &= \frac{1}{2}\left(\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle + \langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right) + \cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle - \langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right)\right) + \\ &\qquad \frac{1}{2}\left(\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle - \langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right) - \cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle + \langle \boldsymbol{\omega}, \boldsymbol{y}\rangle\right)\right) \\ &= \cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle\right). \end{aligned}$$

The other embedding Rahimi and Recht give is

$$z(\boldsymbol{\omega}, \boldsymbol{x}) = \sqrt{2}\cos\left(\langle \boldsymbol{\omega}, \boldsymbol{x}\rangle + b\right) \tag{23}$$

where $b \sim U[0, 2\pi]$. Using a similar argument we can show that this embedding also satisfies 20. However, Sutherland's and Schneider's paper [DJSaJS15] they argue that the Gaussian RBF kernel is better

suited for the embedding given in 22. To summarise their argument we denote

$$\varphi_1(\boldsymbol{x}) = \sqrt{\frac{2}{D}} \begin{bmatrix} \cos\left(\langle \boldsymbol{\omega}_1, \boldsymbol{x} \rangle\right) \\ \cos\left(\langle \boldsymbol{\omega}_2, \boldsymbol{x} \rangle\right) \\ \vdots \\ \cos\left(\langle \boldsymbol{\omega}_{D/2}, \boldsymbol{x} \rangle\right) \\ \sin\left(\langle \boldsymbol{\omega}_1, \boldsymbol{x} \rangle\right) \\ \vdots \\ \sin\left(\langle \boldsymbol{\omega}_{D/2}, \boldsymbol{x} \rangle\right) \end{bmatrix} \tag{24}$$

to be the feature map corresponding to embedding in equation 22 and

$$\varphi_2(\boldsymbol{x}) = \sqrt{\frac{2}{D}} \begin{bmatrix} \cos\left(\langle \boldsymbol{\omega}_1, \boldsymbol{x} \rangle + b_1\right) \\ \vdots \\ \cos\left(\langle \boldsymbol{\omega}_D, \boldsymbol{x} \rangle + b_D\right) \end{bmatrix} \tag{25}$$

to be the feature map corresponding to equation 23. They then show that

$$\begin{aligned} \mathbb{V}\left[\varphi_1(\Delta)\right] &= \frac{1}{D}\left(1 + k(2\Delta) - 2k(\Delta)^2\right) \\ \mathbb{V}\left[\varphi_2(\Delta)\right] &= \frac{1}{D}\left(1 + \frac{1}{2}k(2\Delta) - k(\Delta)^2\right) \end{aligned}$$

meaning the variance of φ_1 is smaller whenever

$$\mathbb{V}\left[\cos\left(\langle \boldsymbol{\omega}, \Delta \rangle\right)\right] = \frac{1}{2} + \frac{1}{2}k(2\Delta) - k(\Delta)^2 \leq \frac{1}{2}.$$

When using the Gaussian kernel,

$$\mathbb{V}\left[\cos\left(\langle \boldsymbol{\omega}, \Delta \rangle\right)\right] = \frac{1}{2}\left(1 - \exp\left(-\frac{2\left\|\Delta\right\|_2^2}{\sigma^2}\right)\right)^2 \leq \frac{1}{2}$$

so that $\varphi_1(\Delta) \leq \varphi_2(\Delta)$ for any $\Delta \in \mathbb{R}^d$. This result is indeed consistent with our preliminary experiments. With this in mind, an embedding of φ_1 is always used for our experimentation.

Another important result Rahimi and Recht show provides a bound on the sup-norm of the difference between a Gram matrix and its RFF approximation stated in proposition 18.

Proposition 18. *Let $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = k(\Delta)$ be a continuous shift-invariant, positive definite function defined on compact subset $\mathcal{M} \subset \mathbb{R}^d$ having radius ℓ where $k(0) = 1$ such that $\nabla^2 k(0)$ exists. Then for the feature mapping defined in equation 24 let $\sigma_p^2 = \mathbb{E}_{\boldsymbol{\omega} \sim p(\cdot)} \left\|\boldsymbol{\omega}\right\|_2^2 = \operatorname{tr} \nabla^2 k(0)$ then for any $\varepsilon \in \mathbb{R}_{>0},\ \varepsilon \leq \sigma_p \ell$ we have*

$$\mathbb{P}\left[\sup_{\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}} \left|\langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle_{\mathbb{R}^D} - k(\boldsymbol{x}, \boldsymbol{y})\right| \geq \varepsilon\right] \leq \alpha\left(\frac{\sigma_p \ell}{\varepsilon}\right)^2 \exp\left(-\frac{D\varepsilon^2}{8(d+2)}\right)$$

where $\alpha \in \mathbb{R}_{>0},\ \alpha < \infty$ does not depend on anything [Rah08, page 3].

Rahimi and Recht prove proposition 18 for $\alpha = 2^8$ although Sutherland and Schneider improve this to $\alpha = 66$ [DJSaJS15, page 3]. Observe that this bound is somewhat determined by the ratio D/d which is why D is often chosen as a multiple of d in experimentation.

These results justify the RFF procedure seen in algorithm 2, which is used to approximate a Gram matrix for the data set $\boldsymbol{X}$ using the feature map from 24.

Algorithm 2: RFF Algorithm

input : $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, the dimension of the feature space D.
output: $\widetilde{\boldsymbol{K}} \approx \boldsymbol{K}$ where $\boldsymbol{K}$ is the Gram matrix corresponding to $\boldsymbol{X}$.

Construct $\boldsymbol{W} \triangleq [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_D]^\intercal \in \mathbb{R}^{D \times d}$ where $\boldsymbol{\omega}_i \overset{\text{iid}}{\sim} p(\cdot)$
$\boldsymbol{Z} = \frac{1}{\sqrt{D}} \left[\cos\left(\boldsymbol{W}\boldsymbol{X}^\intercal\right), \sin\left(\boldsymbol{W}\boldsymbol{X}^\intercal\right)\right]^\intercal$
$\widetilde{\boldsymbol{K}} = \boldsymbol{Z}\boldsymbol{Z}^\intercal$
return $\widetilde{\boldsymbol{K}}$

Algorithm 2 of course assumes an appropriate construction of $\boldsymbol{W}$, commonly called the transformation matrix, and thus has access to a routine which allows one to sample from $p(\cdot)$. When using the RBF Gaussian kernel, the spectral distribution given in equation 19 corresponds to a multivariate Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\left(\frac{\sigma}{\sqrt{2}}\right)^{-2} \mathbb{1}_{D \times D}$. This means $\boldsymbol{W}$ can simply be constructed as $\boldsymbol{W} = \left(\frac{\sigma}{\sqrt{2}}\right)^{-1} [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_D]^\intercal$ where $\boldsymbol{\omega}_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbb{1}_{D \times D})$. Tranformations matrices constructed in this manner are given the notation $\boldsymbol{W}_{\text{RFF}}$. It can be shown that if $\boldsymbol{W}_{\text{RFF}}$ is used as the transformation matrix in algorithm 2 then it produces an unbiased estimate, $\widetilde{\boldsymbol{K}}_{\text{RFF}}$, for the Gram matrix. This stated more precisely in lemma 19.

Lemma 19. *$\widetilde{\boldsymbol{K}}_{RFF}$ is an unbiased estimate of $\boldsymbol{K}$, that is*

$$\mathbb{E}\left[\left(\widetilde{\boldsymbol{K}}_{RFF}\right)_{ij}\right] = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sigma^2}\right)$$

[Yu16, page 3].

Unfortunately, constructing the transformation matrix using $\boldsymbol{W}_{\text{RFF}}$ does not scale to well as the dimension of the feature space increases. Thus the focus of the upcoming sections will be to highlight a few of the more popular alternative methods used in literature for the constructing the transformation matrix.

2.2. **Orthogonal Random Features.** In the previous chapter algorithm 2 assumed some sort of mechanism for producing the transformation matrix $\boldsymbol{W}$. The construction presented in 2.1 involved sampling $\boldsymbol{\omega}_i \overset{\text{iid}}{\sim} p(\cdot)$. For the Gaussian RBF kernel this meant sampling from the multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbb{1}_{D \times D})$. The transformation matrix constructed in this manner was denoted $\boldsymbol{W}_{\text{RFF}}$. Recently, there has been a buzz of activity in literature exploring alternative constructs for the transformation matrix described in section 2.1 [Liu21]. We shall the consider the two methods proposed by Yu *et al.* [Yu16], the first method in this section and the second in the following section. The first method that Yu *et al.* presents is the Orthogonal Random Features (ORF) method with imposes orthogonality on the transformation matrix. To do this a Gaussian matrix $\boldsymbol{G} \in \mathbb{R}^{D \times d}$ is first produced, much like in $\boldsymbol{W}_{\text{RFF}}$. An orthogonal matrix $\boldsymbol{Q}$ is then created by taking the QR-factorization (see section 3.2) of $\boldsymbol{G}$. However, the random orthogonal matrix, $\boldsymbol{Q}$, will not give an unbiased estimate of the kernel matrix. To fix this, we

shall need the assistance of the following common probabilistic identity

$$\|\boldsymbol{z}\|_2^2 \sim \chi_k^2, \text{ where } \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \mathbb{1}_{k\times k})$$

where χ_k^2 is the chi-squared distribution with k degrees of freedom [Bro91, page 41]. This identity is easy enough to show by equating a shared moment generating function of $(1-2t)^{-\frac{k}{2}}$ for $t < \frac{1}{2}$. Taking the square root of both sides gives $\|\boldsymbol{z}\|_2 \sim \chi_k$ where χ_k is the chi distribution with k degrees of freedom. In the RFF method, each $\boldsymbol{\omega}_i \in \mathbb{R}^D$ was independently taken from the multivariate normal Gaussian distribution meaning that using the identity provided above $\|\boldsymbol{\omega}_i\|_2 \sim \chi_D$. The ORF method augments $\boldsymbol{Q}$ by scaling its rows by iid χ_D values which can be done by right multiplying by the matrix $\boldsymbol{S} = \text{diag}(\psi_1, \psi_2, \ldots, \psi_D)$ where $\psi_i \overset{\text{iid}}{\sim} \chi_D$. This means

$$\left\|(\boldsymbol{SQ})_{(i)}\right\|_2 = \left\|\psi_i \boldsymbol{Q}_{(i)}\right\|_2 = \psi_i \sim \chi_D$$

so that the row norms of $\boldsymbol{G}$ and $\boldsymbol{SQ}$ have the same distribution. Thus the transformation matrix for the ORF method is

$$\boldsymbol{W}_{\text{ORF}} = \left(\frac{\sigma}{\sqrt{2}}\right)^{-1} \boldsymbol{SQ}. \tag{26}$$

The main downside the the ORF method is that the QR-factorization brings a computational cost of $\mathcal{O}(Dd)$. Fortunately when using $\boldsymbol{W}_{\text{ORF}}$ as our transformation matrix in algorithm 2 the approximate Gram matrix $\widetilde{\boldsymbol{K}}_{\text{RFF}}$ is an unbiased estimate of $\boldsymbol{K}$, stated more formally in theorem 20.

Theorem 20. *$\widetilde{\boldsymbol{K}}_{ORF}$ is an unbiased estimate of $\boldsymbol{K}$, that is*

$$\mathbb{E}\left[\left(\widetilde{\boldsymbol{K}}_{ORF}\right)_{ij}\right] = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sigma^2}\right)$$

[Yu16, page 3].

Furthermore, the variance of $\left(\widetilde{\boldsymbol{K}}_{\text{ORF}}\right)_{ij}$ is bounded by

$$\mathbb{V}\left[\left(\widetilde{\boldsymbol{K}}_{\text{ORF}}\right)_{ij}\right] - \mathbb{V}\left[\left(\widetilde{\boldsymbol{K}}_{\text{RFF}}\right)_{ij}\right] = \frac{1}{D}\left(\frac{g(\tau)}{d} - \frac{(d-1)e^{-\tau^2}\tau^4}{2d}\right)$$

where $\tau = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 / \frac{\sigma}{\sqrt{2}}$ and

$$g(\tau) = \frac{e^{\tau^2}\left(\tau^8 + 6\tau^6 + 7\tau^4 + \tau\right)}{4} + \frac{e^{\tau^2}\tau^4\left(\tau^6 + 2\tau^4\right)}{2d}$$

[Liu21, page 8]. This shows that there are scenarios for which $\mathbb{V}\left[\left(\widetilde{\boldsymbol{K}}_{\text{ORF}}\right)_{ij}\right] < \mathbb{V}\left[\left(\widetilde{\boldsymbol{K}}_{\text{RFF}}\right)_{ij}\right]$, namely when d is large and τ is small. Also, the ratio in variance between $\widetilde{\boldsymbol{K}}_{\text{ORF}}$ and $\widetilde{\boldsymbol{K}}_{\text{RFF}}$ for large d can be approximated as

$$\frac{\mathbb{V}\left[\left(\widetilde{\boldsymbol{K}}_{\text{ORF}}\right)_{ij}\right]}{\mathbb{V}\left[\left(\widetilde{\boldsymbol{K}}_{\text{RFF}}\right)_{ij}\right]} \approx 1 - \frac{(s-1)e^{-\tau^2}\tau^4}{d(1-e^{-\tau^2})^2}$$

[Liu21, page 8].

