CENTRO INTERNACIONAL DE ESTUDOS
DE DOUTORAMENTO E AVANZADOS
DA USC (CIEDUS)

PH.D. THESIS

A BAYESIAN APPROACH TO SIMULTANEOUSLY CHARACTERIZE THE STOCHASTIC AND DETERMINISTIC COMPONENTS OF A SYSTEM

Author:

Constantino Antonio García Martínez

Supervised by:

Paulo Félix Lamas
Abraham Otero Quintana
Jesús María Rodríguez Presedo

ESCOLA DE DOUTORAMENTO INTERNACIONAL. PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Centro Singular de Investigación en Tecnoloxías da Información
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

January 16, 2019

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Declaración del autor de la tesis
A BAYESIAN APPROACH TO SIMULTANEOUSLY CHARACTERIZE THE STOCHASTIC AND DETERMINISTIC COMPONENTS OF A SYSTEM

Don Constantino Antonio García Martínez

Presento mi tesis, siguiendo el procedimiento adecuado al Reglamento, y declaro que:

1. *La tesis abarca los resultados de la elaboración de mi trabajo.*
2. *En su caso, en la tesis se hace referencia a las colaboraciones que tuvo este trabajo.*
3. *La tesis es la versión definitiva presentada para su defensa y coincide con la versión enviada en formato electrónico.*
4. *Confirmo que la tesis no incurre en ningún tipo de plagio de otros autores ni de trabajos presentados por mí para la obtención de otros títulos.*

En Santiago de Compostela, 16 de enero de 2019

Fdo. Constantino Antonio García Martínez

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Autorización del Director/Tutor de la Tesis

A BAYESIAN APPROACH TO SIMULTANEOUSLY CHARACTERIZE THE STOCHASTIC AND DETERMINISTIC COMPONENTS OF A SYSTEM

Don Paulo Félix Lamas, Profesor Titular del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

Don Abraham Otero Quintana, Profesor Titular del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad San Pablo CEU

Don Jesús María Rodríguez Presedo, Profesor Titular del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

INFORMAN:

Que la presente tesis, corresponde con el trabajo realizado por ***Don Constantino Antonio García Martínez*** *bajo nuestra dirección, y autorizamos su presentación, considerando que reúne los requisitos exigidos en el Reglamento de Estudios de Doctorado de la USC, y que como directores de ésta no incurre en las causas de abstención establecidas en Ley 40/2015.*

En Santiago de Compostela, 16 de enero de 2019

Fdo. Paulo Félix Lamas
Director tesis

Fdo. Abraham Otero Quintana
Director tesis

Fdo. Jesús María Rodríguez Presedo
Director tesis

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Agradecimientos

En primer lugar, me gustaría expresar mi más sincero agradecimiento a mis directores de tesis Paulo Félix, Jesús Presedo y Abraham Otero por toda la implicación, esfuerzo, conocimiento y apoyo que me han brindado durante el desarrollo de esta tesis. A pesar de darme toda la libertad necesaria para desarrollar mis propias ideas, siempre habéis sido capaces de darme valiosísimos consejos y de orientarme en la dirección correcta. Espero que este documento sea capaz de reflejar una pequeña parte de lo que he aprendido con vosotros.

También me gustaría agradecer a todos los miembros de ForWind, de la Universidad de Oldenburg, el excelente trato que me ofrecieron durante mi estancia de investigación. Gracias especialmente a Pedro Lind, no solo por introducirme muchos de los conceptos que hicieron posible esta tesis, sino también por hacer que me sintiera como en casa.

Por supuesto, también es necesario dar las gracias a todas las instituciones que hicieron posible esta tesis. Principalmente, me gustaría agradecer el apoyo recibido de la Xunta de Galicia bajo el programa "Plan I2C" (cofinanciado parcialmente por el Fondo Social de la Unión Europea) y el apoyo del programa de Formación de Profesorado Universitario (FPU) del Ministerio de Educación español (Ref. FPU14 / 02489). Asimismo, es necesario reconocer la ayuda del Ministerio de Economía y Competitividad español (MINECO) en el marco del proyecto TIN2014-55183-R, y de la Universidad San Pablo CEU bajo la subvención PPC12/2014. Finalmente, quisiera agradecer al CiTIUS su apoyo económico para la realización de la estancia en la Universidad de Oldenburg.

Gracias también a todas las personas del CiTIUS, por hacer del centro un segundo hogar durante tanto tiempo. Merecen una mención aparte mis compañeros y amigos, sin los cuales mi salud mental se hubiese visto seriamente comprometida. Gracias a Adrián, Ismael, Martín, Pablo, Tomás, Tefa, Arturo, Jano, David Cohete, Borja, Víctor, Andrea, Yago, Álvaro, Pablo Viqueira, Jorge y Ángel. Gracias por vuestra pasión, vuestras ideas, las preguntas incisivas,

las terapias emocionales, las partidas de ajedrez, el servicio técnico, las noches de *Cosmos* ~~y las tardes en la ciudad de la cultura~~.

En un plano más personal, también me gustaría dedicar parte de esta tesis a mis amigos: Rubén, Óscar, Alberto, Lino, Luis, Ricardo y Jesús. No solo por las sesudas derivaciones matemáticas que abordamos juntos, sino también por los ánimos y los momentos de desconexión. Gracias especialmente a toda mi familia: tíos, primos y sobre todo, a mi hermana Ivette y a mis padres Otilia y José María. Gracias por el cariño, el apoyo incondicional, por preguntarme por lo que hacía (aunque sabíais que no iba a saber explicarme), por las impresiones nocturnas, por aguantar mi inexorable conquista de todos los rincones de la casa, ... ¡Gracias por todo!

Por último, gracias a Cristina por todo su apoyo, paciencia y amor durante toda esta etapa de nuestra vida. ¡Y por las correcciones de signos!

A todas las personas arriba referidas, creo sinceramente que esta tesis también os pertenece.

Santiago de Compostela, 16 de enero de 2019

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Contents

Notation

General Math Notation

We use $x(t)$ and $\boldsymbol{x}(t)$ to denote a continuous scalar and multivariate time series, respectively; and $x[n]$ and $\boldsymbol{x}[n]$ to denote a discrete scalar and multivariate time series, respectively. We also use $\boldsymbol{x}$ and $x_{1:N}$ to denote the N samples of the finite discrete time series $x[n]$. Occasionally we use $X(t)$ or $X[n]$ to denote stochastic time series (see below the probability notation). When denoting a collection of data that is not a time series, we prefer the notation $\{\boldsymbol{x}_i\}_{i=1}^{N}$.

Symbol	Meaning
$\dot{y}(t)$ or $y'(t)$	Derivative of $y(t)$ with respect to the independent variable t
$\boldsymbol{x} * \boldsymbol{y}$ or $x[n] * y[n]$	Convolution of $\boldsymbol{x}$ ($x[n]$) and $\boldsymbol{y}$ ($y[n]$)
$\boldsymbol{x} \odot \boldsymbol{y}$	Hadamard product of $\boldsymbol{x}$ and $\boldsymbol{y}$ (element-wise product)
$\propto$	Proportional to (e.g., $y = ax$ can be written as $y \propto x$)
$\triangleq$	Defined as
$\mathcal{O}(\cdot)$	Big-O notation: limiting behavior of a function at infinity
$\#\{\chi\}$	Cardinality of the set χ
$1:n$	Equivalent to $1, 2, ..., n$ (`R`/`Matlab`-like notation)
$\delta(x)$	Dirac delta function

Linear Algebra Notation

Vectors will be denoted with a lowercase bold letter (e.g., $\boldsymbol{x}$), whereas that uppercase bold letters will be reserved for matrices ($\boldsymbol{X}$). Vectors are assumed to be column vectors.

Symbol	Meaning
$\text{tr}(\boldsymbol{X})$	Trace of matrix $\boldsymbol{X}$
$\lvert\boldsymbol{X}\rvert$	Determinant of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{-1}$	Inverse of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{T}$	Transpose of matrix $\boldsymbol{X}$
$\boldsymbol{X}^{-T}$	Transpose of the inverse matrix of $\boldsymbol{X}$
$\boldsymbol{x}^{T}$	Transpose of vector $\boldsymbol{x}$
$\text{diag}(\boldsymbol{x})$	Diagonal matrix made from vector $\boldsymbol{x}$
$\text{diag}(\boldsymbol{X})$	Diagonal vector extracted from matrix $\boldsymbol{X}$
$\boldsymbol{I}$ or $\boldsymbol{I}_d$	Identity matrix of size $d \times d$
$\mathbf{0}$ or $\mathbf{0}^d$	Vector of zeros of size d
x_i	i-th element of vector $\boldsymbol{x}$
X_{ij}	Element (i, j) of matrix $\boldsymbol{X}$
$\boldsymbol{X}_{i,.}$	i-th row of matrix $\boldsymbol{X}$
$\boldsymbol{X}_{.,i}$	i-th column of matrix $\boldsymbol{X}$ (a column vector)

Probability Notation

We denote random and constant scalars with lowercase, random and constant vectors with bold lowercase and random and fixed matrices by bold uppercase. Occasionally, we may use non-bold uppercase to denote scalar random variables for clarity purposes. We use $p()$ for both discrete and continuous random variables.

Symbol	Meaning
$X \mid Y$	Conditional random variable X given Y
$X \sim p$	X is distributed according to distribution p
$\text{Var}(X)$	Variance of X
$\text{Cov}(X, Y)$	Covariance of X and Y
$\mathbb{E}[X]$	The expected value of X
$\mathbb{E}_q[X]$	The expected value of X with respect to distribution q
$\mathbb{H}(X)$, $\mathbb{H}(p)$ or $\mathbb{H}_p(\cdot)$	The entropy of distribution $p(X)$. $\mathbb{H}_p(y)$ highlights that the entropy depends on y
$\mathcal{KL}(p \mid q)$	Kullback-Leibler distribution from p to q
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	($\boldsymbol{x}$ has a) multivariate Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{N}(\mu, \sigma^2)$ or $\mathcal{N}(x \mid \mu, \sigma^2)$	(x has a) univariate Normal distribution with mean μ and variance σ^2
$\mathcal{T}_f(\mu, \sigma^2)$ or $\mathcal{T}_f(x \mid \mu, \sigma^2)$	(x has a) non-standardized-Student's t distribution with f degrees of freedom, location parameter μ and scale parameter σ through the relation $X = \mu + \sigma \cdot T$, being X the non-standardized-Student's t and T a Student's t random variable
$\mathcal{B}(p)$ or $\mathcal{B}(x \mid p)$	(x has a) Bernoulli distribution with probability of success p
$\mathcal{U}(a, b)$ or $\mathcal{U}(x \mid a, b)$	(x has a) uniform distribution with a as minimum value and b as maximum value
$\Gamma(\alpha, \beta)$ or $\Gamma(x \mid \alpha, \beta)$	(x has a) Gamma distribution with shape α and rate β
$\mathcal{NG}(\boldsymbol{x}, \lambda \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \alpha, \beta)$	x has a Gaussian-Gamma distribution with parameters μ, Σ, α and β
$\mathcal{GP}(y(\boldsymbol{\xi}) \mid m(\boldsymbol{\xi}), k(\boldsymbol{\xi}, \boldsymbol{\xi}'))$	$y(\xi)$ is a Gaussian Process with mean function $m(\boldsymbol{\xi})$ and covariance function $k(\boldsymbol{\xi}, \boldsymbol{\xi}')$

Symbol	Meaning
GP-Gamma$(y(\boldsymbol{\xi}), \lambda \mid m(\boldsymbol{\xi}), k(\boldsymbol{\xi}, \boldsymbol{\xi}'), \alpha, \beta)$	$y(\boldsymbol{\xi})$ and λ follow a Gaussian-Gamma process with parameters $m(\boldsymbol{\xi})$, $k(\boldsymbol{\xi}, \boldsymbol{\xi}')$, α and β (see Section 4.2 for details)

Resumen de la tesis

El estudio de fenómenos dinámicos, aquellos cuyo estado evoluciona con el tiempo, siempre ha ocupado un lugar central en todas las disciplinas científicas: desde los trabajos fundacionales de Newton sobre mecánica y astronomía, pasando por la química y la economía, hasta campos más recientes como la bioinformática y la robótica. En el estudio de cualquier sistema, determinar un modelo matemático adecuado de su dinámica es una tarea fundamental. No solo porque la construcción de dicho modelo es imprescindible para poder predecir el futuro, sino también porque a veces, ésta es la única forma de comprobar si nuestro conocimiento sobre las leyes que rigen la evolución del sistema es certero.

La construcción de un modelo dinámico es un proceso tedioso que generalmente requiere de múltiples iteraciones. Para acelerar el proceso, pueden hacerse diversas hipótesis que simplifiquen el tratamiento matemático del sistema en estudio. Una de las más habituales, es la de asumir que dicho sistema puede describirse mediante un modelo lineal. Si esta hipótesis es válida, es posible descomponer el sistema en partes más pequeñas, seguramente más sencillas, y estudiar cada una de estas partes por separado. Una vez comprendido el funcionamiento de cada uno de estos subsistemas, se recupera el comportamiento del sistema original superponiendo las contribuciones de sus partes. Se dice que los sistemas lineales cumplen el principio de superposición.

El hecho de que un sistema lineal pueda ser descompuesto para su estudio da lugar a modelos matemáticos manejables, para los cuales es incluso posible encontrar soluciones cerradas. A pesar de lo conveniente que es un modelo lineal en términos matemáticos, su aplicabilidad en la práctica es muy limitada. Es evidente que la mayor parte de los sistemas observados en la naturaleza no cumplen el principio de superposición. La mayoría de los fenómenos naturales muestran algún tipo de interacción no-lineal que no puede ignorarse en su modelado. A pesar de la dificultad de manejar términos no-lineales, se ha dado la

circunstancia de que la constatación de su importancia en el modelado de los fenómenos naturales ha allanado la adopción de la teoría de sistemas dinámicos en las ciencias de la vida. Ciertamente, nadie esperaría comprender de forma precisa el sistema cardiovascular sin considerar las interacciones entre los pulmones y el corazón.

Otra dificultad habitual es que, en la mayoría de los sistemas con cierta complejidad, como gases, líquidos u organismos microscópicos, las interacciones no-lineales ocurren entre un gran número de subsistemas. De hecho, incluso contando con un conocimiento exacto de las leyes que rigen la evolución de estos sistemas, es común que su resolución sea imposible (tanto en términos matemáticos como computacionales). El enfoque más frecuente para poder abordar su modelado es recurrir a aproximaciones probabilísticas, en el mismo espíritu que la mecánica estadística. En este tipo de aproximaciones el ruido, representando aquellos grados de libertad que no podemos modelar, se convierte en parte integral de la dinámica. Existe otro tipo de sistemas en los cuales el ruido es parte fundamental de su dinámica; en este caso, el ruido no aparece como consecuencia de una aproximación, sino porque su propia interpretación física requiere del concepto de probabilidad. La mecánica cuántica es el ejemplo arquetípico de una teoría que requiere del concepto de probabilidad.

La creciente evidencia de la naturaleza estocástica de la mayor parte de los fenómenos naturales terminó con el paradigma científico predominante en el siglo XIX, que podría resumirse en el intento de modelar la naturaleza a través de ecuaciones diferenciales. Con la llegada del siglo XX, los métodos estadísticos y los modelos probabilísticos ganaron más y más peso en la ciencia.

No solo la física se benefició de los avances en modelos estocásticos, sino también campos como la biología, la medicina o la economía. Estas disciplinas se ocupan de fenómenos macroscópicos y, por tanto, siempre han tenido la necesidad de manejar una gran cantidad de subsistemas. A diferencia de la física, los fenómenos estudiados por estas disciplinas tienen tal complejidad que a menudo se basan en un conjunto de reglas que, al menos desde un punto de vista matemático, no están tan bien definidas. En este contexto, la construcción de un modelo a partir de los principios básicos sigue siendo un reto.

En aquellos casos en los que (1) la no-linealidad, (2) la estocasticidad o (3) la falta de conocimiento impide la creación de modelos usando principios básicos, una alternativa efectiva para el desarrollo de los mismos es su construcción en base a datos experimentales. No se puede sino suponer que esta estrategia será cada vez más habitual. Con la llegada del "Big Data", la cantidad de datos disponibles en cualquier campo de la ciencia y la tecnología

se ha incrementado exponencialmente. El desafío es doble: por una parte son necesarias técnicas de análisis que escalen de forma efectiva con el tamaño y la dimensionalidad de los datos, pero por otra parte éstas no deben apoyarse en modelos puramente deterministas o en métodos lineales, para los cuales existe una extensa literatura.

En esta tesis se desarrollan métodos para construir modelos dinámicos a partir de la observación de series temporales en las que se asume que concurren tanto mecanismos de carácter determinista como de carácter estocástico. Para ello, se ha adoptado un enfoque basado en la interpretación Bayesiana de la probabilidad. En el enfoque Bayesiano, la probabilidad se usa para cuantificar la incertidumbre sobre algo, en lugar de describir la frecuencia de un evento en ensayos repetidos (interpretación frecuentista). Esta concepción aparentemente sencilla de la probabilidad ha dado lugar a todo un conjunto de métodos que permiten hacer deducciones acerca de hechos sobre los que se desea aprender a partir de la observación de datos; y de cuantificar la incertidumbre acerca de dicho conocimiento mediante distribuciones de probabilidad.

Asimismo, vale la pena señalar que, aunque en este trabajo se exploran modelos flexibles del campo del aprendizaje automático capaces de capturar características dinámicas complejas, estos modelos están firmemente basados en conceptos físicos bien conocidos. El principal motivo para adoptar esta aproximación es que, en nuestra opinión, es solo a través de la construcción explícita de modelos por la que se obtiene una comprensión más profunda acerca de cómo se comporta un sistema. Por lo tanto, un modelo creado a partir de datos no solo debe proporcionar predicciones precisas, sino que también debe ayudar al investigador a tomar decisiones de modelado, y esto es más fácil si tanto el investigador como el modelo comparten el mismo vocabulario, el vocabulario de la física.

Dentro del enfoque Bayesiano, esta tesis considera dos aproximaciones principales para caracterizar una serie temporal en la que el ruido forma parte integral de la dinámica. En una primera aproximación, se desarrolla un modelo sencillo que descompone una serie temporal en una componente estocástica y una componente determinista que no interaccionan entre sí. En este modelo, la parte determinista representa la componente del sistema que se desea caracterizar de forma precisa, mientras que la componente estocástica representa aquellos grados de libertad que los métodos experimentales no pueden precisar o aquellos fenómenos que sólo se desean caracterizar de forma aproximada. La segunda de las aproximaciones de esta tesis se ocupa de desarrollar modelos en los que sí existe cierto tipo de interacción entre la componente estocástica y la determinista.

De forma más específica, el trabajo recogido en este documento se organiza tal y como se detalla a continuación. Además de la estructura de la tesis, se resumen las aportaciones más importantes de cada uno de los capítulos:

- En el Capítulo 1 se hace un breve recorrido histórico acerca de cómo ha evolucionado la búsqueda de modelos dinámicos capaces de caracterizar los fenómenos naturales observados. En esta exposición, se introducen de forma intuitiva los conceptos más importantes de la tesis y se discute la necesidad de (1) tener en cuenta las interacciones no-lineales, (2) la inclusión de términos estocásticos para el modelado preciso de sistemas complejos y (3) la necesidad de escalar las técnicas de análisis para afrontar la era del "Big Data".

- En el Capítulo 2 se propone un modelo estocástico-determinista que proporciona una base útil para el estudio de sistemas complejos con correlaciones a largo plazo. Para ello, se asume que las interacciones entre las componentes estocásticas y deterministas son despreciables. La hipótesis de no-interacción plantea un desafío interesante, dado que el modelo debe ser capaz de generar dinámicas complejas sin depender de las interacciones entre las componentes deterministas y estocásticas y, al mismo tiempo, debe ser lo suficientemente realista como para explicar fenómenos naturales. Estos requisitos nos han llevado a considerar los procesos $1/f$ como el bloque de modelado básico del Capítulo 2. En las últimas décadas, se ha hecho evidente que las fluctuaciones de tipo $1/f$ son muy comunes en la naturaleza, apareciendo en campos tan diversos como la física, la biología, la astrofísica, la economía, el lenguaje e incluso la música. La presencia del ruido $1/f$ en sistemas tan distintos ha llevado a los investigadores a plantear la hipótesis de que debe de haber alguna ley fundamental de la naturaleza que explique su ubicuidad. La extensa investigación derivada de esta apreciación ha proporcionado descripciones matemáticas precisas que son capaces de reproducir las propiedades de memoria a largo plazo del ruido $1/f$, la mayoría de ellas basadas en modelos fractales estocásticos como, por ejemplo, el movimiento fraccional Browniano. Por tanto, en el Capítulo 2 se asume que la componente estocástica del modelo de superposición es un movimiento fraccional Browniano. Por otra parte, se hipotetiza que la otra componente del sistema es de banda limitada. Dado que es razonable asumir que es posible predecir con precisión una señal de banda limitada, nos referimos a dicha componente como determinista. Bajo los supuestos de este modelo, se proporciona un método que es capaz

de caracterizar la componente estocástica fractal y de proporcionar una estimación de las componentes deterministas presentes en una serie temporal dada. Más concretamente, las contribuciones más importantes de este capítulo son: (1) la observación clave de que la presencia de la componente determinista produce una desviación medible en la energía por escala que cabría esperar si el sistema fuese un movimiento fraccional Browniano puro; y (2) explotar dicha observación mediante técnicas de estimación Bayesianas y la transformada Wavelet para reconstruir aproximadamente la componente determinista y, al mismo tiempo, caracterizar la componente estocástica fractal. El método se valida sobre señales simuladas y sobre una señal real de origen económico. Estos ejemplos ilustran el potencial del modelo para explorar la dualidad estocástico-determinista de sistemas complejos, así como para descubrir patrones ocultos en series temporales. Además, para facilitar la reproducibilidad de los resultados, los métodos de este capítulo se han implementado, bajo una licencia de código abierto, en el paquete de `R fracdet`. Cabe destacar que liberar el código como un paquete de `R` garantiza que éste se puede instalar fácilmente en cualquier sistema operativo y que sus funciones están bien documentadas. Los resultados recogidos en este capítulo fueron publicados en [48].