2.3. **Random Ortho-Matrices and Structured Orthogonal Random Matrices.** The second method we shall consider for producing a transformation matrix also originates from Yu's *et al.* paper, which Choromanski *et al.* [Cho17] generalizes as Random Ortho-Matrices (ROM). This second class of methods is motivated by creating transformation matrices with the same variance reductions as ORF with the added benefit of time and memory savings. The transformation matrices generated using ROM take the form

$$\boldsymbol{W}_{\text{ROM}} = \sqrt{d}\prod_{i=1}^{k} \boldsymbol{S}\boldsymbol{D}_i \tag{27}$$

where $\boldsymbol{S} \in \mathbb{R}^{D\times D}$ has orthogonal rows and $\boldsymbol{D} = \operatorname{diag}(\delta_1, \ldots, \delta_D) \in \mathbb{R}^{D\times D}$ where $\delta_i \overset{\text{iid}}{\sim} U(\{-1, 1\})$. This matrix can be forced into a $\mathbb{R}^{D\times d}$ sized matrix by simply extracting the first d columns of $\boldsymbol{D}_1$. The matrix to take the role of $\boldsymbol{S}$ in virtually every application of ROM is the Hadamard matrix, defined in 21, which admits a fast $m\log(n)$ matrix multiplication for a size $m \times n$ Hadamard matrix called the Fast Walsh-Hadamard transform (FWHT) [FaA76].

Definition 21 (Hadamard Matrix)**.** *The Hadamard matrix $\boldsymbol{H}_i \in \mathbb{R}^{\left(2^{i-1}\times 2^{i-1}\right)}$ is defined recursively as*

$$\boldsymbol{H}_i = \begin{cases} [1] & , i = 1 \\ \frac{1}{\sqrt{2}}\begin{bmatrix} \boldsymbol{H}_{i-1} & \boldsymbol{H}_{i-1} \\ \boldsymbol{H}_{i-1} & -\boldsymbol{H}_{i-1} \end{bmatrix} & , i > 1 \end{cases}.$$

Note that while Hadamard matrices are only defined for dimensions of exact powers of 2, although other sizes can be constructed by removing portions of the matrix given in definition 21 or by padding with 0. This gives a concrete means for which one can generate a transformation matrix

$$\sqrt{d}\prod_{i=1}^{k} \boldsymbol{H}\boldsymbol{D}_i \tag{28}$$

where $\boldsymbol{H}$ is an appropriately sized Hadamard matrix. It is easy to check that the matrix generated by equation 28 shares the same expected rows norm lengths as $\boldsymbol{W}_{\text{ORF}}$ and thus enjoys the same variance reduction benefits. Moreover, since matrix multiplication with $\boldsymbol{H}$ can be performed in $\mathcal{O}(D\log(d))$ time (using FWHT) and multiplication with $\boldsymbol{H}$ can be performed in $\mathcal{O}(D)$ time, the SORF method has the added benefit of improved run time complexity $\mathcal{O}(D\log(d))$ using only $\mathcal{O}(D)$ extra memory. Table 1 gives a comparison of the time and space complexities for the methods mentioned so far.

Method	*Time*	*Extra Space*
RFF [Rah08]	$\mathcal{O}(Dd)$	$\mathcal{O}(Dd)$
ORF [Yu16]	$\mathcal{O}(Dd)$	$\mathcal{O}(Dd)$
ROM (SORF) [Cho17, Yu16]	$\mathcal{O}(D\log(d))$	$\mathcal{O}(D)$

TABLE 1. A comparison of various methods for computing a suitable transformation matrix with the Random Fourier Features paradigm. Typically the dimension of the feature space, D, is chosen as some multiple of the dimension of data, d.

Despite the wide use of the ROM method in various machine learning tasks [Cho17, And15, Cho20, Liu21] a number of high-interest theoretical properties remain unsolved problems, leaving many aspects

of this method shrouded in mystery. Instead, much of what we understand about ROM's estimate capabilities comes from empirical analysis. Nonetheless, we shall still cover a smaller number of important results that have been established.

Choromanski *et al.* [Cho17] show that there is diminishing returns (estimate wise) for choosing larger values of k in equation 28. They also show that choosing odd values of k in 28 provide better estimates then its even $k-1$ and $k+1$ counterparts. For this reason a k value of 3 is usually chosen which gives rise to the transformation matrix estimate given in equation 29. The method for constructing transformation matrices in this manner is referred to as Structured Orthogonal Random Features (SORF).

$$\boldsymbol{W}_{\text{SORF}} = \sqrt{d}\boldsymbol{H}\boldsymbol{D}_3\boldsymbol{H}\boldsymbol{D}_2\boldsymbol{H}\boldsymbol{D}_1 \tag{29}$$

This is the same transformation matrix estimate that Yu *et al.* provides. Unfortunately using the SORF method in algorithm 2 does not produce an unbiased estimate of the Gram matrix, however it does satisfy an asymptotic unbiased property

$$\left| \mathbb{E}\left[\left(\widetilde{\boldsymbol{K}}_{\text{SORF}} \right)_{ij} \right] - \mathbb{E}\left[\left(\widetilde{\boldsymbol{K}}_{\text{RFF}} \right)_{ij} \right] \right| \leq \frac{6\tau}{\sqrt{d}}$$

where τ is again $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 / \frac{\sigma}{\sqrt{2}}$ [Liu21, page 8].

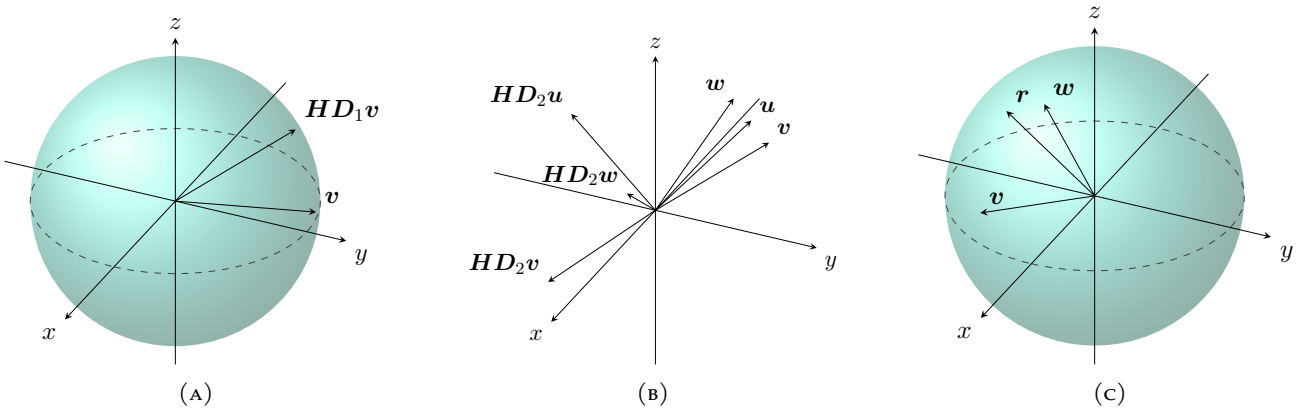


FIGURE 6. A visual representation for the roles of each matrix block in the SORF method. The first block $\boldsymbol{H}\boldsymbol{D}_1$ rotates $\boldsymbol{v}$ so that single dimension bears too much of the l^2 norm seen in (A). In (B) the second block $\boldsymbol{H}\boldsymbol{D}_2$ transforms vectors so that their image is near-orthogonal. Figure (C) shows that the projection of a random vector $\boldsymbol{r}$ onto two near-orthogonal vector $\boldsymbol{v}, \boldsymbol{w}$ yields a near-independent vector.

Bojarski *et al.* [Boj16, page 4] give an intuitive explanation for the roles of each of the different blocks $\boldsymbol{H}\boldsymbol{D}_1$, $\boldsymbol{H}\boldsymbol{D}_2$ and $\boldsymbol{H}\boldsymbol{D}_3$. The first block can be shown to satisfy

$$\mathbb{P}\left[\|\boldsymbol{H}\boldsymbol{D}_1\boldsymbol{x}\|_\infty > \frac{\log D}{\sqrt{D}} \right] \leq 2d \exp\left(-\frac{\log^2 D}{8} \right), \quad \boldsymbol{x} \in \mathbb{R}^D$$

[Liu21, page 8] so that it can be thought as a "balancer" leaving no single dimension bearing too much of the l^2 norm. For the second block, the cost of using a structured matrix is the loss of independence. The purpose of the second block is to mitigate this effect by making similar input vectors near-orthogonal.

Finally the third block controls the capacity of the entire structure by providing a vector of parameters. Near-independence is now implied by the near-orthogonality (achieved by $\boldsymbol{HD}_2$) and the fact that the projections of the Gaussian vector or Radamacher vector onto "almost orthogonal directions" are "close to independent". These roles are portrayed visually in figure 6.

3. Krylov Subspace Methods

In this section we will focus on how iterative methods, in particular a class of iterative methods called Krylov Subspace methods, may be used to solve a linear system $\boldsymbol{Ax} = \boldsymbol{b}$. While non-iterative methods exist to solve such systems virtually all of them carry an unwieldy runtime of $\mathcal{O}\left(n^3\right)$ for a system of n parameters. Even for current computer systems, this renders many common matrix problems untractable. Consequently the focus of solving linear systems has shifted towards iterative methods. While iterative methods typically demand certain structural properties of the matrices, such as symmetry and positive definiteness, this generally is not a problem since the majority of large matrix problems that, by mature, endow these systems with the desired properties. For example, in the context of this paper the Gram matrices used to solve linear systems in Gaussian Processes possess both symmetry and positive definiteness. There are also a number of other properties of iterative methods which make them rather attractive to users. To start, iterative Krylov subspace methods are guranteed to converge to an exact solution within a finite number of iterations and even if the method is prematurely stopped before reaching an exact solution, the approximation obtained on the final iteration will in some sense be a good enough estimate of our exact solution. Furthermore, unlike most non-iterative methods, Krylov subspace methods do not require an explicit form of the matrix $\boldsymbol{A}$ and instead only requires some routine or process for computing $\boldsymbol{Ax}$.

3.1. Krylov Subspaces.

We will motivate the Krylov subspaces by observing their usefullness in solving linear systems. To this end, consider the problem of solving the linear system

$$\boldsymbol{Ax}^\star = \boldsymbol{b} \tag{30}$$

where no explicit form of $\boldsymbol{A}$ is available and instead one must draw information from $\boldsymbol{A}$ solely through a routine that can evaluate $\boldsymbol{Av}$ for any $\boldsymbol{v}$. How could this routine be utilized in such a manner to provide with a solution to equation 30? Before answering this, consider the following theorem

Theorem 22. *For $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ if $\|\boldsymbol{A}\| = q < 1$ then $\mathbb{1} - \boldsymbol{A}$ is invertible and its inverse admits the following representation*

$$(\mathbb{1} - \boldsymbol{A})^{-1} = \sum_{k=0}^{\infty} \boldsymbol{A}^k.$$

[Ber96]

Consider a matrix for which $\|\boldsymbol{A}\| < 2$, it follows that $\|\mathbb{1} - \boldsymbol{A}\| < 1$ meaning $\mathbb{1} - (\mathbb{1} - \boldsymbol{A})$ is invertible and $\boldsymbol{A}^{-1} = (\mathbb{1} - (\mathbb{1} - \boldsymbol{A}))^{-1} = \sum_{k=0}^{\infty} (\mathbb{1} - \boldsymbol{A})^k$. Thinking back to equation 30 for any $x_0 \in \mathbb{K}^n$ we have

$$\begin{aligned}
\boldsymbol{x}^\star &= \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{A}^{-1}\left(\boldsymbol{Ax}^\star - \boldsymbol{Ax_0} + \boldsymbol{Ax_0}\right) \\
&= \boldsymbol{x_0} + \boldsymbol{A}^{-1}\boldsymbol{r_0} \\
&= \boldsymbol{x_0} + \sum_{k=0}^{\infty} (\mathbb{1} - \boldsymbol{A})^k
\end{aligned}$$

where $\boldsymbol{r_0} = \boldsymbol{Ax}^\star - \boldsymbol{Ax_0}$. A natural question that arises is that can we find a closed form solution of the above equation? To answer this question we need to enlist the help of the Cayley-Hamilton theorem.