- En el Capítulo 3 se aborda la principal limitación del método presentado en el Capítulo 2: no contempla las posibles interacciones entre la componente estocástica y la componente determinista. Para explorar dichas interacciones, se hace uso de ecuaciones diferenciales estocásticas. La aplicación de ecuaciones diferenciales estocásticas al análisis de series temporales recibe cada vez mayor atención debido a su capacidad de describir dinámicas complejas con ecuaciones físicamente interpretables. Las ecuaciones diferenciales estocásticas se han empleado con éxito en la caracterización de fenómenos tan diversos como: difusión de granos en un líquido, turbulencias, fluctuaciones de un plasma, reacciones químicas, atascos de tráfico, fluctuaciones económicas, electroencefalogramas o la expresión de los genes. Intuitivamente, las ecuaciones diferenciales estocásticas combinan una ecuación de evolución determinista junto con un término adicional de ruido. Para simplificar el análisis matemático de las ecuaciones diferenciales estocásticas, generalmente se usa un proceso de Wiener como la fuente de aleatoriedad del sistema. Dicho de otra forma, las capacidades de modelado a largo plazo del movimiento fraccional Browniano se sacrifican para introducir un modelo de interacción. De hecho, nuestras suposiciones acerca de las ecuaciones diferenciales

estocásticas requieren que la serie temporal sea Markoviana. Bajo estas hipótesis, el Capítulo 3 presenta un método no-paramétrico para estimar los términos deterministas y de ruido de una ecuación diferencial estocástica a partir de la observación de una serie temporal discreta. Para introducir el marco Bayesiano en el problema, el método desarrollado hace uso de procesos Gaussianos, los cuales nos permiten definir distribuciones de probabilidad sobre funciones (en este caso, los términos a estimar de la ecuación diferencial estocástica). El uso de procesos Gaussianos como distribuciones a priori permite trabajar directamente en el espacio de las funciones. De esta forma, el aprendizaje de los términos de la ecuación diferencial estocástica tiene lugar de forma natural en este espacio. En este contexto, el aprendizaje consiste en la actualización de las distribuciones a priori (especificadas por los procesos Gaussianos) tras la observación de una realización de la serie temporal. El cálculo de estas distribuciones a posteriori se hace mediante técnicas de inferencia variacional. Las contribuciones más importantes de este capítulo son: (1) proporcionar estimaciones de cualquier componente de ruido, lo que requiere el uso de una aproximación de Laplace; y (2) desarrollar una aproximación de la distribución a posteriori para hacer frente a la complejidad computacional que requiere el uso de procesos Gaussianos; de esta forma es posible realizar predicciones de forma eficiente en series temporales de tamaño medio (con tamaños $N \approx 10^3 - 10^5$). Debe resaltarse que, aunque esta aproximación permite trabajar con una gran cantidad de conjuntos de datos, estos están aumentando rápidamente en tamaño, lo que puede limitar la aplicabilidad del método en el futuro. El problema está relacionado con el uso de técnicas de inferencia variacional, que requieren trabajar con todo el conjunto de datos en cada iteración. El método se valida tanto con datos simulados como con datos reales de economía y paleoclimatología. La aplicación del método a datos reales demuestra su capacidad para captar el comportamiento de sistemas complejos. Además, se ha liberado una implementación del método en el paquete de `R` de código abierto `voila`. Los resultados más importantes de este capítulo dieron lugar a [47].

- El Capítulo 4 intenta superar la principal restricción que impone el uso de ecuaciones diferenciales estocásticas: el supuesto de Markovianidad. Para ello, proponemos un nuevo auto-codificador variacional estructurado con un espacio de fases descrito por ecuaciones diferenciales estocásticas. Este auto-codificador es capaz de descubrir una función, parametrizada como una red neuronal, que transforma una serie temporal arbitraria en una serie temporal Markoviana. Gracias a esta transformación, cualquier

serie temporal es susceptible de ser analizada por medio de los métodos desarrollados en el Capítulo 3. Para evitar que la transformación satisfaga la propiedad Markoviana mediante la creación de un espacio de fases demasiado simple, también se busca una transformación inversa. Esto evita que se descarte información valiosa a la hora de crear el espacio de fases. La transformación inversa también se parametriza con una red neuronal y, por lo tanto, el par de transformaciones actúa como una red neuronal codificador-decodificador. Siguiendo la arquitectura de los auto-codificadores variacionales estructurados, se fuerza a que el espacio de fases siga una dinámica descrita por ecuaciones diferenciales estocásticas mediante el uso de modelos gráficos. Para lograr una mejor optimización, el aprendizaje de las transformaciones y de la dinámica del espacio de fases se realiza simultáneamente. Finalmente, se aborda la otra desventaja principal del método del Capítulo 3: el procedimiento de inferencia se adapta para soportar conjuntos de datos que no caben en la memoria principal. Para ello, se combinan técnicas de optimización estocástica con la inferencia variacional, lo que generalmente se conoce como inferencia variacional estocástica. Las principales aportaciones de este capítulo son: (1) combinar los auto-codificadores variacionales estructurados con los métodos del Capítulo 3, lo que permite modelar de forma efectiva dinámicas no-lineales; (2) encontrar una transformación de series temporales en series Markovianas con significado físico, ya que la nueva representación de los datos puede ser interpretada como un espacio de fases; y (3) explotar avances recientes en sistemas dinámicos basados en procesos Gaussianos para muestrear de manera eficiente el espacio de fases, tal y como requiere el algoritmo de entrenamiento de los auto-codificadores variacionales estructurados. El método se valida utilizando datos de modelos físicos conocidos, demostrando que es capaz de aprender espacios de fases plausibles y de sintetizar nuevas series temporales con características dinámicas similares a las de las series originales. Los resultados de este capítulo son un claro ejemplo de los esfuerzos realizados para aprovechar las capacidades predictivas de los métodos de aprendizaje automático Bayesianos y, al mismo tiempo, mantener la interpretabilidad física de los modelos. La interpretabilidad del espacio de fases del auto-codificador variacional estructurado no solo puede ayudar al investigador a dilucidar modelos físicamente plausibles, sino que también puede inspirar nuevos métodos de análisis tomados de otros campos. Por ejemplo, el espacio de fases proporciona por sí mismo una gran cantidad de información valiosa y, al mismo tiempo, puede ser objeto de nuevas investigaciones por

medio de métodos de análisis de sistemas dinámicos o el estudio de atractores. También cabe destacar que se ha liberado código implementando los métodos principales de este capítulo.

- En el Capítulo 5 se sintetizan los principales resultados y conclusiones obtenidas en esta tesis. Finalmente, se discuten posibles líneas de trabajo futuro.

- En los Apéndices A y B se describe brevemente cómo puede emplearse el software desarrollado en esta tesis. El Apéndice A introduce `fracdet`, un paquete de `R` que implementa el modelo estocástico-determinista propuesto en el Capítulo 2. El Apéndice B resume cómo puede usarse `voila`, un paquete de `R` para la estimación de ecuaciones diferenciales estocásticas mediante el método descrito en el Capítulo 3.

- Finalmente, el Apéndice C detalla la deducción de la matriz de Fisher de una distribución Gaussiana-Gamma, necesaria en la exposición del Capítulo 4.

Chapter 1

Introduction and aims

The study of dynamical phenomena is an important aspect in a wide variety of disciplines, comprising both classical fields, such as astronomy, chemistry and fluid mechanics; and modern fields, such as econophysics, bioinformatics, robotics and drone control. Although nowadays it is an interdisciplinary subject, dynamics was originally a small branch of physics, whose origin can be tracked back to the works of Newton in the mid 1660s. Newton invented differential equations, formulated the laws of motion and applied them to solve the so-called two-body problem: calculating the motion of a planet around the sun. The key to Newton's success was building an accurate model of the celestial bodies from first physical principles (which, of course, he discovered). Determining a proper model of the system's dynamics still remains a fundamental task in the study of any dynamical system. Not only because building a model is necessary for predicting the future, but also because sometimes, this is the only way to gain insight about the laws governing the evolution of the system.

Systems that can be accurately described by a reduced set of principles are indeed a rare exception. Yet, even when this is the case, an accurate model does not imply complete understanding. After the resolution of the two-body problem, mathematicians and physicists devoted considerable efforts to extend Newton's methods to the three-body problem (i.e., the study of the movement of three celestial bodies) [56, Section 3.12]. It turned out that the problem was unsolvable, in the sense of not having an analytical solution. Therefore, the frustrating truth is that, despite having an accurate model of the laws governing the three-body problem, we still lack complete knowledge about its future states. This situation is not specific to the three-body problem. Even a simple mechanical system such as an ideal

pendulum without friction is governed by an equation that is difficult to solve. The swinging of a pendulum is described by

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0, \tag{1.1}$$

where θ is the angle of the pendulum from vertical, g is the gravitational acceleration and L is the length of the pendulum.

The reason why both the three-body problem and the pendulum equation are so hard is that they are described by nonlinear equations. Consider, for example, the nonlinear $\sin\theta$ term from Equation (1.1). Using the small angle approximation $\sin\theta \approx \theta$ for $\theta << 1$, the problem converts into a linear one, which can be easily solved. This leads us to another question. What's the key difference between a nonlinear equation and a linear one, that makes the latter so much easier to solve? The answer lies at the very definition of a linear system: linear systems fulfil the superposition principle and, therefore, can be decomposed into simpler parts [38, Chapter 25]. Since they can be decoupled and decomposed in subsystems, a high-dimensional linear system is essentially the same as a large set of one-dimensional systems whose different variables do not interact.

Most systems observed in nature do not fulfil the superposition principle. Therefore, whenever parts of a system interact, we must resort to nonlinear terms to model them. The development of nonlinear dynamical theory was largely influenced by the work of Poincaré (which tackled the three-body problem), and later benefited from the widespread adoption of computers, which permitted experimenting with, until that moment, unsolvable equations.

An additional difficulty is that, in most systems showing some degree of complexity, the nonlinear interactions usually involve a large number of subsystems. Typical systems encountered in our daily life like gases, liquids or biological organisms consist of, at least, 10^{23} interacting subsystems. From our previous discussion about the pendulum or the three-body problem, it is clear that complete knowledge of the microscopic laws governing these systems provides very little for understanding their macroscopic behavior. Indeed, solving these problems with such a large quantity of interacting particles is still beyond the capabilities of modern computers.

Statistical mechanics emerged due to the need of studying systems with a large number of interacting units. Its development required the formulation of new concepts based on probabilistic and statistical approximations able to handle the large amount of degrees of freedom of macroscopic systems [13, 112]. In these approximations, noise representing those degrees of freedom that we cannot model accurately becomes an intrinsic part of the dynamics.

There is also another type of systems in which noise cannot be disentangled from the description of their dynamics. In this case, noise does not appear as a consequence of an approximation, but because its own physical interpretation requires the concept of probability. Quantum mechanics is the archetypal example of a theory requiring an intrinsic probability [58].

The increasing evidence of the stochastic features of natural phenomena ended with the trend of the nineteenth century science, which could be summarized as the attempt to model nature through differential equations. Instead, emphasis changed towards statistics and probabilistic models, and the study of fluctuating phenomena gained importance.