Theorem 23 (Cayley-Hamilton)**.** *Let $p_n(\lambda) = \sum_{i=0}^{n} c_i \lambda^i$ be the characteristic polynomial of the matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$, then $p_n(\boldsymbol{A}) = \boldsymbol{0}$.* ***THIS NEEDS A CITATION***

The Cayley-Hamilton theorem implies that

$$\begin{aligned} 0 &= c_0 + c_1\boldsymbol{A} + \ldots + c_{n-1}\boldsymbol{A}^{n-1} + c_n\boldsymbol{A}^n \\ 0 &= \boldsymbol{A}^{-1}c_0 + c_1 + \ldots + c_{n-1}\boldsymbol{A}^{n-2} + c_n\boldsymbol{A}^{n-1} \\ \boldsymbol{A}^{-1} &= \alpha_0 + c_1 + \ldots + \alpha_{n-1}\boldsymbol{A}^{n-2} + \alpha_n\boldsymbol{A}^{n-1} \end{aligned}$$

where $\alpha_i = -c_i/c_0$. This demonstrates that $\boldsymbol{A}^{-1}$ can be represented as a matrix polynomial of degree $n-1$. This means that $\sum_{k=0}^{\infty}(\mathbb{1} - \boldsymbol{A})^k$ indeed possess a closed form solution namely

$$\boldsymbol{x}^\star = \boldsymbol{x_0} + \boldsymbol{A}^{-1}\boldsymbol{r_0} = \alpha_0 + c_1 + \ldots + \alpha_{n-1}\boldsymbol{A}^{n-2} + \alpha_n\boldsymbol{A}^{n-1}.$$

This also shows that $\boldsymbol{x}^\star \in \text{l.s}\left\{\boldsymbol{r_0}, \boldsymbol{A}\boldsymbol{r_0}, \boldsymbol{A}^2\boldsymbol{r_0}, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{r_0}\right\}$. One idea for finding a solution to equation 30 is to use our routine for evaluting $\boldsymbol{A}\boldsymbol{v}$ to iteratively compute new basis elements for the space generated by $\left\{\boldsymbol{r_0}, \boldsymbol{A}\boldsymbol{r_0}, \boldsymbol{A}^2\boldsymbol{r_0}, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{r_0}\right\}$ and at each step carefully choosing a $\boldsymbol{x_k}$ such that $\boldsymbol{x_k}$ approaches $\boldsymbol{x}^\star$, in some form. The subspace constructed using this technique is so important that is has its own name.

Definition 24 (Krylov Subspace)**.** *The Krylov Subspace of order k generated by the matrix $\boldsymbol{A} \in \mathbb{K}^{n\times n}$ and the vector $\boldsymbol{v} \in \mathbb{K}$ is defined as*

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) = \text{l.s}\left\{\boldsymbol{r_0}, \boldsymbol{A}\boldsymbol{r_0}, \boldsymbol{A}^2\boldsymbol{r_0}, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{r_0}\right\}$$

for $k \geq 1$ and $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) = \{\boldsymbol{0}\}$.

For the purposes of solving equation 30 it is of much interest to understand how $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$ grows for larger and larger k since a solution for equation 30 will be present in a Krylov Subspace that cannot be grown any larger. In other words, an exact solution can be constructed once we have extracted all the information from $\boldsymbol{A}$ through multiplication of $\boldsymbol{r_0}$. The following theorem provides information on how exactly the Krylov Subspace grows as k increases.

Theorem 25. *There is a positive called the grade of $\boldsymbol{v}$ with respect to $\boldsymbol{A}$, denoted $t_{\boldsymbol{v},\boldsymbol{A}}$, where*

$$\dim\left(\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})\right) = \begin{cases} k, & k \leq t \\ t, & k \geq t \end{cases}$$

Theorem 25 essentially tells us that for $k \leq t_{\boldsymbol{v},\boldsymbol{A}}$ that $\boldsymbol{A}^k\boldsymbol{v}$ is linearly independent to $\boldsymbol{A}^i\boldsymbol{v}$ for $0 \leq i \leq k-1$ meaning $\left\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{v}\right\}$ serves as a basis for $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$ and that $\mathcal{K}_{k-1}(\boldsymbol{A}, \boldsymbol{v}) \subsetneq \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$. Conversely, any new vectors formed beyond $t_{\boldsymbol{v},\boldsymbol{A}}$ will be linearly independent meaning $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) \subsetneq \mathcal{K}_{k+1}(\boldsymbol{A}, \boldsymbol{v})$ for $k \geq t_{\boldsymbol{v},\boldsymbol{A}}$. While $t_{\boldsymbol{v},\boldsymbol{A}}$ clearly plays a role in determining a suitable basis for which $\boldsymbol{A}^{-1}\boldsymbol{b}$ lies in its importance is made abundantly clear in the following corollary.

Corollary 26.

$$t_{\boldsymbol{v},\boldsymbol{A}} = \min\left\{k \mid \boldsymbol{A}^{-1}\boldsymbol{v} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})\right\}$$

Proof. Recall from Cayley-Hamilton (theorem 23) that

$$\boldsymbol{A}^{-1}\boldsymbol{v} = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{A}^i \boldsymbol{v}$$

But since $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) = \mathcal{K}_{k+1}(\boldsymbol{A}, \boldsymbol{v})$ for $k \geq t_{\boldsymbol{v},\boldsymbol{A}}$

$$\boldsymbol{A}^{-1}\boldsymbol{v} = \sum_{i=0}^{t-1} \beta_i \boldsymbol{A}^i \boldsymbol{v}$$

meaing $\boldsymbol{A}^{-1}\boldsymbol{v} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$ for $k \geq t_{\boldsymbol{v},\boldsymbol{A}}$. Suppose for the sake of contradiction that this also holds for $k = t_{\boldsymbol{v},\boldsymbol{A}} - 1$, that is, $\boldsymbol{A}^{-1}\boldsymbol{v} = \sum_{i=0}^{t-2} \gamma_i \boldsymbol{A}^i \boldsymbol{v}$. However, this gives

$$\boldsymbol{v} = \sum_{i=0}^{t-2} \gamma_i \boldsymbol{A}^{i+1}\boldsymbol{v} = \sum_{i=0}^{t-1} \gamma_{i-1} \boldsymbol{A}^i \boldsymbol{v}$$

implying $\left\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, \ldots, \boldsymbol{A}^{t-1}\boldsymbol{v}\right\}$ are linearly dependent which means that $\dim\left(\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})\right) < t$, which provides us with our contrdiction. □

This machinery allows us to make a much stronger statement on the where abouts of $\boldsymbol{x}^\star$ in relation to the Krylov Subspaces.

Corollary 27. *For any $\boldsymbol{x_0}$, we have*

$$\boldsymbol{x}^\star \in \boldsymbol{x_0} + \mathcal{K}_{t_{\boldsymbol{r_0},\boldsymbol{A}}}(\boldsymbol{A}, \boldsymbol{r_0})$$

where $\boldsymbol{r_0} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x_0}$.

3.2. **Gram-Schmidt Process and QR factorisations.** Many areas of linear algebra involving studing the column space of matrices. The QR factorisation provides us with a powerful tool to better understand the column space of a matrix as well as serving as an important factorisation mechanism for many numerical methods. Suppose that a matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n] \in \mathbb{K}^{n\times n}$ has full rank. The idea of a QR factorisation is to find an alternative orthornormal basis for $(\boldsymbol{a}_i)_{i=1}^n$, say $(\boldsymbol{q}_i)_{i=1}^n$, and to somehow relate the original matrix $\boldsymbol{A}$ to a new matrix whose columns are $(\boldsymbol{q}_i)_{i=1}^n$. Consider the following procedure that allows us to find an orthornormal basis $(\boldsymbol{q}_i)_{i=1}^n$ for which $\mathrm{l.s}\{(\boldsymbol{a}_i)_{i=1}^n\} = \mathrm{l.s}\{(\boldsymbol{q}_i)_{i=1}^n\}$. First set $\boldsymbol{q}_1 = \frac{\boldsymbol{a}_1}{\|\boldsymbol{a}_i\|}$, clearly $\mathrm{l.s}\{\boldsymbol{a}_1\} = \mathrm{l.s}\{\boldsymbol{q}_1\}$. Next, construct a vector $\boldsymbol{q}_2' = \boldsymbol{a}_2 - r_{1,2}\cdot\boldsymbol{q}_1$ so that $\boldsymbol{q}_2' \perp \boldsymbol{q}_1$. This means

$$\begin{aligned} 0 &= \langle \boldsymbol{q}_1, \boldsymbol{q}_2' \rangle \\ 0 &= \langle \boldsymbol{q}_1, \boldsymbol{a}_2 - r_{1,2}\cdot\boldsymbol{q}_1 \rangle \\ 0 &= \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle - r_{1,2}\cdot\langle \boldsymbol{q}_1, \boldsymbol{q}_1 \rangle \\ r_{1,2} &= \langle \boldsymbol{q}_1, \boldsymbol{a}_2 \rangle \end{aligned}$$

Since $\boldsymbol{q}_2'$ may not be a unit vector we set $\boldsymbol{q}_2 = \frac{\boldsymbol{q}_2'}{\|\boldsymbol{q}_2'\|}$ where $\mathrm{l.s}(\{\boldsymbol{a}_1, \boldsymbol{a}_2\}) = \mathrm{l.s}(\{\boldsymbol{q}_1, \boldsymbol{q}_2\})$. Continuing the vector $\boldsymbol{q}_3'$ is constructed so that

$$\boldsymbol{q}_3' = \boldsymbol{a}_3 - \boldsymbol{r}_{1,3}\boldsymbol{q}_1 - \boldsymbol{r}_{2,3}\boldsymbol{q}_2$$

are chosen so that $\boldsymbol{q}_3'$ is orthogonal to both $\boldsymbol{q}_2$ and $\boldsymbol{q}_1$. This amounts to setting $r_{1,3} = \langle \boldsymbol{q}_1, \boldsymbol{a}_3 \rangle$ and $r_{2,3} = \langle \boldsymbol{q}_2, \boldsymbol{a}_3 \rangle$. Similarly, $\boldsymbol{q}_3'$ is normalized so that $\boldsymbol{q}_3 = \frac{\boldsymbol{q}_3'}{\|\boldsymbol{q}_3'\|}$ and $\mathrm{l.s}(\{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3\}) = \mathrm{l.s}(\{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3\})$.

Continuing in this fashion the k^{th} vector in our orthornormal basis is computed as

$$\boldsymbol{q}_k = \frac{\boldsymbol{a}_k - \sum_{i=1}^{k-1} r_{i,k} \cdot \boldsymbol{q}_i}{r_{k,k}} \tag{31}$$

where $r_{i,k} = \langle \boldsymbol{q}_i, \boldsymbol{a}_k \rangle$, $r_{k,k} = \|\boldsymbol{a}_k - \sum_{i=1}^{k-1} r_{i,k} \cdot \boldsymbol{q}_i\|$ and $\mathrm{l.s}\left(\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k\}\right) = \mathrm{l.s}\left(\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k\}\right)$. This procedure is famiously known as the Gram-Schmidt process [Ber96, Tre97, Dem97] and is summarized in the following algorithm.

Algorithm 3: Classical Gram-Schmidt

input : A basis $(\boldsymbol{a}_i)_{i=1}^n$.
output: An orthornormal basis $(\boldsymbol{q}_i)_{i=1}^n$ such that $\mathrm{l.s}\left\{(\boldsymbol{a}_i)_{i=1}^n\right\} = \mathrm{l.s}\left\{(\boldsymbol{q}_i)_{i=1}^n\right\}$

for $k = 1$ **to** n **do**
  $\boldsymbol{q}'_k = \boldsymbol{a}_k$
  for $i = 1$ **to** $k-1$ **do**
    $r_{i,k} = \langle \boldsymbol{q}_i, \boldsymbol{a}_k \rangle$
    $\boldsymbol{q}'_k = \boldsymbol{q}'_k - r_{i,k}\boldsymbol{q}_i$
  end
  $r_{k,k} = \|\boldsymbol{q}'_k\|$
  $\boldsymbol{q}_k = \boldsymbol{q}'_k / r_{k,k}$
end
return $(\boldsymbol{q}_i)_{i=1}^n$

Relating the column space of $\boldsymbol{A}$ to the orthornormal basis $(\boldsymbol{q}_i)_{i=1}^n$ in a matrix form

$$[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots \boldsymbol{a}_n] = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots \boldsymbol{q}_n] \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ & r_{2,2} & & \vdots \\ & & \ddots & \vdots \\ & & & r_{n,n} \end{bmatrix}$$

or more succinctly

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} \tag{32}$$

where $\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots \boldsymbol{q}_n]$ and $(\boldsymbol{R})_{i,j} = r_{i,j}$ for $i \leq j$ and $(\boldsymbol{R})_{i,j} = 0$ for $i > j$. This is exactly the QR factorisation for a full rank matrix. Note that $\mathrm{Range}\,(\boldsymbol{A}) = \mathrm{Range}\,(\boldsymbol{Q})$. In general, any square matrix $\boldsymbol{A} \in \mathbb{K}^{m \times n}$ may be decomposed as $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$ where $\boldsymbol{Q} \in \mathbb{K}^{m \times m}$ is an orthogonal matrix and $\boldsymbol{R} \in \mathbb{K}^{m \times n}$ is an upper triangular matrix. This is known as a full QR factorisation. Since bottom $(m-n)$ rows of this $\boldsymbol{R}$ consists entirely of zeros, it is often useful to partition the full QR factorisation in the following manner to shed vacuous entries

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = \boldsymbol{Q} \begin{bmatrix} \hat{\boldsymbol{R}} \\ \boldsymbol{0}_{(m-n)\times n} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{Q}} & \boldsymbol{Q}' \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{R}} \\ \boldsymbol{0}_{(m-n)\times n} \end{bmatrix} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}.$$

This alternate decomposition is called the reduced (or somtimes the thin) QR-factorization. We shall state the following two theorems on the QR-factorization are stated without proof.