The observation, in 1827 by the botanist Robert Brown, of the random motion of pollen particles floating in water under the microscope was of special importance for the study of stochastic phenomena. Since then, this phenomenon has been called Brownian motion. Important efforts were made to explain it, but a satisfactory solution had to wait until the independent works of Einstein [35] and Smoluchowski [133], who was also responsible for the development of the mathematical theory of random walks. These works were based on the assumption that the Brownian motion was caused by impacts of the molecules of the liquid in which the pollen was suspended, and they used a statistical description of the force exerted by those molecules.

Langevin further developed these ideas, presenting a new method which eventually resulted in a new field of mathematics now known as Stochastic Differential Equations (SDEs) or Langevin equations [84]. His key idea was to directly include the effect of random forces in the equations of motion, transforming these deterministic differential equations into SDEs. Intuitively, a SDE is a deterministic equation of motion with some noisy fluctuations interfering in its dynamical evolution.

Not only physics benefited from the advances in stochastic models but also fields such as biology, medicine or economics. While physics has been endowed with models backed by rigorous experimental methods, other scientific disciplines exhibit a complexity whose formal description only began to be approachable through the new statistical paradigms, which have permitted the characterization of the order that underlies the complexity of the natural phenomena. The major reasons for this are that these disciplines, that refer to macroscopic phenomena, are specially concerned by the need of handling a large number of interacting subsystems; and because they often rely on a set of rules that are less well understood. In this context, building a model from first principles still remains a challenge.

In those cases where (1) nonlinearity, (2) stochasticity or (3) lack of knowledge prevents

building a model from first principles, an alternative and effective strategy is building it based on observed data from the system. We take for granted that this strategy will become increasingly common due to the explosion of data of complex phenomena that has occurred within all fields of science and technology. The advances in technological trends that have open the door to this massive amounts of information are usually referred to as big data technologies. The challenge is twofold: there is a need to create analysis techniques that scale effectively with the size and dimensionality of the data, but we cannot rely on deterministic dynamical models or linear state space models, for which a large body of research exists.

In this thesis, we consider methods for building dynamical models from complex time series in which noise is an intrinsic part of the dynamics. Furthermore, we would like to be able to effectively model nonlinear dynamics. To this end, we adopt a Bayesian approach. In the Bayesian framework, probability is used to quantify uncertainty about something, rather than describing repeated trials [68]. This deceptively simple interpretation of probability provides the Bayesian framework with a great flexibility, and enables the incorporation of prior knowledge or the combination of information from several sources.

More specifically, we consider two main approaches to characterize complex time series. In Chapter 2 we consider a simple yet realistic model in which there is an stochastic and a deterministic component of the dynamics which barely interact. In this model, the deterministic part represents the component which we would like to study, whereas the stochastic part represents those degrees of freedom that our experiment cannot resolve or we are not interested in. The other approach, presented in Chapters 3 and 4, deals with models in which there is some kind of interaction between the two parts.

In our first approach, the non-interaction hypothesis poses an interesting challenge: the model should be able to simulate complex dynamics without relying on the interaction of the stochastic and the deterministic parts but, at the same time it should be realistic enough to be useful in explaining natural phenomena. These requirements lead us to consider $1/f$ processes as the basic modeling block in Chapter 2. In the past few decades, it has been recognized that $1/f$ fluctuations are common in nature, covering fields as diverse as physics, biology, astrophysics, economics, language and even music [36]. The presence of $1/f$ noise in such a diverse group of systems has led researchers to hypothesize that there must be some profound law of nature that explains its ubiquity. Although several models have been proposed, no single mechanism is nowadays suitable for all the systems. This extensive research has provided accurate mathematical descriptions that are able to capture the long-

term memory properties of $1/f$ fluctuations, most of them based on stochastic fractal models. In Chapter 2 we provide a method that is able to characterize the fractal component and to provide an estimation of the deterministic components present in a given time series. Since both components do not interact, the successful achievement of this objective will greatly simplify the subsequent study of the system, expediting explicit model building. Given that we do not try to characterize the dynamic evolution of the deterministic component, in this chapter we permit the signals to be non-stationary, which greatly broadens the applicability of the method.

The main drawback of the fractal-deterministic approach is that it ignores the very likely interactions between the stochastic and the deterministic components. To address this issue, we turn our attention to systems driven by a SDE. In Chapter 3 a non-parametric method for estimating the terms of a SDE from a densely observed time series is proposed. To enable a flexible modeling of the SDE terms, Gaussian Processes (GPs) are employed. A GP is a non-parametric Bayesian technique that can be interpreted as a prior over functions. Unfortunately, GPs do not scale well with the size of the data. To overcome this issue, a sparse approximation to the GP is employed, which scales linearly with the size of the dataset. Although an improvement with respect to the full GP, the proposed method would still not be able to scale to massive datasets, since the inference procedure requires a full pass over the dataset to compute an update. Furthermore, there is another drawback that may prevent the application of the method to many complex time series. The use of a SDE to describe a system implies that its dynamics can be effectively approximated by a Markov model. In a Markov model, the probability density for the future states of the system is uniquely determined by the current state. However, as argued in Chapter 2, many natural systems show long-term memory and therefore, we cannot expect to model them accurately by using a SDE.

Chapter 4 is mainly devoted to developing a method that is able to discover a Markovian representation of a time series so that the methods of Chapter 3 can be applied. Furthermore, since a deterministic differential equation is just a special case of a SDE, the resulting method can also infer representations of the phase space of a deterministic dynamical model from its observations. It must be stressed that the application of the SDE inference procedure is not conceived to be used on top of the discovered Markovian representation, but that both the Markovian representation and its dynamics (described by a SDE) are discovered simultaneously. This permits a better optimization since both methods can share information. Finally, the other major drawback of the method of Chapter 3 is addressed: the inference procedure is

scaled up to handle massive datasets by means of stochastic optimization techniques.

Finally, Chapter 5 summarises the main results and conclusions obtained in this thesis, and also provides possible directions for future work.

CHAPTER 2

COMPLEX DYNAMICS WITHOUT INTERACTION: DETERMINISTIC AND FRACTAL STOCHASTIC COMPONENTS IN NON-STATIONARY DYNAMICS

In many physical systems, when measuring a quantity along time, we often obtain a time series which seems to fluctuate in a non-periodic, apparently random manner, with complex correlations that extend over all measured time scales. These long-term dependencies are specially evident in the frequency domain and they regularly appear over wide ranges of frequencies as a $1/f$ process, that is, a random process with a power spectral density $S(f) = \text{constant}/|f|^{\gamma}$, where γ receives the name of spectral index [36]. Figure 2.1 shows an example, represented in both time and frequency domains, of a $1/f$ process. A large number of physical and informational systems from biology, chemistry, geology, meteorology, economics or engineering, among others, exhibit such behavior [6], which has fascinated multiple researchers. A $1/f$ process has a long and dynamic memory; the effects of an event persists in time, and the influence of recent events is added to and gradually supersedes the influence of distant events [74]. This long-memory property also accounts for the behavior of informational systems, which exhibit an evolving increase in structure and complexity by accumulating information, combining the strong influence of past events with the influence of current events. Another important feature of $1/f$ processes is the lack of a characteristic time scale, exhibiting sta-

tistical self-similar properties. This permits the explanation of the thermodynamic behavior of a large number of physical systems, based on the concurrence of multiple processes with relaxation times comparable to all time scales of interest. Moreover, this feature is particularly expected in biological systems, since the existence of a preferred frequency of operation would seriously limit the reaction capability of a living system [55, 59]. Figure 2.2 illustrates the self-similar properties of a $1/f$ process by plotting it at different time scales.

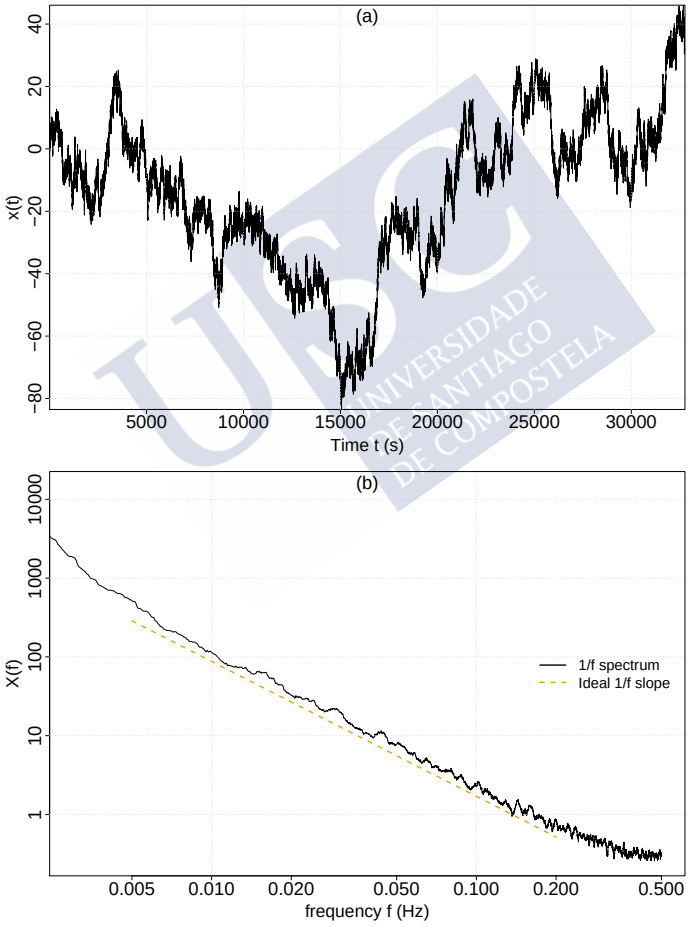

Figure 2.1: $1/f$ process in time (a) and frequency (b). Note that the Power Spectral Density (PSD) in log-log scale can be accurately modeled using a straight line.

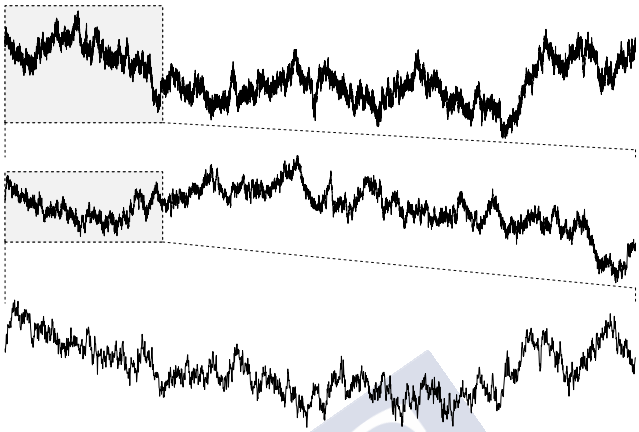

Figure 2.2: A $1/f$ process plotted at different time scales exhibits self-similarity.

Despite the existence of effective mathematical tools for studying such $1/f$ processes, modeling systems with long-term dependencies remains a challenge. This is mainly because real systems do not usually consist of merely stochastic or deterministic components, but they often manifest both random and predictable features. The need to integrate stochastic and deterministic approaches has already been considered by means of a wide variety of techniques. Explicit modeling and non-stationary approaches like [18, 116, 123] are only possible when the mechanisms that underlie the generation of the signal are, at least to some extent, understood. This makes possible building precise mathematical models that account for the behavior of the observed time series. In our opinion, it is through explicit model building that we learn about how a system behaves. However, model building is a complicated task which usually requires many iterations. An accurate characterization of the main stochastic and deterministic features can help in taking modeling decisions.

One of the most successful analysis tools among those that, without relying on explicit modeling, exploit the main properties of the signals, is the Coarse Graining Spectral Analysis (CGSA) [136]. CGSA assumes that the physical quantity under study is a sum of a fractal stochastic model (a fractional Brownian motion) and a sum of stationary sinusoids, and aims to isolate the fractal stochastic spectrum in order to calculate more precisely the parameters that characterize it. Its main drawbacks are that it does not provide any temporal estimation of the harmonic or the fractal stochastic signals, and that it assumes an stationary harmonic

signal.

The success of the CGSA technique is suggestive because it implies that valuable knowledge can be obtained by simply assuming that the system is a superposition of a deterministic and a fractal stochastic component that they do not interact. This approach is far more simple than other methods that model the dynamics of the system by integrating the deterministic and long-memory stochastic contributions. SDEs driven by a fractional noise are the most widely used models taking this approach [10]. However, the mathematical basis of fractional-driven-SDEs is complex and difficult to grasp, which prevents them from becoming more popular.

Therefore, in this chapter we adopt the view of the CGSA model. Our aim is to expedite model building by providing reliable simultaneous estimates of the deterministic and stochastic components present in a system where the deterministic-stochastic interactions may be ignored. We expect these estimates to be useful for exploring the deterministic-stochastic duality in complex systems with long-term correlations, providing a valuable starting point towards the development of interpretable models. To this end, we first propose a realistic, yet simple model, that is able to capture key features of these sort of complex systems. From this model we have developed a method that enables us to characterize the fractal stochastic component and to provide an estimate of the deterministic components of the system. As it will become apparent below, our model may be seen as a generalization of the CGSA that provides temporal estimations and addresses the issue of non-stationarity in the deterministic component.

The proposed model and the signal estimation method have been implemented in `fracdet`, an open-source package for the `R` environment that is freely available at github [44]. A brief overview of the `fracdet` package is provided in Appendix A.

The remainder of the chapter is organized as follows. In Section 2.1, we describe the specific model in which we are going to focus and its statistical properties. We also propose a Bayesian method that exploits these statistical properties to estimate the deterministic and fractal stochastic parts of a signal. In Sections 2.2 and 2.3, we apply our method on simulated data and real data. Finally, the results of this chapter are discussed and some conclusions are given in Section 2.4.

The results presented in this chapter are an adaptation of the material already published in [48].

2.1 Time series with stochastic and deterministic contributions

Let us consider a complex physical system characterized by the temporal series Y. We assume that Y is the result of the superposition of a stochastic series B and some band-limited deterministic signal x:

$$Y[n] = x[n] + B[n] \quad n = 1, 2, \ldots, N.$$

Note that, for clarity, we denote the stochastic time series using uppercase letters.

For the moment, we will focus on the stochastic part and we will assume that our signal $Y[n]$ can be approximated by $Y[n] = B[n]$, where $B[n]$ is an evolutionary random process with long-term dependencies. Therefore, its present behavior is strongly influenced by its entire history. According to a convenient modeling approach [36], long-memory signals are treated as realizations of one of two processes: either a fractional Gaussian noise (fGn), or a fractional Brownian motion (fBm). Since the increments of a non-stationary fBm signal yield a stationary fGn signal, we will focus on the fBm model without any loss of generality. We shall denote a fBm model using $B(t)$ and a discrete fractional Brownian motion (dfBm) by $B[k] = B(k \cdot T_s)$, being T_s the sampling period of the signal [90]. Figure 2.3 highlights the relation between dfBms and discrete fractional Gaussian noises (dfGns).

The dfBm is a stochastic non-stationary process with zero mean that is fully characterized by its variance σ^2 and the so-called Hurst exponent $0 < H < 1$. Due to the relationship between dfGn and dfBm, dfGn models are also fully specified by these two parameters. Figure 2.3 shows both fGn and fBm processes for different values of H. The non-stationary behavior of the dfBm signals is apparent when considering its covariance [39]:

$$\mathrm{Cov}(B[k], B[l]) = \frac{\sigma^2 T_s^{2H}}{2}(|k|^{2H} + |l|^{2H} - |k-l|^{2H}). \tag{2.1}$$

Although non-stationary, dfBm does have stationary increments $B[k+d] - B[k]$, and its probability properties only depend on the lag d, the exponent H and σ^2. This increment process follows a dfGn distribution [90] (see Figure 2.3), and it is self-similar in the sense that, for any $a > 0$, B fulfills

$$(B[k + a \cdot d] - B[k]) \sim a^H (B[k + d] - B[k]).$$

In summary, two important features appear as relevant when analyzing dfBm signals: non-stationarity, which demands for a time-dependent analysis; and self-similarity, which demands

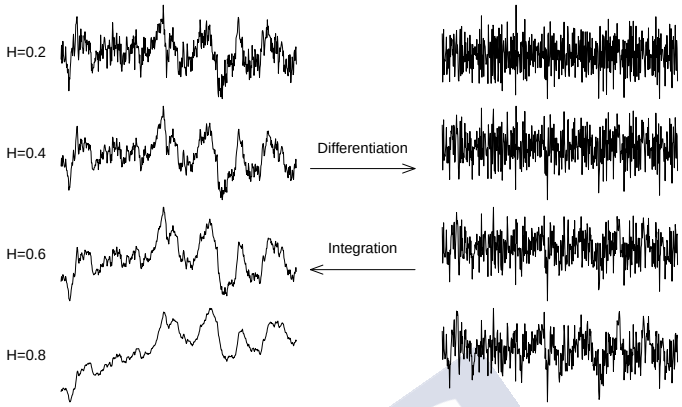

Figure 2.3: Several dfBms (left) and dfGns (right) for different values of the Hurst exponent. Note that the differentiation of a dfBm yields a dfGn.

for a scale-dependent analysis. Wavelet analysis seems to be a proper framework for studying this kind of time series, providing their description in terms of an elementary waveform by means of shifts and dilations, resulting in a time and scale dependent method. Furthermore, they have proved its usefulness in many applications [8, 33, 37, 49, 88]. The next section summarizes the most important statistical properties of the dfBm wavelet coefficients.

2.1.1 Exploring self-similarity with wavelets

Let us assume that B is an infinite signal, so that we can ignore border effects on the wavelet transform. The wavelet transform of B decomposes the signal in terms of a set of oscillating functions $\{\psi_{j,l}\}$, each of which is well localized in time around l and it is related with a resolution level j. In the notation of this chapter, the j-th resolution level is associated with the scale 2^{-j} and thus, larger j correspond to finer scales. Since the set of functions $\{\psi_{j,l}\}$ are generated by parametrizing the function ψ, the latter is called mother wavelet. The wavelet decomposition of B is

$$B[n] = \sum_{j=J}^{\infty} \sum_{l \in \mathbb{Z}} wB_{j,l}[n]\psi_{j,l}[n] + \sum_{l \in \mathbb{Z}} aB_J[n]\phi_{J,l}[n],$$

where $\phi_{J,l}[n]$ is the so-called scaling function (closely related to the mother wavelet), wB_j are the wavelet coefficients of the j-th resolution level, and aB_J are the approximation coefficients associated with some J-th resolution level taken as reference level.

Usually, the wavelet coefficients are computed by using the so-called wavelet filter g, which is uniquely determined by the mother wavelet ψ. Specifically, the wavelet coefficients in the j-th resolution level can be expressed as

$$wB_j[n] = \sum_{p=-\infty}^{\infty} g_j[p-2n]B[p],$$

being g_j the band-pass filter associated with the j-th resolution level and the wavelet filter g. Using mother wavelets with a large number of vanishing moments leads to almost uncorrelated wavelet coefficients wB_j [126]. The wavelet coefficients of a dfBm form a random discrete sequence which is a stationary Gaussian process with zero mean [39]. Following the reasoning exposed in [39] for the variance:

$$\begin{aligned}\mathrm{Var}(wB_j[n]) &= \mathbb{E}(wB_j^2[n]) \\ &= \mathbb{E}\left(\sum_{p=-\infty}^{\infty} g_j[p-2n]B[p] \cdot \sum_{q=-\infty}^{\infty} g_j[q-2n]B[q]\right) \\ &= \sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty} g_j[p-2n]g_j[q-2n]\mathbb{E}\left(B[p]\cdot B[q]\right).\end{aligned}$$

Using Equation (2.1) and given that the dfBm has zero mean and $\mathrm{Cov}\left(B[p], B[q]\right) = \mathbb{E}\left(B[p]\cdot B[q]\right)$, we obtain

$$\mathrm{Var}(wB_j[n]) = \sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty} g_j[p-2n]g_j[q-2n] \cdot \frac{\sigma^2 T_s^{2H}}{2}(|p|^{2H} + |q|^{2H} - |p-q|^{2H}). \quad (2.2)$$

Rearranging the summation indexes and since g_j filters have zero mean [105], Equation (2.2) can be written as

$$\mathrm{Var}(wB_j[n]) = -\frac{\sigma^2 T_s^{2H}}{2}\sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty} |p|^{2H} g_j[q]g_j[q+p].$$

Finally, using the convolution operator and defining $\bar{g}[n] = g[-n]$ yields

$$\mathrm{Var}(wB_j[n]) = -\frac{\sigma^2 T_s^{2H}}{2}\sum_{p=-\infty}^{\infty} |p|^{2H}(g_j * \bar{g}_j)[-p], \quad (2.3)$$

Thus, after selecting a wavelet filter g, $\mathrm{Var}(wB_j[n])$ only depends on j. Although it is not apparent from Equation (2.3), the variance of the wavelet coefficients has a power-law behavior [39]. Note that there is no dependence on the time index n due to the stationarity of the wavelet coefficients of the dfBm. This enables the estimation of $\mathrm{Var}(wB_j)$ from the wB_j sequence. Therefore, after performing the estimation of the variance for the wavelet coefficients in each resolution level, the Hurst exponent and the variance of the dfBm process can be estimated by fitting Equation (2.3) to the data. Figure 2.4 illustrates this procedure.

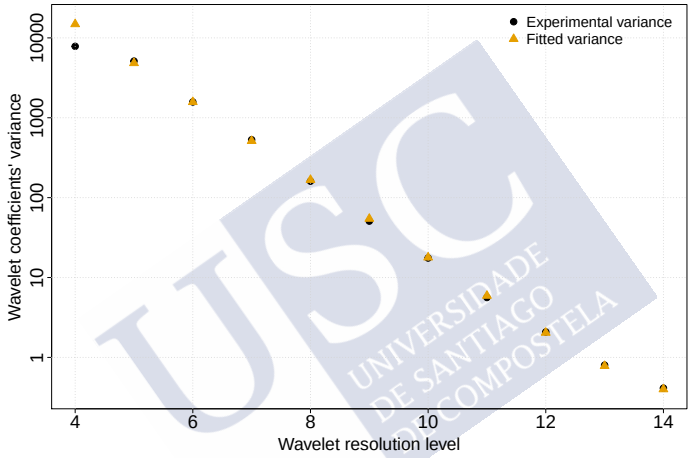

Figure 2.4: H and σ^2 estimation by fitting Equation (2.3) to the experimental wavelet coefficients' variance of a dfBm process. Real parameters of the process are $H = 0.300$ and $\sigma^2 = 1.00$. The resulting estimates are $\hat{H} = 0.302$ and $\widehat{\sigma^2} = 0.996$.