Theorem 28. *Every $\boldsymbol{A} \in \mathbb{K}^{m\times n},\ (m \geq n)$ has a full QR factorisation, hence also a reduced QR factorisation.* [Tre97]

Theorem 29. *Each $\boldsymbol{A} \in \mathbb{K}^{m\times n},\ (m \geq n)$ of full rank has a unique reduced QR factorisation $\boldsymbol{A} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$ with $r_{k,k} > 0$.* [Tre97]

In practice the classical Gram-Schmidt process described in algorithm 3 is rarely used as the procedure becomes numerically unstable if $(\boldsymbol{a}_i)_{i=1}^n$ are almost linearly dependent. Before looking for ways to resolve these numerical instabilities a quick recap of projectors has been devised. A square matrix $\boldsymbol{P}_G$ acting on a Hilbert space H that sends $\boldsymbol{x} \in H$ to its projection onto a subspace G is called the projector onto G. If $(\boldsymbol{q}_k)_{k=1}^m$ is an orthornormal basis in G then

$$\boldsymbol{P}_G = \boldsymbol{Q}\boldsymbol{Q}^*$$

where $\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \dots \boldsymbol{q}_m, 0, \dots, 0] \in \mathbb{K}^{n\times n}$. A special class of projectors which isolates the components of a given vector onto a one dimensional subspace spanned by a single unit vector $\boldsymbol{q}$ called a rank one orthogonal projector, denoted as $\boldsymbol{P}_q$. Each k in the classical Gram-Schmidt process $\boldsymbol{q}'_k$ using the following orthogonal projection

$$\boldsymbol{q}'_k = \boldsymbol{P}_{A_k^\perp}\boldsymbol{a}_k \tag{33}$$

where $A_k = \text{l. s}\,\{\boldsymbol{a}_i\}_{i=1}^k$ and $\boldsymbol{P}_{A_1^\perp} = \mathbb{1}$ for convenience. A modified version of the Gram-Schmidt process performs the same orthogonal projection broken up as $k-1$ orthogonal projections of rank $n-1$ as so

$$\begin{aligned}
\boldsymbol{q}'_k &= \boldsymbol{P}_{A_k^\perp}\boldsymbol{a}_k \\
&= \left(\mathbb{1} - \boldsymbol{Q}_k\boldsymbol{Q}_k^*\right)\boldsymbol{a}_k \\
&= \left(\prod_{i=1}^{k-1}\left(\mathbb{1} - \boldsymbol{q}_i\boldsymbol{q}_i^*\right)\right)\boldsymbol{a}_k \\
&= \left(\mathbb{1} - \boldsymbol{q}_1\boldsymbol{q}_1^*\right)\left(\mathbb{1} - \boldsymbol{q}_1\boldsymbol{q}_1^*\right)\cdots\left(\mathbb{1} - \boldsymbol{q}_{k-1}\boldsymbol{q}_{k-1}^*\right)\boldsymbol{a}_k \\
&= \boldsymbol{P}_{\boldsymbol{q}_k^\perp}\cdots\boldsymbol{P}_{\boldsymbol{q}_1^\perp}\boldsymbol{a}_k
\end{aligned}$$

While its clear that $\boldsymbol{P}_{A_k^\perp}\boldsymbol{a}$ and $\boldsymbol{P}_{\boldsymbol{q}_k^\perp}\cdots\boldsymbol{P}_{\boldsymbol{q}_1^\perp}\boldsymbol{a}_k$ used for computing $\boldsymbol{q}'_k$ are algebraically, they differ arithmetically as the latter expression evaluates $\boldsymbol{q}'_k$ using the follow procedure

$$\begin{aligned}
\boldsymbol{q}_k^{(1)} &= \boldsymbol{a}_k \\
\boldsymbol{q}_k^{(2)} &= \boldsymbol{P}_{\boldsymbol{q}_1^\perp}\boldsymbol{q}_k^{(1)} \\
\boldsymbol{q}_k^{(3)} &= \boldsymbol{P}_{\boldsymbol{q}_2^\perp}\boldsymbol{q}_k^{(2)} \\
&\vdots \\
\boldsymbol{q}'_k = \boldsymbol{q}_k^{(k)} &= \boldsymbol{P}_{\boldsymbol{q}_{k-1}^\perp}\boldsymbol{q}_k^{(k-1)}
\end{aligned}$$

Applying projections sequentially in this manner produces smaller numerical errors. The modified Gram-Schmidt process [Tre97, Dem97] is summarized in the following algorithm.

Algorithm 4: Modified Gram-Schmidt

input : A basis $\{\boldsymbol{a}_i\}_{i=1}^n$.
output: An orthornormal basis $\{\boldsymbol{q}_i\}_{i=1}^n$ such that $\mathrm{l.s}\{\boldsymbol{a}_i\}_{i=1}^n = \mathrm{l.s}\{\boldsymbol{q}_i\}_{i=1}^n$

for $k = 1$ **to** n **do**
  $\boldsymbol{q}'_k = \boldsymbol{a}_k$
end
for $k = 1$ **to** n **do**
  $r_{k,k} = \|\boldsymbol{q}'_k\|$
  $\boldsymbol{q}_k = \boldsymbol{q}'_k / r_{k,k}$
  for $i = k+1$ **to** n **do**
    $r_{i,k} = \langle \boldsymbol{q}_k, \boldsymbol{q}'_i \rangle$
    $\boldsymbol{q}_i = \boldsymbol{q}_i - r_{i,k}\boldsymbol{q}_i$
  end
end
return $\{\boldsymbol{q}_i\}_{i=1}^n$

3.3. **Arnoldi and Lanczos Algorithm.** As a quick reminder, we are in search of an iterative process to solve the linear system $\boldsymbol{A}\boldsymbol{x}^\star = \boldsymbol{b}$ where no explicit form of $\boldsymbol{A}$ is available and we may only rely on a routine that computes $\boldsymbol{A}\boldsymbol{v}$ for any $\boldsymbol{v}$ to extract information on $\boldsymbol{A}$. In section 3.1 it wa discovered that $\boldsymbol{x}^\star \in \mathcal{K}_{t_{\boldsymbol{r}_0,\boldsymbol{A}}}(\boldsymbol{A}, \boldsymbol{r}_0)$. With many iterative methods, computing an exact value for $\boldsymbol{x}^\star$ is out the question with the view that $t_{\boldsymbol{r}_0,\boldsymbol{A}}$ is impractically large. We must instead resort to approximating $\boldsymbol{x}^\star$ by $\boldsymbol{x}_k$ for which $\boldsymbol{x}^k \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$ where $k \ll t_{\boldsymbol{r}_0}$. To find an appropriate value for $\boldsymbol{x}_k$, a good start would be to find a basis $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$. Definition 24 showed us that $\left\{\boldsymbol{A}^{i-1}\boldsymbol{r}_0\right\}_{i=1}^k$ serves as a basis for $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$. However, for numerical reasons this is a poor choice of basis since this each consecutive term becomes closer and closer to being linearly dependent. From now on, for more convenient notation we shall set $n = t_{\boldsymbol{r}_0,\boldsymbol{A}}$ so that $\boldsymbol{x}^\star \in \mathcal{K}_n(\boldsymbol{A}, \boldsymbol{r}_0)$. To search for a more appriporate basis let $\boldsymbol{K} \in \mathbb{K}^{n\times n}$ be the invertible matrix

$$\boldsymbol{K} = \left[\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{r}_0\right].$$

Since $\boldsymbol{K}$ is invertible we can compute $\boldsymbol{c} = -\boldsymbol{K}^{-1}\boldsymbol{A}^n\boldsymbol{r}_0$ so that

$$\boldsymbol{A}\boldsymbol{K} = \left[\boldsymbol{A}\boldsymbol{r}_0, \boldsymbol{A}^2\boldsymbol{r}_0, \ldots, \boldsymbol{A}^n\boldsymbol{r}_0\right]$$
$$\boldsymbol{A}\boldsymbol{K} = \boldsymbol{K}\cdot[\boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n, -\boldsymbol{c}] \triangleq \boldsymbol{K}\boldsymbol{C}$$

or, in a more verbose form

$$\boldsymbol{K}^{-1}\boldsymbol{A}\boldsymbol{K} = \boldsymbol{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_1 \\ 1 & 0 & \cdots & 0 & -c_2 \\ 0 & 1 & \cdots & 0 & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_n \end{bmatrix}.$$

Note here that $\boldsymbol{C}$ is upper Hessenberg. While this form is simple, it is of little practical use since the matrix $\boldsymbol{K}$ is very likely to be ill-conditioned. To remedy this we can replace $\boldsymbol{K}$ with an orthogonal matrix which

spans the same space. These are exactly the properties that the $\boldsymbol{Q}$ matrix offers in the QR-factorisation of $\boldsymbol{K}$. With this in mind let $\boldsymbol{K} = \boldsymbol{QR}$ be the full QR-factorisation of $\boldsymbol{K}$. Then

$$\begin{aligned}\boldsymbol{AQR} &= \boldsymbol{AK}\\ \boldsymbol{AQ} &= \boldsymbol{AKR}^{-1}\\ \boldsymbol{AQ} &= \boldsymbol{KCR}^{-1}\\ \boldsymbol{AQ} &= \boldsymbol{QRCR}^{-1}\\ \boldsymbol{AQ} &\triangleq \boldsymbol{QH}.\end{aligned}$$

Since $\boldsymbol{R}$ and $\boldsymbol{R}^{-1}$ and both upper triangular and $\boldsymbol{C}$ is upper Hessenberg, $\boldsymbol{H}$ is also upper Hessenberg. This form provides us with a $\boldsymbol{Q}$ such that the range of $\boldsymbol{Q}$ is $\mathcal{K}_n(\boldsymbol{A}, \boldsymbol{r}_0)$ and

$$\boldsymbol{Q}^\intercal \boldsymbol{AQ} = \boldsymbol{H}. \tag{34}$$

Again, in practice, it may be very difficult to compute this entire expression forcing us to search for approximative alternatives. Consider equation 34 for which the only first k columns of $\boldsymbol{Q}$ have been computed. Let $\boldsymbol{Q}_k = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k]$ and $\boldsymbol{Q}_u = [\boldsymbol{q}_{k+1}, \boldsymbol{q}_{k+2}, \ldots, \boldsymbol{q}_n]$. Then

$$\begin{aligned}\boldsymbol{Q}^\intercal \boldsymbol{AQ} &= \boldsymbol{H}\\ [\boldsymbol{Q}_k, \boldsymbol{Q}_u]^\intercal \boldsymbol{A} [\boldsymbol{Q}_k, \boldsymbol{Q}_u] &= \begin{bmatrix}\boldsymbol{H}_k & \boldsymbol{H}_{u,k}\\ \boldsymbol{H}_{k,u} & \boldsymbol{H}_u\end{bmatrix}\\ \begin{bmatrix}\boldsymbol{Q}_k^\intercal \boldsymbol{AQ}_k & \boldsymbol{Q}_k^\intercal \boldsymbol{AQ}_u\\ \boldsymbol{Q}_u^\intercal \boldsymbol{AQ}_k & \boldsymbol{Q}_u^\intercal \boldsymbol{AQ}_u\end{bmatrix} &= \begin{bmatrix}\boldsymbol{H}_k & \boldsymbol{H}_{u,k}\\ \boldsymbol{H}_{k,u} & \boldsymbol{H}_u\end{bmatrix}\end{aligned}$$

where $\boldsymbol{H}_k, \boldsymbol{H}_{u,k}, \boldsymbol{H}_{k,u}$ and $\boldsymbol{H}_u$ are the relevant sub matrices. This provides us with the equality

$$\boldsymbol{Q}_k^\intercal \boldsymbol{AQ}_k = \boldsymbol{H}_k \tag{35}$$

noting that $\boldsymbol{H}_k$ is upper Hessenberg for the same reason that $\boldsymbol{H}$ is. We know that when $n = t_{\boldsymbol{r}_0, \boldsymbol{A}}$ we can find a $\boldsymbol{Q} \in \mathbb{K}^{n \times n}$ and $\boldsymbol{H} \in \mathbb{K}^{n \times n}$ that satisfies $\boldsymbol{AQ} = \boldsymbol{QH}$. However, in general, we may not be so fortunate in finding a $\boldsymbol{Q}_k \in \mathbb{K}^{n \times k}$ and $\boldsymbol{H}_k \in \mathbb{K}^{n \times k}$ so satisfy $\boldsymbol{AQ}_k = \boldsymbol{Q}_k \boldsymbol{H}_k$ for any $k < n$. Instead we can adjust this equality by adding an error $\boldsymbol{E}_k \in \mathbb{K}^{n \times k}$ so that we do get equality. Our expression now becomes

$$\boldsymbol{Q}_k^\intercal \boldsymbol{AQ}_k = \boldsymbol{H}_k + \boldsymbol{E}_k. \tag{36}$$