It should be noted that, the smaller j is, the larger the uncertainty in the estimation of $\mathrm{Var}(wB_j)$. This is a consequence of the dependence of the length of wB_j on j: $\mathrm{length}(wB_j) = 2^j$. Thus, we have fewer samples for estimating $\mathrm{Var}(wB_j)$ in the smallest resolution levels, and hence the greater uncertainty in the estimation of their wavelet's variance. Due to this dependence of the estimate's uncertainty on j, the fitting procedure illustrated in Figure 2.4 would benefit from a weighted regression scheme. However, we shall not consider it in the following sections for the sake of simplicity. Instead, we shall just ignore the estimates performed in resolution levels 0-3.

In a more realistic situation, our physical system will not just behave as a dfBm or a dfGn process. Let us return to our initial model for the time series: $Y[n] = x[n] + B[n]$.

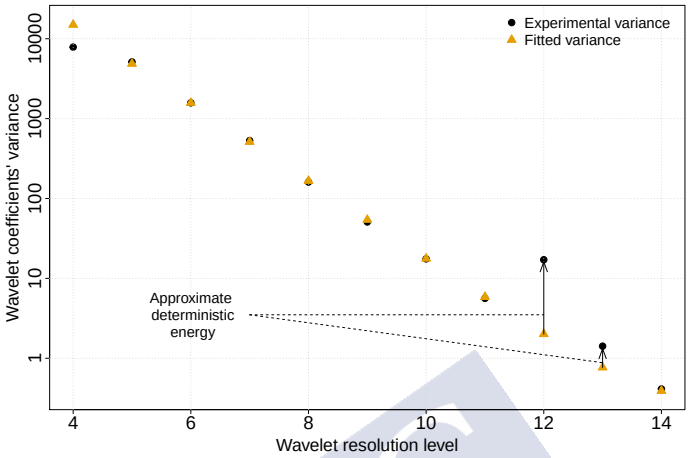

Figure 2.5: H and σ^2 estimation by fitting Equation (2.3) to the experimental wavelet coefficients' variance of a mix of a dfBm and a sine. Real parameters of the process are $H = 0.300$ and $\sigma^2 = 1.00$. Note the deterministic contribution to the 12th and 13th wavelet resolution levels. The resulting estimates are $\hat{H} = 0.303$ and $\widehat{\sigma^2} = 0.998$.

When working with a time series $Y[n]$ involving both dfBm and deterministic contributions, we obtain $wY_j = wx_j + wB_j$ for each wavelet resolution level, being wx_j the deterministic wavelet coefficients. Due to the presence of the deterministic contribution, the variance of the observed wavelet coefficients, $\mathrm{Var}(wY_j[n])$, does not fulfill Equation (2.3). However, let us assume that the deterministic contributions present in Y are band-limited and that they have negligible low-frequency energy in comparison with the $1/f$ power spectrum of the dfBm/dfGn process. Under this assumption, Equation (2.3) will be approximately fulfilled in some wavelet resolution levels, most of them associated with low resolutions (low-frequencies). Thus, the dfBm parameters can be estimated by fitting Equation (2.3) to the wavelet levels free from deterministic contributions. Assuming that $B[n]$ and $x[n]$ interact weakly, the resulting fitted curve enables the estimation of the variance (the energy) of wx in each wavelet resolution level. The present method thus becomes a generalization of CGSA [136], coping with non-stationarity and involving a wider family of deterministic contributions.

It is worth briefly discussing a key implication of the hypothesis that $x[n]$ is band-limited. So far, we have referred to $x[n]$ as the deterministic component, but nothing in our model prevented $x[n]$ to be another stochastic signal. However, if we work under the assumption that

$x[n]$ is band-limited, it is reasonable to expect $x[n]$ to be predictable. Hence, we may refer to this component as deterministic.

Figure 2.5 shows a dfBm mixed with a sine. Since the sine is well located in frequency, its energy is also well located in scale (resolution levels 12 and 13). The fit was performed by using the wavelet resolution levels ranging from 4 to 11. Note that the fit permits the estimation of the cosine energy in levels 13 and 14. The next section explains how to use this information about the deterministic signal to estimate it.

2.1.2 The deterministic contribution

By fitting Equation (2.3) we have not only characterized the dfBm process, but we have also gained information about the deterministic contributions in each wavelet resolution level, information that can be used to obtain an estimate of the deterministic components. A selective wavelet reconstruction has been proven successful for adaptive estimation of signals contaminated with Gaussian background noise [30]. Let us assume that we are given data $Y[n] = x[n]+B[n]$, where x is a potentially complex, temporarily inhomogeneous function; that is, its larger wavelet coefficients occur exclusively near the areas of major activity. Therefore the wavelet coefficients wY would consist of a few large coefficients attributable to the signal x, and many small ones attributable to the noise B. Thus, it is possible to find an estimate $\hat{x}$ (to denoise the data) by shrinking the noisy wavelet coefficients towards 0, and then inverting the wavelet transform of the denoised coefficients. To quantify what is a large wavelet coefficient we need to estimate the size of the noise through its variance. When the noise is not white, as with dfBm, noisy coefficients have a variance that depends on each wavelet resolution level. Thus, the shrinkage procedure should be level dependent. One natural way to do so is via a Bayesian formulation, by placing a prior distribution on each coefficient [1, 20, 131]. A Bayesian approach enables the exploitation of previous knowledge about the properties of the unknown deterministic component x, as that provided by Equation (2.3).

Bayesian statistical properties of the model

Wavelet-based denoising exploits the fact that the wavelet transform enables a sparse representation of a great variety of signals. This means that only a few wavelet coefficients (those with the largest energy) summarize the important features of the signal. Following this reasoning, we propose as prior model for the deterministic coefficients a mixture of a non-standardized-Student's t with f degrees of freedom $\mathcal{T}_f(\cdot)$ and a Gaussian distribution

$\mathcal{N}(\cdot)$:

$$wx_j[n] \mid \phi_j \sim \phi_j \mathcal{T}_f(0, \mu_j^2) + (1 - \phi_j)\mathcal{N}(0, \tau_j^2), \tag{2.4}$$

The Student's t distribution was selected to permit wx_j having a heavy-tailed distribution. Assuming that $\tau_j < \mu_j$, our model is a mixture of a non-standardized-Student's t with squared scale parameter μ_j^2 and f degrees of freedom and a narrow Gaussian centered in 0 with variance τ_j^2. The Student's t characterizes the high-energy coefficients whereas that the Gaussian characterizes the low-energy coefficients. The mixing parameter ϕ_j follows a Bernoulli distribution $\phi_j \sim \mathcal{B}(p_j)$.

The parameters H and σ^2 do not only characterize the dfBm signal, but they also determine the statistical properties of its wavelet coefficients $wB_j[n]$. Taking into account that these coefficients will be approximately uncorrelated, we can write

$$wB_j[n] \mid \widehat{\sigma^2}_j \sim \mathcal{N}(0, \widehat{\sigma^2}_j),$$

where $\widehat{\sigma^2}_j$ is the fitted wavelet variance for the j-th level. Since our knowledge on $\widehat{\sigma^2}_j$ is not exact, we can introduce an informative distribution on $\widehat{\sigma^2}_j$. The dispersion of this distribution will depend on our confidence on the dfBm's parameters estimation. For the sake of simplicity we work with the precision parameter $\lambda_j = 1/\widehat{\sigma^2}_j$. The conjugate prior for a Gaussian with known mean is a Gamma distribution with shape α and rate (or inverse scale) β [51]. Thus, the precision in each wavelet level will be modeled as follows:

$$\lambda_j \sim \Gamma(\alpha_j, \beta_j).$$

Conditioned to the values of wx_j and λ_j, the distribution of the observed wavelet coefficients is

$$wY_j[n] \mid wx_j[n], \lambda_j \sim \mathcal{N}(wx_j[n], \lambda_j^{-1}).$$

Once we have selected proper values for the hyperparameters of the model (see Section 2.1.2), our aim is to obtain the best estimator $\widehat{wx}_j$. The marginal distribution over wY_j given wx_j is a non-standardized-Student's t:

$$wY_j \mid wx_j \sim \mathcal{T}_{2\alpha_j}(wY_j \mid wx_j, (\alpha_j/\beta_j)^{-1})$$

On the other hand, Equation (2.4) yields $wx_j \sim p \cdot \mathcal{T}_f(0, \mu_j^2) + (1-p) \cdot \mathcal{N}(0, \tau_j^2)$, and thus:

$$f_{wx_j}(wx_j \mid wY_j) = \frac{f_{wY_j}(wY_j \mid wx_j) f_{wx_j}(wx_j)}{\int_{-\infty}^{\infty} f_{wY_j}(wY_j \mid wx_j) f_{wx_j}(wx_j)\, dwx_j} = \\ = \frac{\left[p \cdot \mathcal{T}_f(wx_j|0, \mu_j^2) + (1-p) \cdot \mathcal{N}(wx_j|0, \tau_j^2)\right] \cdot \mathcal{T}_{2\alpha_j}\left(wY_j|wx_j, \beta_j/\alpha_j\right)}{p\Pi_{j,1}(wY_j) + (1-p)\Pi_{j,0}(wY_j)} \tag{2.5}$$

being f_X the probability density function of the random variable X and

$$\Pi_{j,0}(wY_j) = \int_{-\infty}^{\infty} \mathcal{N}(x|0, \tau_j^2) \cdot \mathcal{T}_{2\alpha_j}(wY_j|x, \beta_j/\alpha_j)dx,$$

$$\Pi_{j,1}(wY_j) = \int_{-\infty}^{\infty} \mathcal{T}_f(x|0, \mu_j^2) \cdot \mathcal{T}_{2\alpha_j}(wY_j|x, \beta_j/\alpha_j)dx.$$

The best estimator under squared loss function is $\delta(wY_j) = \mathbb{E}(wx_j|wY_j)$, which receives the name of Bayes's decision rule [7]. Applying the expectation operator to Equation (2.5) we obtain the Bayes's rule for our model in the j-th resolution level:

$$\delta_j(wY_j[n]) = \frac{p\Lambda_{j,1}(wY_j[n]) + (1-p)\Lambda_{j,0}(wY_j[n])}{p\Pi_{j,1}(wY_j[n]) + (1-p)\Pi_{j,0}(wY_j[n])} \tag{2.6}$$

being

$$\Lambda_{j,0}(wY_j[n]) = \int_{-\infty}^{\infty} x \cdot \mathcal{N}(x|0, \tau_j^2) \cdot \mathcal{T}_{2\alpha_j}(wY_j[n]|x, \beta_j/\alpha_j)dx,$$
$$\Pi_{j,0}(wY_j[n]) = \int_{-\infty}^{\infty} \mathcal{N}(x|0, \tau_j^2) \cdot \mathcal{T}_{2\alpha_j}(wY_j[n]|x, \beta_j/\alpha_j)dx,$$
$$\Lambda_{j,1}(wY_j[n]) = \int_{-\infty}^{\infty} x \cdot \mathcal{T}_f(x|0, \mu_j^2) \cdot \mathcal{T}_{2\alpha_j}(wY_j[n]|x, \beta_j/\alpha_j)dx,$$
$$\Pi_{j,1}(wY_j[n]) = \int_{-\infty}^{\infty} \mathcal{T}_f(x|0, \mu_j^2) \cdot \mathcal{T}_{2\alpha_j}(wY_j[n]|x, \beta_j/\alpha_j)dx.$$

Figure 2.6 shows an illustrative representation of the Bayes rule that results from Equation (2.6). Note that the Bayes rule shrinks the smaller coefficients towards 0 while it keeps the largest ones almost unaltered.

The final estimate $\hat{x}$ is obtained by performing the inverse wavelet transform of the resulting wavelet coefficients $\widehat{wx}_j$.

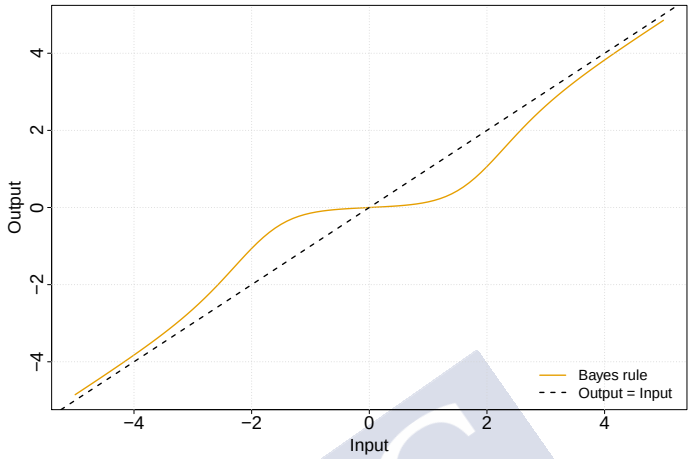

Figure 2.6: An illustrative example of the Bayes decision rule that results from Equation (2.6)

Parametrization of the model through hyperparameters

To complete the description of the estimation algorithm it is necessary to provide some guidelines on the selection of the hyperparameters and its impact on the performance of the method. Note that the election of these hyperparameters is independent for each wavelet resolution level. The prior distribution for the precision is $\Gamma(\alpha_j, \beta_j)$ so that its mean is α_j/β_j and its variance is α_j/β_j^2. For a fixed level j we may choose α_j and β_j, therefore $\beta_j/\alpha_j = \widehat{\sigma_j^2}$ and $\alpha_j/\beta_j^2 = (\Delta\sigma_j^2)^2/(\widehat{\sigma_j^2})^4$, where $\Delta\sigma_j^2$ is the estimated regression error in the j-th wavelet resolution level.

The parameter τ_j characterizes the dispersion of the smaller deterministic wavelet coefficients. The contribution of any wavelet coefficient $w_j[n]$ to the final reconstructed signal is $w_j[n] \cdot \psi_j$, being ψ_j the wavelet of the j-th resolution level. This contribution should be negligible when compared to the mean amplitude of the deterministic signal A_x. Given the shape of the Gaussian distribution, $3\tau_j$ can be interpreted as a soft bound of the negligible coefficients. Thus we require $3\tau_j \cdot A_{\psi_j} \approx k_j \cdot A_x$, being A_{ψ_j} the amplitude of the j-th wavelet and k_j a small number [20]. As a practical rule for estimating A_x we propose detrending the Y series (since we associate the long-term fluctuations with the dfBm) and setting $A_x = \text{MAD}(dY)$, being MAD the Median Absolute Deviation and dY the detrended series. An appropriate value for k_j could be $k_j = k = 0.01$, meaning that the contributions of the largest negligible

deterministic coefficients at a given scale j should not exceed 1% of the amplitude of the deterministic signal.

Since $\phi_j \sim \mathcal{B}(p_j)$, p_j is the probability of finding a non-negligible deterministic wavelet coefficient in the j-th resolution level. Therefore, finding a proper value of p_j first requires defining what is negligible. To that end, we resort to the "universal threshold" of Donoho et al. [31], which can be understood as an upper bound on the size of the noise. The universal threshold was derived under the assumption that some signal was contaminated with independent and identically distributed (iid) Gaussian noise $O[n] \sim \mathcal{N}(0, \sigma^2)$. The form of the universal threshold when observing N coefficients of the contaminated signal is

$$t = \sqrt{2\log(N)}\sigma.$$

The rationale for this threshold is that, as $N \to \infty$,

$$p(\max_n\{|O[n]|\} < t) \to 1,$$

and therefore, the original signal could be estimated by removing (thresholding) the coefficients smaller than t, since asymptotically the vast majority of noise would be eliminated. Note that the threshold is biased towards overestimating the magnitude of the noise, since σ is multiplied by $\sqrt{2\log(N)}$. That's the reason why, as previously stated, the universal threshold can be interpreted as an upper bound on the size of the noise.

Following [20], we use the universal threshold to define a boundary between the negligible and non-negligible signal coefficients. We propose as follows:

$$p_j = \frac{\#\{wY_j[n] : |wY_j[n]| > \sqrt{2 \cdot \log(2^j)(\widehat{\sigma^2}_j + \tau_j^2)}\}}{2^j},$$

where # denotes the cardinality of a set.

Finally, the variance of the observed wavelet coefficients is

$$\mathrm{Var}(wY_j) = p_j\mu_j^2\frac{f}{f-2} + (1-p_j)\tau_j^2 + \widehat{\sigma^2}_j. \tag{2.7}$$

For a fixed f, it is possible to estimate μ_j^2 from Equation (2.7). Increasing the value of f yields more conservative Bayes' rules. To exploit the heavy tails of the Student's t distribution, we suggest the use of f ranging from 3 to 10, since for moderately large values of f the Student's t approaches a Gaussian distribution.

Effects of the observational noise

Typically, most experimental time series are affected by observational noise. For the sake of completeness, we briefly review how it would affect the method proposed in the previous sections. Suppose now that, to account for the effect of the observational noise, we model our time series as

$$Y[n] = x[n] + B[n] + O[n],$$

where $O[n] \sim \mathcal{N}(0, v^2)$ represents the effect of the undesired observational noise. Although simple, this assumption is accurate enough for most applications. We assume that the strength of the noise does not completely mask the main features of both $x[n]$ and $B[n]$. An important property of an orthonormal transform (like the wavelet transform) of identically independently distributed Gaussian noise is that the transformed noise has the same statistical properties as the original one. Therefore, all the stochastic contributions can be modeled in wavelet domain as follows:

$$wB_j[n] + wO_j[n] \mid \widehat{\sigma^2}_j, v^2 \sim \mathcal{N}(0, \widehat{\sigma^2}_j) + \mathcal{N}(0, v^2) \sim \mathcal{N}(0, \widehat{\sigma^2}_j + v^2).$$

Thus, Equation (2.6) can still be applied as decision rule provided that the precision parameters are used to model $\lambda_j = 1/(\widehat{\sigma^2}_j + v^2)$ and that we have a reasonable estimate (or prior value) of v^2.

We also propose another approach which tries to exploit the fact that, in most experimental time series, the contribution of the dfBm will predominate over the observational noise in the low frequencies. Intuitively, this is due to the fact that observational noise is modeled as an stochastic term without drift. Therefore, the proposed estimation method can be used as if the observational noise did not exist to obtain an estimate of $x[n] + O[n]$, $\widehat{x[n] + O[n]}$. Then, any thresholding or shrinkage algorithm such as [1, 20, 31, 131] could be applied on $\widehat{x[n] + O[n]}$ to obtain the final estimate of $x[n]$ and an estimate of v^2. The estimate of v^2 can be subsequently used to tune the estimate of the dfBm parameter σ^2.

2.2 Validation over synthetic data

In this section, we compare the performance of our proposal with the universal thresholding rule of Donoho et al. [31] using synthetic data. In addition to being one of the most widely used thresholding methods, its selection for the comparison was also due to the availability of an open-source implementation of the algorithm [99]. The fitted variances for each resolution

level $\widehat{\sigma^2}_j$ were provided to Donoho's algorithm for a more fair comparison. Note that the use of synthetic data allows us to know the exact mathematical model that generated the data, and therefore which is the correct answer the model should provide, unlike real data. The synthetic signals consist of a mixture of a dfBm with either $H = 0.3$ or $H = 0.8$ and some deterministic series. For each case, simulations were conducted using different Signal-to-Noise Ratios (SNRs). We employed a set of complex, band-limited time series as deterministic components, so that the resulting signals met the assumptions of our model. All the synthetic signals had a length of 2^{15} points.

The "Rossler" signal from Table 2.1 was generated by sampling $r_x \cdot r_y + r_z$ every second, where r_x, r_y and r_z are the cartesian components of the Rossler system:

$$\begin{aligned}\dot{r}_x &= -(r_y + r_z)\\ \dot{r}_y &= r_x + 0.2 \cdot r_y\\ \dot{r}_z &= 0.2 + r_z \cdot (r_x - 5.7)\end{aligned}$$

The "Lorenz" signal data was obtained by studying the z component of the Lorenz system (l_z) in steps of 0.01 seconds:

$$\begin{aligned}\dot{l}_x &= 10 \cdot (l_y - l_x)\\ \dot{l}_y &= l_x \cdot (28 - l_z) - l_y\\ \dot{l}_z &= -\frac{8}{3} l_z + l_x \cdot l_y\end{aligned}$$

The "Chirp" signal was generated using a linear chirp whose frequencies ranged from 0.025 to 0.2 Hz, and sampling it every second. Finally, the "Henon" signal was obtained from the y component of the Henon map (h_y):

$$\begin{aligned}h_x[n] &= 1 - 1.4 \cdot h_x^2[n-1] + h_y[n-1]\\ h_y[n] &= 0.3 \cdot h_x[n-1].\end{aligned}$$

Table 2.1 shows the mean square error of the estimates obtained with our Bayesian algorithm and with Donoho's universal threshold for the different values of H and SNRs. As shown in Table 2.1, our model shows a slightly better performance than [31]. This is not surprising, since we developed our model for this particular problem and it was inspired by the pioneering work of [1, 20, 131]. In this sense, we are not trying to establish a new denoising-algorithm

Signal	SNR	H=0.3		H=0.8	
		Proposal	[31]	Proposal	[31]
Rossler	3	**0.248 (0.169)**	0.684 (0.179)	**0.253 (0.112)**	0.431 (0.141)
	5	**0.188 (0.149)**	0.558 (0.196)	**0.134 (0.032)**	0.320 (0.094)
	7	**0.149 (0.134)**	0.467 (0.182)	**0.101 (0.020)**	0.267 (0.078)
Lorenz	3	**0.578 (0.027)**	0.974 (0.012)	1.161 (0.416)	**1.006 (0.010)**
	5	**0.438 (0.022)**	0.935 (0.022)	1.024 (0.241)	**1.002 (0.007)**
	7	**0.355 (0.017)**	0.889 (0.027)	**0.959 (0.154)**	1.001 (0.005)
Chirp	3	**0.431 (0.075)**	0.827 (0.095)	**0.536 (0.090)**	0.877 (0.069)
	5	**0.361 (0.058)**	0.681 (0.117)	**0.460 (0.081)**	0.779 (0.087)
	7	**0.316 (0.041)**	0.562 (0.099)	**0.414 (0.077)**	0.703 (0.095)
Henon	3	**0.380 (0.207)**	0.886 (0.144)	**0.235 (0.019)**	0.567 (0.059)
	5	**0.249 (0.075)**	0.786 (0.094)	**0.190 (0.022)**	0.484 (0.060)
	7	**0.231 (0.055)**	0.741 (0.066)	**0.161 (0.020)**	0.424 (0.054)

Table 2.1: Mean square error of our Bayesian estimation algorithm and the algorithm from Donoho et al. [31] using different test functions, and for different levels of SNRs. Here, the "Signal" is the deterministic component and "Noise" is the dfBm component. The hyper-parameters for our model were selected following the rules exposed in Section 2.1.2 with $f = 5$. Standard errors are given in parentheses. The best result for each test is marked in bold (smaller is better).

standard, but we are only testing that our model performs well enough when used to characterize a system where deterministic and stochastic phenomena concur. Figures 2.7-2.10 show the results obtained with our model on different test signals. The upper panels show the original signals consisting of a band limited deterministic signal and a dfBm. The lower panels show the deterministic component and its estimate obtained using our method. The SNR was set to 3 in all the examples.