A careful choice of $\boldsymbol{E}_k$ must be made to also retain equality in equation 35, meaning $\boldsymbol{Q}_k^\intercal \boldsymbol{E}_k = \boldsymbol{0}$. Since $\{\boldsymbol{q}_i\}_{i=1}^k$ forms an orthornormal basis for $\mathcal{K}_n(\boldsymbol{A}, \boldsymbol{r}_0)$, consider the following choice of $\boldsymbol{E}_k$,

$$\boldsymbol{E}_k = \boldsymbol{q}_{k+1} \boldsymbol{h}_k^\intercal$$

where $\boldsymbol{h}_k$ is any vector in $\mathbb{K}^k$. Notice that

$$\boldsymbol{Q}_k^\intercal \boldsymbol{E} = \boldsymbol{Q}^\intercal (\boldsymbol{q}_{k+1} \boldsymbol{h}_k) = (\boldsymbol{Q}^\intercal \boldsymbol{q}_{k+1}) \boldsymbol{h}_k^\intercal = \boldsymbol{0}.$$

Since this holds for any $\boldsymbol{h}_k \in \mathbb{K}^k$, to preserve sparsity and to keep this form as simple as possible we can set $\boldsymbol{h}_k = [0, 0, \ldots, h_{k+1,k}]^\intercal$. This means $\boldsymbol{AQ}_k$ can be written as

$$\boldsymbol{AQ}_k = \boldsymbol{Q}_k \boldsymbol{H}_k + \boldsymbol{q}_{k+1} \boldsymbol{h}_k^\intercal \tag{37}$$

where

$$\boldsymbol{Q}_k\boldsymbol{H}_k = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k] \begin{bmatrix} h_{1,1} & \cdots & \cdots & \cdots & h_{1,k} \\ h_{2,1} & \cdots & \cdots & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{k,k-1} & h_{k,k} \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix}.$$

Equating the j^{th} columns of equation 37 yields

$$\boldsymbol{A}\boldsymbol{q}_j = \sum_{i=1}^{j+1} h_{i,j}\boldsymbol{q}_i.$$

Again since $\{\boldsymbol{q}_i\}_{i=1}^n$ form an orthornormal basis, multiplying both sides by $\boldsymbol{q}_m$ for $1 \leq m \leq j$ gives

$$\boldsymbol{q}_m^\intercal \boldsymbol{A}\boldsymbol{q}_j = \sum_{i=1}^{j+1} h_{i,j}\boldsymbol{q}_m^\intercal \boldsymbol{q}_i = h_{m,j}$$

and so

$$h_{j+1,j}\boldsymbol{q}_{j+1} = \boldsymbol{A}\boldsymbol{q}_j - \sum_{i=1}^{j} h_{i,j}\boldsymbol{q}_i. \tag{38}$$

From equation 38 we find that $\boldsymbol{q}_{j+1}$ can be computed using a recurrance involving its previous Krylov factors. Notice this bears a striking resemblance to equation 31 having a virtually an identical setup to computing an orthornormal basis using the modified Gram-Schmidt process (algorithm 4). As such, values for $\boldsymbol{q}_{j+1}$ and $h_{j+1,j}$ can be evaluted using a procedure very similar to the modified Gram-Schmidt process better known as the Arnoldi algorithm [Tre97, Dem97], presented in algorithm 5.

Algorithm 5: Arnoldi Algorithm

input : $\boldsymbol{A}, \boldsymbol{r}_0$ and k, the number of columns of $\boldsymbol{Q}$ to compute.
output: $\boldsymbol{Q}_k, \boldsymbol{H}_k$.

$\boldsymbol{q}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|$
for $j = 1$ **to** k **do**
  $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{q}_j$
  for $i = 1$ **to** j **do**
    $h_{i,j} = \langle \boldsymbol{q}_i, \boldsymbol{z} \rangle$
    $\boldsymbol{z} = \boldsymbol{z} - h_{i,j}\boldsymbol{q}_i$
  end
  $h_{j+1,j} = \|\boldsymbol{z}\|$
  if $h_{j+1,j} = 0$ **then**
    return $\boldsymbol{Q}_k, \boldsymbol{H}_k$
  end
  $\boldsymbol{q}_{j+1} = \boldsymbol{z}/h_{j+1,j}$
end
return $\boldsymbol{Q}_k, \boldsymbol{H}_k$

When $\boldsymbol{A}$ is symmertic then $\boldsymbol{H} = \boldsymbol{T}$ becomes a tridiagonal matrix, simplifying a large amount of the Arnoldi algorithm since the matrix elements from $\boldsymbol{T}$ can be written as

$$\boldsymbol{T} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

As before, equating the j^{th} columns of $\boldsymbol{AQ} = \boldsymbol{QT}$ yields

$$\boldsymbol{A}\boldsymbol{q}_j = \beta_{j-1}\boldsymbol{q}_{j-1} + \alpha_j\boldsymbol{q}_j + \beta_j\boldsymbol{q}_{j+1}. \tag{39}$$

Again since $\{\boldsymbol{q}_i\}_{i=1}^n$ form an orthornormal basis, multiplying both sides of equation 39 by $\boldsymbol{q}_j$ gives $\boldsymbol{q}_j\boldsymbol{A}\boldsymbol{q}_j = \alpha_j$. A simplified version of the Arnoldi algorithm can be devised can be used to compute $\{\boldsymbol{q}_i\}_{i=1}^n$ and $\boldsymbol{T}$ for symmetric matrices known as the Lanczos algorithm [Dem97]. The Lanczos algorithm is presented in algorithm 6.

Algorithm 6: Lanczos Algorithm

input : $\boldsymbol{A}, \boldsymbol{r}_0$ and k, the number of columns of $\boldsymbol{Q}$ to compute.
output: $\boldsymbol{Q}_k, \boldsymbol{T}_k$.

$\boldsymbol{q}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|$, $\beta_0 = 0$, $\boldsymbol{q}_0 = 0$
for $j = 1$ **to** k **do**
  $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{q}_j$
  $\alpha_j = \langle \boldsymbol{q}_j, \boldsymbol{z} \rangle$
  $\boldsymbol{z} = \boldsymbol{z} - \alpha_j \boldsymbol{q}_j - \beta_{j-1}\boldsymbol{q}_{j-1}$
  $\beta_j = \|z\|$
  if $\beta_j = 0$ **then**
    return $\boldsymbol{Q}_k, \boldsymbol{T}_k$
  end
  $\boldsymbol{q}_{j+1} = \boldsymbol{z}/\beta_j$
end
return $\boldsymbol{Q}_k, \boldsymbol{T}_k$

For the Lanczos algorithm, equation 37 can be re-written in the a more compact form as

$$\boldsymbol{A}\boldsymbol{V}_k = \boldsymbol{V}_k \boldsymbol{T}_{k+1,k} \tag{40}$$

where $\boldsymbol{T}_{k+1,k} = \boldsymbol{T}_k + \boldsymbol{v}_{k+1}\boldsymbol{t}_k^\intercal$.

3.4. **Optimality Conditions.** So far we have shown that $\boldsymbol{x}^\star \in \mathcal{K}_{t_{\boldsymbol{r}_0},\boldsymbol{A}}(\boldsymbol{A}, \boldsymbol{r}_0)$ where $n = t_{\boldsymbol{r}_0}$ is the grade of $\boldsymbol{r}_0$ with respect to $\boldsymbol{A}$. Moreover from section 3.3 we found ways to construct a basis for $\mathcal{K}_{t_{\boldsymbol{r}_0},\boldsymbol{A}}(\boldsymbol{A}, \boldsymbol{r}_0)$ allowing us to generate vectors with these affine spaces, namely the Arnoldi algorithm (algorithm 5)and Lanczos algorithm (algorithm 6) for non-symmertic and symmertic systems respectively. From now on $\mathcal{K}_{t_{\boldsymbol{r}_0},\boldsymbol{A}}(\boldsymbol{A}, \boldsymbol{r}_0)$ will be abbreviated to $\mathcal{K}_{t_{\boldsymbol{r}_0},\boldsymbol{A}}$ when the context is clear. The question still remains however, how should one choose an $\boldsymbol{x}_k$ that best approximates $\boldsymbol{x}^*$ satisfying equation 30? Here are a few of the most well known methods for selecting a suitable $\boldsymbol{x}_k$.

(1) Select an $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k$ which minimizes $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2$. While this method seems like the most intuitive and natural way to select $\boldsymbol{x}_k$, it is unfortunately of no practical use since there is not enough information in the Krylov subspace to find an $\boldsymbol{x}_k$ which matches this specification.
(2) Select an $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k$ which minimizes $\|\boldsymbol{r}_k\|_2$ (recall this is the residual of $\boldsymbol{x}_k$, that is, $\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$). This method is possible to implement. Two well known algorithms stem from this class of methods, namely MINRES (minimum residual) and GMRES (general minimum residual) which solve linear systems for symmetric and non-symmertic $\boldsymbol{A}$ respectively.
(3) When $\boldsymbol{A}$ is a positive definite matrix it defines a norm $\|r\|_{\boldsymbol{A}} = (\boldsymbol{r}^\intercal \boldsymbol{A}\boldsymbol{r})^{\frac{1}{2}}$, called the energy norm. Select an $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k$ which minimizes $\|r\|_{\boldsymbol{A}^{-1}}$ which is equivalent to minimizing $\|\boldsymbol{x}_k - \boldsymbol{x}\|_A$. This technique is known as the CG (conjugate gradient) algorithm.
(4) Select an $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k$ for which $\boldsymbol{r}_k \perp \mathcal{W}_k$ where $\mathcal{W}_k$ is some k-dimensional subspace. Two well known algorithms that belong to this family of methods are SYMMLQ (Symmetric LQ Method) and a variant of GMRES used for solving symmetric and non-symmetric methods respectively.

Interestingly, when $\boldsymbol{A}$ is symmetric positive definite and $\mathcal{W}_k = \mathcal{K}_k$ the last two selection methods are equivalent. This is stated more precisely in theorem 30 without proof.

Theorem 30. *In the context of the above selection method, if $\boldsymbol{A} \succ \boldsymbol{0}$ and $\mathcal{W}_k = \mathcal{K}_k$ in method (4) then it produces the same $\boldsymbol{x}_k$ in method (3)* [Dem97].

In fact the very last method can be used to bring together a number of different analytical aspects and unify them in a general framework known as projection methods. Selecting an $\boldsymbol{x}_k$ from our Krylov subspace allows k degrees of freedom meaning k constraints must be used to determine a unique $\boldsymbol{x}_k$ for selection. As seen in method (4) already, typically orthogonality constraints are imposed on the residual $\boldsymbol{r}_k$. Specifically we would like to find a $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k$ where $\boldsymbol{r}_k \perp \mathcal{W}_k$. This is sometimes referred to as the Petrov-Galerkin (or just Galerkin) conditions. Projection methods for which $\mathcal{W}_k = \mathcal{K}_k$ are known as orthogonal projections while methods for which $\mathcal{W}_k = \boldsymbol{A}\mathcal{K}_k$ are known as oblique projections. If we set $\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{z}_k$ for some $\boldsymbol{z}_k \in \mathcal{K}_k$ then the Petrov-Galerkin conditions imply $\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{z}_k \perp \mathcal{W}_k$, or alternatively $\langle \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{z}_k, \boldsymbol{w} \rangle = 0$ for every $\boldsymbol{w} \in \mathcal{W}_k$. To impose these conditions it will help to have an appropriate basis for $\mathcal{K}$ and $\mathcal{W}$. Suppose we have access to such a basis where $\{\boldsymbol{q}_i\}_{i=1}^k$ and $\{\boldsymbol{w}_i\}_{i=1}^k$ are basis elements for $\mathcal{K}$ and $\mathcal{W}$ respectively. Let

$$\boldsymbol{K}_k \triangleq [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k] \in \mathbb{K}^{n \times k}$$
$$\boldsymbol{W}_k \triangleq [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k] \in \mathbb{K}^{n \times k}$$

then the Petrov-Galerkin conditions can be imposed as follows

$$\boldsymbol{K}_k \boldsymbol{y}_k = \boldsymbol{z}_k, \quad \text{for some } \boldsymbol{y}_k \in \mathbb{K}^k$$
$$\boldsymbol{W}_k^\intercal \left(\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{K}_k \boldsymbol{y}_k \right) = \boldsymbol{0}.$$

Moreover if $\boldsymbol{W}_k^\intercal \boldsymbol{A}\boldsymbol{K}_k$ is invertible then $\boldsymbol{x}_k$ can be expressed as

$$\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{K}_k \left(\boldsymbol{W}_k^\intercal \boldsymbol{A}\boldsymbol{K}_k \right)^{-1} \boldsymbol{W}_k \boldsymbol{r}_0. \tag{41}$$

This justifies a general form of the projection method algorithm presented in algorithm 7.

Algorithm 7: General Projection Method

output: An approximation of $\boldsymbol{x}^*$, $\boldsymbol{x}_k$.

for $k = 1, \ldots$ **until** *convergence* **do**
- Select $\mathcal{K}_k$ and $\mathcal{W}_k$
- Form $\boldsymbol{K}_k$ and $\boldsymbol{W}_k$
- Solve $\left(\boldsymbol{W}_k^\intercal \boldsymbol{A}\boldsymbol{K}_k \right) \boldsymbol{y}_k = \boldsymbol{W}_k^\intercal \boldsymbol{r}_0$
- $\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{K}_k \boldsymbol{y}_k$

end

return $\boldsymbol{x}_k$

3.5. **Conjugate Gradient Algorithm.** From section 3.4 that the Petrov-Galerkin conditions for the CG algorithm used an orthogonal projection and the matrix $\boldsymbol{A}$ was assumed to be positive definite. To derive the CG algorithm we can start be using some machinery that the Lanczos algorithm provides us with. Recall, the Lanczos algorithm produces the form $\boldsymbol{A}\boldsymbol{Q}_k = \boldsymbol{Q}_k \boldsymbol{T}_k + \boldsymbol{q}_{k+1} \boldsymbol{t}_k^\intercal$ where $\boldsymbol{t}_k \triangleq [0, 0, \ldots, 0, \beta_k]^\intercal \in \mathbb{K}^k$

and the columns of $\boldsymbol{Q}_k$ span $\mathcal{K}_k$. Recall that $\boldsymbol{x}_k$ can be expressed as $\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{K}_k \left(\boldsymbol{W}_k^\intercal \boldsymbol{A}\boldsymbol{K}_k\right)^{-1} \boldsymbol{W}_k \boldsymbol{r}_0$ (equation 41) when $\boldsymbol{W}_k^\intercal \boldsymbol{A}\boldsymbol{K}_k$ is invertible. For the CG algorithm $\mathcal{K} = \mathcal{W}$ and $\boldsymbol{A} \succ \boldsymbol{0}$. Under these conditions we can easily show that $\boldsymbol{W}_k^\intercal \boldsymbol{A}\boldsymbol{K}_k$ is indeed invertible. This means the approximate vector can be expressed as $\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{z}_k$ where $\boldsymbol{z}_k \in \mathcal{K}_k$. In terms of the Petrov-Galerkin conditions this means that $\boldsymbol{z}_k$ must satisfy $\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{z}_k \perp \mathcal{W}_k$. Furthermore since $\mathcal{K}_k = \operatorname{Range}(\boldsymbol{Q}_k)$ where $\boldsymbol{Q}_k$ has full column rank then $\boldsymbol{z}_k$ can be represented as $\boldsymbol{z}_k = \boldsymbol{Q}_k \boldsymbol{y}$ for a unique $\boldsymbol{y} \in \mathbb{K}^k$ so that

$$\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{Q}_k \boldsymbol{y}. \tag{42}$$

Coupling this with the Petrov-Galerkin conditions means

$$\begin{aligned} \boldsymbol{Q}_k^\intercal (\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{Q}_k \boldsymbol{y}) &= \boldsymbol{0} \\ \boldsymbol{Q}_k^\intercal \boldsymbol{A}\boldsymbol{Q}_k \boldsymbol{y} &= \boldsymbol{Q}_k^\intercal \boldsymbol{r}_0 \\ \boldsymbol{T}_k \boldsymbol{y} &= \|\boldsymbol{r}_0\| \boldsymbol{e}_1. \end{aligned} \tag{43}$$

In the CG algorithm $\boldsymbol{x}_{k+1}$ is computed as the recurrance of the following three sets of vectors

(1) The approximate solutions $\boldsymbol{x}_k$
(2) The residual vectors $\boldsymbol{r}_k$
(3) The conjugate gradient vectors $\boldsymbol{p}_k$

The conjugate gradient vectors are given the name gradient since the attempt to find the direction of steepest descent that minimizes $\|\boldsymbol{r}_k\|_{\boldsymbol{A}^{-1}}$. The are also given the name conjugate since $\langle \boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_j \rangle = 0$ for $i \neq j$, that is, vectors $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are mutally A-conjugate.

Since $\boldsymbol{A}$ is symmetric positive definite then so is $\boldsymbol{T}_k = \boldsymbol{Q}_k \boldsymbol{A}\boldsymbol{Q}_k$. We can take the Cholesky decomposition of $\boldsymbol{T}_k$ to get

$$\boldsymbol{T}_k = \boldsymbol{L}_k \boldsymbol{D}_k \boldsymbol{L}_k^\intercal \tag{44}$$

where $\boldsymbol{L}_k$ is a unit lower bidiagonal matrix and $\boldsymbol{D}_k$ is diagonal written as

$$\boldsymbol{L}_k = \begin{bmatrix} 1 & & & \\ l_1 & \ddots & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix}, \quad \boldsymbol{D}_k = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_k \end{bmatrix}.$$

Combining equations 42, 43 and 44

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}_0 + \boldsymbol{Q}_k \boldsymbol{y} \\ \boldsymbol{x}_k &= \boldsymbol{x}_0 + \|\boldsymbol{r}_0\| \boldsymbol{Q}_k \boldsymbol{T}_k^{-1} \boldsymbol{e}_1 \\ \boldsymbol{x}_k &= \boldsymbol{x}_0 + \|\boldsymbol{r}_0\| \boldsymbol{Q}_k \left(\boldsymbol{L}_k \boldsymbol{D}_k \boldsymbol{L}_k^\intercal\right)^{-1} \boldsymbol{e}_1 \\ \boldsymbol{x}_k &= \boldsymbol{x}_0 + \left(\boldsymbol{Q}_k \boldsymbol{L}_k^{-\intercal}\right) \left(\|\boldsymbol{r}_0\| \boldsymbol{D}_k^{-1} \boldsymbol{L}_k^{-1} \boldsymbol{e}_1\right) \\ \boldsymbol{x}_k &\triangleq \boldsymbol{x}_0 + \tilde{\boldsymbol{P}}_k \tilde{\boldsymbol{y}}_k \end{aligned}$$

where $\tilde{\boldsymbol{P}}_k = \boldsymbol{Q}_k \boldsymbol{L}_k^{-\intercal}$ and $\tilde{\boldsymbol{y}}_k = \|\boldsymbol{r}_0\| \boldsymbol{D}_k^{-1} \boldsymbol{L}_k^{-1} \boldsymbol{e}_1$. The matrix $\tilde{\boldsymbol{P}}_k$ can be written as $\tilde{\boldsymbol{P}}_k = [\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, \ldots, \tilde{\boldsymbol{p}}_k]$. Lemma 31 shows that the columns of $\tilde{\boldsymbol{P}}_k$ are A-conjugate.

Lemma 31. *The columns of $\tilde{\boldsymbol{P}}_k$ are A-conjugate, in otherwise $\tilde{\boldsymbol{P}}_k^\intercal \boldsymbol{A} \tilde{\boldsymbol{P}}_k$ is diagonal.*

Proof. We compute

$$\begin{aligned}
\tilde{\boldsymbol{P}}_k^\intercal \boldsymbol{A} \tilde{\boldsymbol{P}}_k &= \left(\boldsymbol{Q}_k \boldsymbol{L}_k^{-\intercal}\right)^\intercal \boldsymbol{A} \left(\boldsymbol{Q}_k \boldsymbol{L}_k^{-\intercal}\right) \\
&= \boldsymbol{L}_k^{-1} \left(\boldsymbol{Q}_k^\intercal \boldsymbol{A} \boldsymbol{Q}_k\right) \boldsymbol{L}_k^{-\intercal} \\
&= \boldsymbol{L}_k^{-1} \left(\boldsymbol{T}_k\right) \boldsymbol{L}_k^{-\intercal} \\
&= \boldsymbol{L}_k^{-1} \left(\boldsymbol{L}_k \boldsymbol{D}_k \boldsymbol{L}_k^\intercal\right) \boldsymbol{L}_k^{-\intercal} \qquad \text{(equation 44)} \\
&= \boldsymbol{D}_k
\end{aligned}$$

as wanted. □

Since $\boldsymbol{L}_k$ is a lower bidiagonal, setting $\boldsymbol{a} \triangleq l_{k-1}\boldsymbol{e}_{k-1}$, it can be written in the form

$$\boldsymbol{L}_k = \begin{bmatrix} \boldsymbol{L}_{k-1} & \boldsymbol{0} \\ \boldsymbol{a}^\intercal & 1 \end{bmatrix}$$

meaning

$$\boldsymbol{L}_k^{-1} = \begin{bmatrix} \boldsymbol{L}_{k-1}^{-1} & \boldsymbol{0} \\ \star & 1 \end{bmatrix}.$$

With this a recurrance for the columns of $\tilde{\boldsymbol{P}}_k$ can now be derived in terms of $\boldsymbol{y}_k$. To start we can show that the first $k-1$ entries of $\tilde{\boldsymbol{y}}_k$ shares the first $k-1$ entires with $\tilde{\boldsymbol{y}}_{k-1}$ and that $\tilde{\boldsymbol{P}}_k$ and $\tilde{\boldsymbol{P}}_{k-1}$ share the same first $k-1$ columns. To start we can compute a recurrance for $\tilde{\boldsymbol{y}}_k$ as follows

$$\begin{aligned}
\tilde{\boldsymbol{y}}_k &= \|\boldsymbol{r}_0\| \boldsymbol{D}_k^{-1} \boldsymbol{L}_k^{-1} \boldsymbol{e}_1^k \\
&= \|\boldsymbol{r}_0\| \begin{bmatrix} \boldsymbol{D}_{k-1}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & d_k^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_{k-1}^{-1} & \boldsymbol{0} \\ \star & 1 \end{bmatrix} \boldsymbol{e}_1^k \\
&= \|\boldsymbol{r}_0\| \begin{bmatrix} \boldsymbol{D}_{k-1}^{-1} \boldsymbol{L}_{k-1}^{-1} & \boldsymbol{0} \\ \star & d_k^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_1^k \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \tilde{\boldsymbol{y}}_{k-1} \\ \eta_k \end{bmatrix}
\end{aligned}$$

To get a recurrance for the columns of $\tilde{\boldsymbol{P}}_{k-1} = [\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, \ldots, \tilde{\boldsymbol{p}}_k]$ since $\boldsymbol{L}_{k-1}^\intercal$ is upper triangular then so is $\boldsymbol{L}_{k-1}^{-\intercal}$, thus forming the leading $(k-1)-$by$-(k-1)$ submatrix of $\boldsymbol{L}_k^{-\intercal}$. This means that $\tilde{\boldsymbol{P}}_{k-1}$ is identical to the leading $k-1$ columns of

$$\tilde{\boldsymbol{P}}_k = \boldsymbol{Q}_k \boldsymbol{L}_k^{-\intercal} = [\boldsymbol{Q}_{k-1}, \boldsymbol{q}_k] \begin{bmatrix} \boldsymbol{L}_{k-1}^{-1} & \boldsymbol{0} \\ \star & 1 \end{bmatrix} = \left[\boldsymbol{Q}_{k-1} \boldsymbol{L}_{k-1}^{-1}, \tilde{\boldsymbol{p}}_k\right] = \left[\tilde{\boldsymbol{P}}_{k-1}, \tilde{\boldsymbol{p}}_k\right].$$

Moreover rearranging $\tilde{\boldsymbol{P}}_k = \boldsymbol{Q}_k \boldsymbol{L}_k^{-\intercal}$ we get $\tilde{\boldsymbol{P}}_k \boldsymbol{L}_k^\intercal = \boldsymbol{Q}_k$. Equating the k^{th} column yields

$$\tilde{\boldsymbol{p}}_k = \boldsymbol{q}_k - l_{k-1} \tilde{\boldsymbol{p}}_{k-1}. \tag{45}$$

Finally we can use

$$\boldsymbol{x}_k = \boldsymbol{x}_0 + \tilde{\boldsymbol{P}}_k \tilde{\boldsymbol{y}}_k = \boldsymbol{x}_0 + \begin{bmatrix} \tilde{\boldsymbol{P}}_{k-1}, \tilde{\boldsymbol{p}}_k \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{y}}_{k-1} \\ \eta_k \end{bmatrix} = \boldsymbol{x}_0 + \tilde{\boldsymbol{P}}_{k-1}\tilde{\boldsymbol{y}}_{k-1} + \eta_k \tilde{\boldsymbol{p}}_k = \boldsymbol{x}_{k-1} + \eta_k \tilde{\boldsymbol{p}}_k \tag{46}$$

as a recurrance for $\boldsymbol{x}_k$. A recurrance for $\boldsymbol{r}_k$ is easily computed as

$$\boldsymbol{r}_k = b - \boldsymbol{A}\boldsymbol{x}_k = b - \boldsymbol{A}\left(\boldsymbol{x}_{k-1} + \eta_k \tilde{\boldsymbol{p}}_k\right) = \left(b - \boldsymbol{A}\boldsymbol{x}_{k-1}\right) - \eta_k \boldsymbol{A}\tilde{\boldsymbol{p}}_k = \boldsymbol{r}_{k-1} - \eta_k \boldsymbol{A}\tilde{\boldsymbol{p}}_k \tag{47}$$

Altogether we are left with recurrences for $\boldsymbol{q}_k$ from Lanczos, $\tilde{\boldsymbol{p}}_k$ (equation 45), the residual $\boldsymbol{r}_k$ (equation 45), and for the approximate solution $\boldsymbol{x}_k$ (equation 46). However, futher simplification can be made for a more efficient algorithm. Recall from section 3.3 that $\boldsymbol{A}\boldsymbol{Q}_k = \boldsymbol{Q}_k\boldsymbol{T}_k + \boldsymbol{q}_{k+1}\boldsymbol{t}_k^\intercal$ where $\boldsymbol{t}_k = [0, 0, \ldots, 0, \beta_k]^\intercal \in \mathbb{K}^k$ meaning

$$\boldsymbol{r}_k = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y}_k = \boldsymbol{r}_0 - \boldsymbol{Q}_k\boldsymbol{T}_k\boldsymbol{y}_k - \langle \boldsymbol{t}_k, \boldsymbol{y}\rangle \boldsymbol{q}_{k+1} = -\beta_k y_k \boldsymbol{q}_{k+1}.$$

This tells us that $\boldsymbol{r}_k$ is parallel to $\boldsymbol{q}_{k+1}$ and orthogonal to all $\boldsymbol{q}_i,\ 1 \le i \le k$. This further implies that $\boldsymbol{r}_k$ is orthogonal to all $\boldsymbol{r}_i,\ 1 \le i \le k-1$ since they are just $\boldsymbol{q}_i$ scaled by some constant factor. So replacing $\boldsymbol{r}_{k-1}$ with $\boldsymbol{q}_k/\eta_k$ and defining $\boldsymbol{p}_k \triangleq \tilde{\boldsymbol{p}}_k/\gamma_k$ gives us a new set of recurrences

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + \alpha_k \boldsymbol{p}_k \\ \boldsymbol{r}_k &= \boldsymbol{r}_{k-1} - \alpha_k \boldsymbol{A}\boldsymbol{p}_k \\ \boldsymbol{p}_k &= \boldsymbol{r}_{k-1} + \beta_k \boldsymbol{p}_{k-1} \end{aligned}$$

where $\alpha_k = \eta_k/\gamma_k$. From theorem 31 we have shown that the columns of $\tilde{\boldsymbol{P}}_k$ are A-conjugate (that is $\langle \tilde{\boldsymbol{p}}_i, \boldsymbol{A}\tilde{\boldsymbol{p}}_j\rangle = 0,\ i \ne j$) and that $\tilde{\boldsymbol{P}}_k^\intercal \boldsymbol{A}\tilde{\boldsymbol{P}}_k = \boldsymbol{D}_k$. This also means that $\langle \boldsymbol{r}_i, \boldsymbol{r}_j\rangle = 0,\ i \ne j$. Now note that from our recurrence for $\boldsymbol{p}_k = \boldsymbol{r}_{k-1} + \beta_k \boldsymbol{p}_{k-1}$ that

$$\langle \boldsymbol{A}\boldsymbol{p}_k, \boldsymbol{p}_k\rangle = \langle \boldsymbol{A}\boldsymbol{p}_k, \boldsymbol{r}_{k-1} + \beta_k \boldsymbol{p}_{k-1}\rangle = \langle \boldsymbol{A}\boldsymbol{p}_k, \boldsymbol{r}_{k-1}\rangle.$$

We can now find an expression for α_k as

$$\begin{aligned} \langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_k\rangle &= \langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1} - \alpha_k \boldsymbol{A}\boldsymbol{p}_k\rangle \\ \langle \boldsymbol{r}_{k-1} - 1\rangle &= \langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1}\rangle - \alpha_k \langle \boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k\rangle \\ \alpha_k &= \frac{\langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1}\rangle}{\langle \boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k\rangle}. \end{aligned}$$

Similarly, using the recurrence for $\boldsymbol{p}_k$, an expression for β_k can be computed as

$$\begin{aligned} \langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{p}_k\rangle &= \langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{r}_{k-1} + \beta_k \boldsymbol{p}_{k-1}\rangle \\ \langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{p}_k\rangle &= \langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{r}_{k-1}\rangle + \beta_k \langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{p}_{k-1}\rangle \\ \beta_k &= -\frac{\langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{r}_{k-1}\rangle}{\langle \boldsymbol{A}\boldsymbol{p}_{k-1}, \boldsymbol{p}_{k-1}\rangle}. \end{aligned}$$

This formula requires an additional dot product which was not present before. Fortunately, this dot product can be eliminated using our recurrence for $\boldsymbol{r}_k$

$$\begin{aligned} \langle \boldsymbol{r}_k, \boldsymbol{r}_k\rangle &= \langle \boldsymbol{r}_k, \boldsymbol{r}_{k-1} - \alpha_k \boldsymbol{A}\boldsymbol{p}_k\rangle \\ \langle \boldsymbol{r}_k, \boldsymbol{r}_k\rangle &= \langle \boldsymbol{r}_k, \boldsymbol{r}_{k-1}\rangle - \alpha_k \langle \boldsymbol{r}_k, \boldsymbol{A}\boldsymbol{p}_k\rangle \\ \alpha_k &= -\frac{\langle \boldsymbol{r}_k, \boldsymbol{r}_k\rangle}{\langle \boldsymbol{r}_k, \boldsymbol{A}\boldsymbol{p}_k\rangle}. \end{aligned}$$

Equating the two expressions for $\boldsymbol{a}_k$ yields

$$-\frac{\langle \boldsymbol{r}_k, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{r}_k, \boldsymbol{A}\boldsymbol{p}_k \rangle} = \frac{\langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1} \rangle}{\langle \boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k \rangle}$$
$$-\frac{\langle \boldsymbol{r}_k, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1} \rangle} = \frac{\langle \boldsymbol{r}_k, \boldsymbol{A}\boldsymbol{p}_k \rangle}{\langle \boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k \rangle}.$$

This means that

$$\beta_k = \frac{\langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1} \rangle}{\langle \boldsymbol{r}_{k-2}, \boldsymbol{r}_{k-2} \rangle}.$$

These recurrences are computed iteratively to form the basis of the CG algorithm, seen in Algorithm 8.

Algorithm 8: CG Algorithm

input : $\boldsymbol{A} \succ \boldsymbol{0}$, $\boldsymbol{b}$ and an initial guess $\boldsymbol{x}_0$.
output: An approximation of $\boldsymbol{x}^*$, $\boldsymbol{x}_k$.

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\boldsymbol{p}_1 = \boldsymbol{r}_0$
for $k = 1, \ldots$ **until** $\|r_{k-1}\| \leq \tau$ **do**
  $\alpha_k = \frac{\langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1} \rangle}{\langle \boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k \rangle}$
  $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k \boldsymbol{p}_k$
  $\boldsymbol{r}_k = \boldsymbol{r}_{k-1} - \alpha_k \boldsymbol{A}\boldsymbol{p}_k$
  $\beta_{k+1} = \frac{\langle \boldsymbol{r}_k, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1} \rangle}$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_k + \beta_{k+1} \boldsymbol{p}_k$
end
return $\boldsymbol{x}_k$

3.6. **Minimum Residual.** In contrast to CG, MINRES is able applicable to a wider range of linear systems and is used for solving symmertic indefinite systems. From section 3.4 we saw that MINRES minimizes the residual $\boldsymbol{r}_k$ with respect to the Euclidean norm at each iteration (hence the name), that is $\boldsymbol{x}_k$ is chosen so that

$$\boldsymbol{x}_k = \underset{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_k}{\arg\min} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 . \tag{48}$$

It can be shown that this is equivalent setting $\mathcal{W} = \boldsymbol{A}\mathcal{K}_k$ in the Petrov-Galerkin conditions where $\det(\boldsymbol{A}) \neq 0$, in other words MINRES is an oblique projection method. We can also show that when $\det(\boldsymbol{A}) \neq 0$ and $\mathcal{W} = \boldsymbol{A}\mathcal{K}_k$ the matrix $\boldsymbol{W}^\intercal \boldsymbol{A}\boldsymbol{K}$ is non-singular. So under the conditions of MINRES, using a similar argument used for CG, $\boldsymbol{x}_k$ can be expressed as $\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{V}_k \boldsymbol{y}_k$. Using equation 40 produced by the Lanczos algorithm in 3.3 we can manipulate our optimality condition 48 as follows

$$\begin{aligned}
\boldsymbol{x}_k &= \underset{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_k}{\arg\min} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \\
\iff \boldsymbol{y}_k &= \underset{\boldsymbol{y} \in \mathbb{R}^k}{\arg\min} \|\boldsymbol{A}(\boldsymbol{x}_0 + \boldsymbol{V}_k \boldsymbol{y}) - \boldsymbol{b}\|_2 \\
&= \underset{\boldsymbol{y} \in \mathbb{R}^k}{\arg\min} \|-(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0) + \boldsymbol{A}\boldsymbol{V}_k \boldsymbol{y}\|_2 \\
&= \underset{\boldsymbol{y} \in \mathbb{R}^k}{\arg\min} \|\boldsymbol{V}_k \boldsymbol{T}_{k+1,k} \boldsymbol{y} - \boldsymbol{r}_0\|_2
\end{aligned}$$

$$= \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \left\|\boldsymbol{T}_{k+1,k}\boldsymbol{y} - \boldsymbol{V}_k^\intercal \boldsymbol{r}_0\right\|_2$$
$$= \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \left\|\boldsymbol{T}_{k+1,k}\boldsymbol{y} - \beta_0\boldsymbol{e}_1\right\|_2$$

[Gre97, page 43]. Using the general project method procedure 7 as a guide, this gives the following high-level description of the MINRES algorithm [Tre97, page 268].

Algorithm 9: High-Level MINRES

output: An approximation of $\boldsymbol{x}^*$, $\boldsymbol{x}_k$.

for $k = 1, \ldots$ **until** *convergence* **do**

Lanczos-Step $(\boldsymbol{A}, \boldsymbol{v}_k, \boldsymbol{v}_{k-1}, \beta_k) \to \alpha_k, \beta_{k+1}, \boldsymbol{v}_{k+1}$

Find $\boldsymbol{y}$ to minimize $\left\|\boldsymbol{T}_{k+1,k}\boldsymbol{y} - \beta_0\boldsymbol{e}_1\right\|_2$

$\boldsymbol{x}_k = \boldsymbol{V}_k\boldsymbol{y}$

end

return $\boldsymbol{x}_k$

At each step solving $\boldsymbol{y}$ is a matter of solving a $(n+1)\times n$ least squares problem with Hessenberg structure. This can be done by performing a QR-factorisation on the successive $\boldsymbol{T}_{k+1,k}$ matrices, whats more using a clever bit of thinking, we can actually compute the QR-factorisation of $\boldsymbol{T}_{k+1,k}$ from the QR-factorisation of $\boldsymbol{T}_{k,k-1}$ using a inexpensive $\mathcal{O}(n)$ Householder reflection [Tre97, page 268]. Computing the QR-factorisation of $\boldsymbol{T}_{k,k-1}$ yields

$$\boldsymbol{Q}_k\boldsymbol{T}_{k,k+1} = \begin{bmatrix}\boldsymbol{R}_k\\ 0\end{bmatrix} = \begin{bmatrix} \gamma_1^{(1)} & \delta_2^{(1)} & \varepsilon_3^{(1)} & & & \\ & \gamma_1^{(2)} & \delta_3^{(2)} & \varepsilon_4^{(1)} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \varepsilon_k^{(1)} \\ & & & & \ddots & \gamma_k^{(2)} \\ & & & & & \delta_k^{(2)} \\ & & & & & 0 \end{bmatrix} \tag{49}$$

where $\boldsymbol{Q}_k = \prod_{i=1}^{k}\boldsymbol{Q}_{i,i+1}$ is the product of Householder reflections designed to annihilate the β_is in the subdiagonal of $\boldsymbol{T}_{k,k+1}$ [Cho07, page 25] where each $\boldsymbol{Q}_{i,i+1}$ is defined as

$$\boldsymbol{Q}_{i,i+1} \triangleq \begin{bmatrix} \mathbb{1}_{(i-1)\times(i-1)} & & & \\ & c_i & s_{i-1} & \\ & & & \mathbb{1}_{(k-i)\times(k-i)} \end{bmatrix}$$

[Cho07, page 22]. As mentioned for each i, $\boldsymbol{Q}_{i,i+1}$ is orthogonal, symmetric and constructed to annihilate β_{k+1} that is the bottom right element of $\boldsymbol{T}_{k,k+1}$. To see this better we can alternatively write $\boldsymbol{Q}_k\boldsymbol{T}_{k,k+1}$ as $\boldsymbol{Q}_{k,k+1}\cdot\boldsymbol{Q}_{2,3}\boldsymbol{Q}_{1,2}\boldsymbol{T}_{k,k+1}$. Notice however that $\boldsymbol{Q}_{k,k+1}$ is the only rotation matrix that will involve β_{k+1} in its matrix multiplication with $\boldsymbol{T}_{k,k+1}$. To study the influence of $\boldsymbol{Q}_{k,k+1}$ in a more compact way we only need to consider the matrix vector product

$$\begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix}\begin{bmatrix} \gamma_k^{(1)} & \delta_{k+1}^{(1)} & 0 \\ \beta_{k+1} & \alpha_{k+1} & \beta_{k+1}\end{bmatrix}. \tag{50}$$

To annihilate β_{k+1}, we find the appropriate choice of c_k and s_k are

$$\rho_k = \sqrt{{\gamma_k^{(1)}}^2 + \beta_{k+1}^2}, \quad c_k \triangleq \frac{\gamma_k^{(1)}}{\rho_k}, \quad s_k \triangleq \frac{\beta_{k+1}}{\rho_k}$$

so that 50 becomes

$$\begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} \begin{bmatrix} \gamma_k^{(1)} & \delta_{k+1}^{(1)} & 0 \\ \beta_{k+1} & \alpha_{k+1} & \beta_{k+1} \end{bmatrix} = \begin{bmatrix} \gamma_k^{(2)} & \delta_{k+1}^{(2)} & \varepsilon_{k+2}^{(1)} \\ 0 & \gamma_{k+1}^{(1)} & \delta_{k+2}^{(1)} \end{bmatrix}.$$

Furthermore, if we set

$$\begin{aligned} \boldsymbol{Q}_k (\beta_0 \boldsymbol{e}_1) &= \prod_{i=1}^{k} \boldsymbol{Q}_{i,i+1} (\beta_0 \boldsymbol{e}_1) \\ &= \beta_0 \boldsymbol{Q}_{k,k+1} \cdots \boldsymbol{Q}_{2,3} \begin{bmatrix} c_1 \\ s_1 \\ \boldsymbol{0}_{k-1} \end{bmatrix} \\ &= \beta_0 \boldsymbol{Q}_{k,k+1} \cdots \boldsymbol{Q}_{3,4} \begin{bmatrix} c_1 \\ s_1 c_1 \\ s_1 s_2 \\ \boldsymbol{0}_{k-2} \end{bmatrix} \\ &= \beta_0 \boldsymbol{Q}_{k,k+1} \begin{bmatrix} c_1 \\ s_1 c_1 \\ \vdots \\ s_1 \cdots s_{k-1} \\ 0 \end{bmatrix} \\ &= \boldsymbol{Q}_{k,k+1} \begin{bmatrix} \boldsymbol{t}_{k-1} \\ \phi_{k-1} \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{t}_k \\ \phi_{k-1} \end{bmatrix} \end{aligned}$$

where $\boldsymbol{t}_k = [\tau_1, \tau_2, \cdots, \tau_k]^\intercal$ then our optimality condition becomes

$$\boldsymbol{y}_k = \underset{\boldsymbol{y} \in \mathbb{R}^k}{\arg\min} \left\| \begin{bmatrix} \boldsymbol{R}_k \\ 0 \end{bmatrix} \boldsymbol{y} - \begin{bmatrix} \boldsymbol{t}_k \\ \phi_{k-1} \end{bmatrix} \right\|_2$$

which can be used to formulate subproblems [Cho07, page 25]. From the new optimality condition it it obvious that the optimal solution will satisfy $\boldsymbol{R}_k \boldsymbol{y}_k = \boldsymbol{t}_k$. However, instead of solving for $\boldsymbol{y}_k$ directly, MINRES solves

$$\boldsymbol{R}_k^\intercal \boldsymbol{D}_k^\intercal = \boldsymbol{V}_k^\intercal \tag{51}$$

where $\boldsymbol{D}_k = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k] \triangleq \boldsymbol{V}_k \boldsymbol{R}_k^{-1}$. This is done by forward substitution obtaining the last column $\boldsymbol{d}_k$ of $\boldsymbol{D}_k$ at iteration k. At the same time $\boldsymbol{x}_k$ is updated as

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{V}_k \boldsymbol{y}_k = \boldsymbol{D}_k \boldsymbol{R}_k \boldsymbol{y}_k = \boldsymbol{D}_k \boldsymbol{t}_k \\ &= [\boldsymbol{D}_{k-1}, \boldsymbol{d}_k] \begin{bmatrix} \boldsymbol{t}_{k-1} \\ \tau_t \end{bmatrix} \\ &= \boldsymbol{x}_{k-1} + \tau_k \boldsymbol{d}_k. \end{aligned} \tag{52}$$

The d_j in equation 52 can be found as

$$\begin{cases} \boldsymbol{d}_1 = \boldsymbol{v}_1/\gamma_1 \\ \boldsymbol{d}_2 = \left(\boldsymbol{v}_2 - \delta_2 \boldsymbol{d}_1\right)/\gamma_2^{(2)} \\ \boldsymbol{d}_j = \left(\boldsymbol{v}_j - \delta_j^{(2)} \boldsymbol{d}_{j-1} - \varepsilon_j \boldsymbol{d}_{j-2}\right)/\gamma_j^{(2)}, \quad j = 3, \dots k \end{cases}$$

from equation 51 [CHO11, page 4]. The final form of the MINRES in presented in algorithm 10.

Algorithm 10: MINRES Algorithm

input : $\boldsymbol{A}$ where $\boldsymbol{A} = \boldsymbol{A}^\intercal$, $\boldsymbol{b}$ and an initial guess $\boldsymbol{x}_0$.

output: An approximation of $\boldsymbol{x}^*$, $\boldsymbol{x}_k$.

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\tau_0 = \beta_0 = \|\boldsymbol{r}_0\|$, $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta_0$, $\delta_1^{(1)} = 0$, $\boldsymbol{v}_0 = \boldsymbol{d}_0 = \boldsymbol{d}_{-1} = \boldsymbol{0}$

for $k = 1, \dots$ **until** *convergence* **do**

Lanczos-Step $(\boldsymbol{A}, \boldsymbol{v}_k, \boldsymbol{v}_{k-1}, \beta_k) \to \alpha_k, \beta_{k+1}, \boldsymbol{v}_{k+1}$

$$\begin{bmatrix} \delta_k^{(2)} & \varepsilon_{k+1}^{(1)} \\ \gamma_k^{(1)} & \delta_{k+1}^{(1)} \end{bmatrix} = \begin{bmatrix} c_{k-1} & s_{k-1} \\ s_{k-1} & -c_{k-1} \end{bmatrix} \begin{bmatrix} \delta_k^{(1)} & 0 \\ \alpha_k & \beta_{k+1} \end{bmatrix}$$

$\rho_k = \sqrt{{\gamma_k^{(1)}}^2 + \beta_{k+1}^2}$, $\quad c_k = \frac{\gamma_k^{(1)}}{\rho_k}$, $\quad s_k = \frac{\beta_{k+1}}{\rho_k}$

$\gamma_k^{(2)} = c_k \gamma_k^{(1)}$

$\tau_k = c_k \phi_{k-1}$, $\quad \phi_k = s_k \phi_{k-1}$

$\boldsymbol{d}_k = \left(\boldsymbol{v}_k - \delta_k^{(2)} \boldsymbol{d}_{k-1} - \varepsilon_k \boldsymbol{d}_{k-2}\right)/\gamma_k^{(2)}$

$\boldsymbol{x}_k + \tau_k \boldsymbol{d}_k$

end

return $\boldsymbol{x}_k$

References

[Ras06] Carl Edward and Williams Rasmussen Christopher K. I, *Gaussian processes for machine learning / Carl Edward Rasmussen, Christopher K.I. Williams.*, Adaptive computation and machine learning, MIT Press, Cambridge, Mass., 2006 (eng).

[Mur12] Kevin P. Murphy, *Machine learning : a probabilistic perspective / Kevin P. Murphy.*, Adaptive computation and machine learning, MIT Press, Cambridge, MA, 2012 (eng).

[Ber96] Z.G. Sheftel Berezansky G.F, *Functional analysis. Volume 1 / Y.M. Berezansky, Z.G. Sheftel, G.F. Us ; translated from the Russian by Peter V. Malyshev.*, 1st ed. 1996., Operator Theory: Advances and Applications, 85, Basel ; Boston ; Berlin : BirkhaIuser Verlag, Basel ; Boston ; Berlin, 1996 (eng).

[Tre97] Lloyd N. (Lloyd Nicholas) and Bau Trefethen David, *Numerical linear algebra / Lloyd N. Trefethen, David Bau.*, SIAM Society for Industrial and Applied Mathematics, Philadelphia, 1997 (eng).

[Dem97] James W Demmel, *Applied numerical linear algebra / James W. Demmel.*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1997 (eng).

[Ste08] Ingo and Christmann Steinwart Andreas, *Support Vector Machines*, 1st ed. 2008., Information Science and Statistics, Springer New York, New York, NY, 2008 (eng).

[Ber03] Alain and Thomas-Agnan Berlinet Christine, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, Springer, SpringerLink (Online service), Boston, MA, 2003 (eng).

[Ste99] Michael L Stein, *Interpolation of Spatial Data Some Theory for Kriging / by Michael L. Stein.*, 1st ed. 1999., Springer Series in Statistics, Springer New York : Imprint: Springer, New York, NY, 1999 (eng).

[Bos92] Bernhard and Guyon Boser Isabelle and Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on computational learning theory, 1992, pp. 144–152 (eng).

[Cor95] Corinna Cortes, *Support-Vector Networks*, Machine learning **20** (1995), no. 3, 273 (eng).

[Kro14] Dirk P and C.C. Chan Kroese Joshua, *Statistical Modeling and Computation by Dirk P. Kroese, Joshua C.C. Chan.*, 1st ed. 2014., Springer New York : Imprint: Springer, New York, NY, 2014 (eng).

[Fle00] R Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, Incorporated, New York, 2000 (eng).

[Bis06] Christopher M Bishop, *Pattern recognition and machine learning / Christopher M. Bishop.*, Information science and statistics, Springer, New York, 2006 (eng).

[Pre92] William H. (William Henry) Press, *Numerical recipes in C : the art of scientific computing / William H. Press ...* [*et al.*], 2nd ed., Cambridge University Press, Cambridge, 1992 (eng).

[Wan] Guorong and Wei Wang Yimin and Qiao, *Generalized Inverses: Theory and Computations*, Developments in Mathematics, vol. 53, Springer Singapore, Singapore (eng).

[Gre97] Anne Greenbaum, *Iterative methods for solving linear systems Anne Greenbaum.*, Frontiers in applied mathematics ; 17, Society for Industrial and Applied Mathematics SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104, Philadelphia, Pa., 1997 (eng).

[Cho07] Sou-Cheng (Terrya) Choi, *Iterative methods for singular linear equations and least -squares problems*, ProQuest Dissertations Publishing, 2007 (eng).

[CHO11] Sou-Cheng T and PAIGE CHOI Christopher C and SAUNDERS, *MINRES-QLP: A KRYLOV SUBSPACE METHOD FOR INDEFINITE OR SINGULAR SYMMETRIC SYSTEMS*, SIAM journal on scientific computing **33** (2011), no. 3-4, 1810–1836 (eng).

[Rah08] Ali and Recht Rahimi Benjamin, *Random Features for Large-Scale Kernel Machines*, Advances in Neural Information Processing Systems, 2008.

[Pot21] Andres and Wu Potapczynski Luhuan and Biderman, *Bias-Free Scalable Gaussian Processes via Randomized Truncations* (2021) (eng).

[Hah33] Hans Hahn, *S. Bochner, Vorlesungen über Fouriersche Integrale: Mathematik und ihre Anwendungen, Bd. 12.) Akad. Verlagsges., Leipzig 1932, VIII. u. 229S. Preis brosch. RM 14,40, geb. RM16*, Monatshefte für Mathematik **40** (1933), no. 1, A27–A27 (ger).

[Liu21] Fanghui and Huang Liu Xiaolin and Chen, *Random Features for Kernel Approximation: A Survey on Algorithms, Theory, and Beyond*, IEEE transactions on pattern analysis and machine intelligence **PP** (2021) (eng).

[HAe16] Haim Avron etal, *Quasi-Monte Carlo Feature Maps for Shift-Invariant Kernels*, Journal of Machine Learning Research **17** (2016), no. 120, 1-38.

[DJSaJS15] Danica J. Sutherland and Jeff Schneider, *On the Error of Random Fourier Features*, 2015.

[Yu16] Felix X and Suresh Yu Ananda Theertha and Choromanski, *Orthogonal Random Features* (2016) (eng).

[Bro91] Peter J and Davis Brockwell Richard A, *Time Series: Theory and Methods*, Second Edition., Springer Series in Statistics, Springer New York, SpringerLink (Online service), New York, NY, 1991 (eng).

[Cho17] Krzysztof and Rowland Choromanski Mark and Weller, *The Unreasonable Effectiveness of Structured Random Orthogonal Embeddings* (2017) (eng).

[FaA76] Fino and Algazi, *Unified Matrix Treatment of the Fast Walsh-Hadamard Transform*, IEEE transactions on computers **C-25** (1976), no. 11, 1142–1146 (eng).

[And15] Alexandr and Indyk Andoni Piotr and Laarhoven, *Practical and Optimal LSH for Angular Distance* (2015) (eng).

[Cho20] Krzysztof and Likhosherstov Choromanski Valerii and Dohan, *Rethinking Attention with Performers* (2020) (eng).

[Boj16] Mariusz and Choromanska Bojarski Anna and Choromanski, *Structured adaptive and random spinners for fast machine learning computations* (2016) (eng).