Table 2.1 and Figures 2.7-2.10 illustrate the ability of our proposal for separating the deterministic component from the stochastic contributions. To completely characterize the observed signal $Y[n]$, the H and σ^2 parameters of the dfBm should also be estimated. Table 2.2 shows the estimates obtained for both parameters using the same synthetic data employed in Table 2.1. Since the estimates are quite accurate and these parameters fully specify the properties of the dfBm, we could reproduce the $1/f$ statistical features of the observed signal $Y[n]$. In this sense, we could generate new surrogates of $Y[n]$ using $\hat{x}[n]$ and synthesizing a new dfBm characterized by $\hat{H}$ and $\widehat{\sigma^2}$. For example, a wavelet-based simulation method of dfBm can be found in [39].

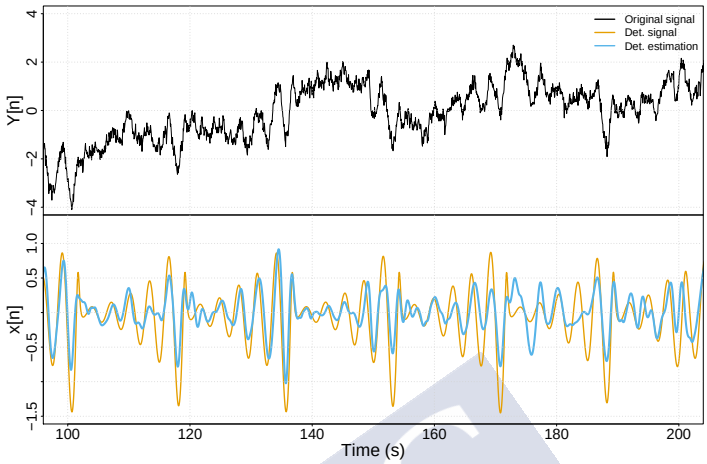

Figure 2.7: Deterministic estimation from the "Rossler" signal (sampled at 20 Hz for a smoother visualization) and a dfBm with $H = 0.3$.

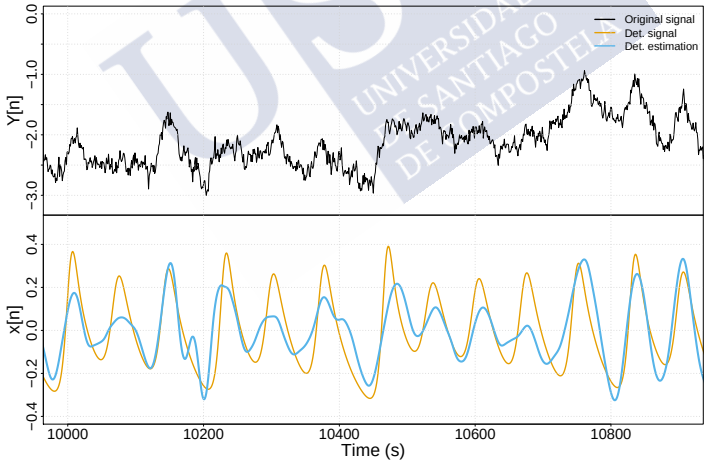

Figure 2.8: Deterministic estimation from the "Lorenz" signal and a dfBm with $H = 0.3$.

2.3 Application to economic data

We have tested the method presented in this chapter by applying it to the study of the mean daily prices of the 95 RON (Research Octane Number) unleaded petrol in the fuel stations of Spain (from 2011-02-13 to 2016-05-02, see Figure 2.11). Since many financial models incorporate

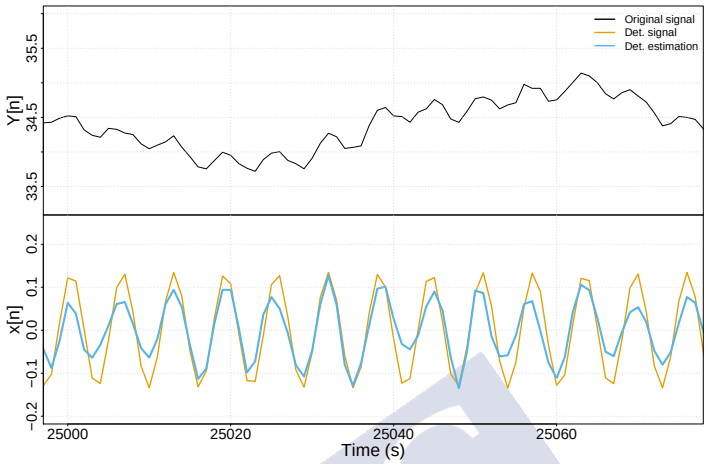

Figure 2.9: Deterministic estimation from the "Chirp" signal and a dfBm with $H = 0.8$.

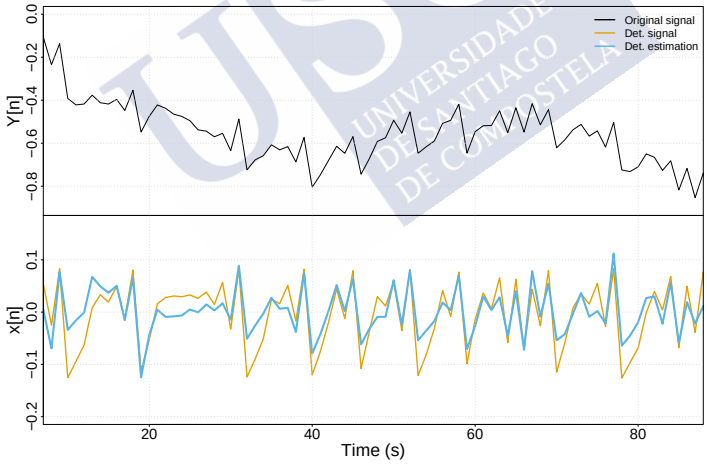

Figure 2.10: Deterministic estimation from the "Henon" signal and a dfBm with $H = 0.8$.

Brownian motion (a fBm with $H = 0.5$), our model provides a reasonable framework for studying the unleaded petrol data. Figure 2.12 shows the fitted variance for each wavelet resolution level. As expected, the variance-resolution curve is compatible with a $1/f$ spectrum, and the estimated Hurst exponent is close to the theoretical exponent of the Brownian motion, $\hat{H} \approx 0.56$. The most interesting deviations from the theoretical variance distribution occur

Signal	SNR	H=0.3		H=0.8	
		H estimate	σ^2 estimate	H estimate	σ^2 estimate
Rossler	3	0.321 (0.043)	0.297 (0.138)	0.820 (0.050)	0.324 (0.095)
	5	0.325 (0.043)	0.173 (0.083)	0.812 (0.047)	0.204 (0.061)
	7	0.324 (0.043)	0.123 (0.059)	0.810 (0.049)	0.148 (0.047)
Lorenz	3	0.300 (0.022)	0.334 (0.020)	0.802 (0.018)	0.335 (0.015)
	5	0.303 (0.016)	0.199 (0.009)	0.799 (0.023)	0.201 (0.010)
	7	0.301 (0.022)	0.143 (0.009)	0.799 (0.023)	0.144 (0.007)
Chirp	3	0.315 (0.035)	0.306 (0.108)	0.807 (0.043)	0.342 (0.076)
	5	0.314 (0.035)	0.186 (0.064)	0.807 (0.043)	0.205 (0.046)
	7	0.314 (0.035)	0.133 (0.046)	0.807 (0.043)	0.146 (0.033)
Henon	3	0.313 (0.053)	0.325 (0.177)	0.827 (0.030)	0.305 (0.044)
	5	0.288 (0.022)	0.226 (0.051)	0.818 (0.036)	0.192 (0.034)
	7	0.276 (0.016)	0.188 (0.028)	0.815 (0.037)	0.139 (0.024)

Table 2.2: H and σ^2 estimates of our Bayesian estimation algorithm using different test functions, and for different levels of SNRs. To simplify the table, the deterministic test functions were normalized to unit energy. Therefore, the theoretical σ^2 is $1/\text{SNR}$ for all rows.

in levels 9 and 10, which correspond to fluctuations occurring on a 2-8 daily basis. The deviations occurring in levels ranging from 3 to 4 also seem to be important. However, the variance estimates of these resolution levels have great uncertainty and hence, the resulting deterministic estimates would also be subject to strong uncertainties. Since the main aim of this section is to validate our proposal with a real application, they will not be considered. In the remainder of this section, we will focus on the deterministic component extracted from resolution levels 9-10.

Both the distribution of the original data and its deterministic component against the weekday are shown in Figure 2.13. Even though they spread over different scales, the deterministic component shows a more clear pattern. On average, the prices on Mondays tend to decrease whereas that they slightly increase during the weekends. The increase in prices during the weekends is likely driven by increased demand. On the weekends people tend to travel longer distances with their cars, and they also often take advantage of their free time to refuel. The fall of the petrol prices on Mondays is a very well-documented effect in Europe that is referred to as the "Monday effect" [25]. The explanation of this effect is that the member states of the European Union must report their fuel prices (petrol, diesel and biofuel) to the European Commission every Monday for the "European Commission Oil Bulletin". By lowering their prices on the measurement day, petrol stations distort the official statistics. Table 2.3 sum-

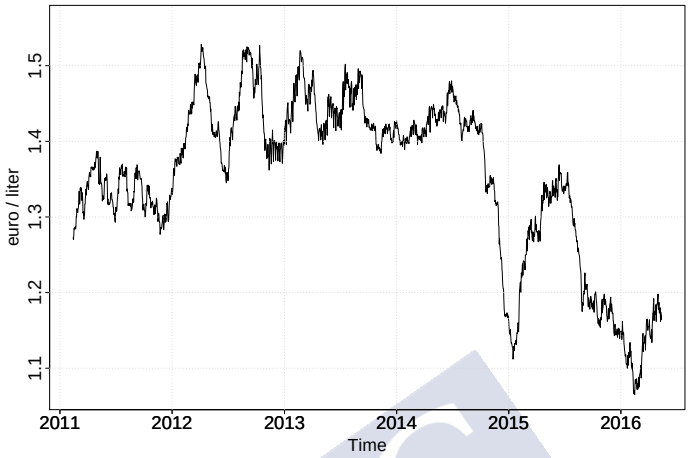

Figure 2.11: Mean daily prices of the 95 RON unleaded petrol in the fuel stations of Spain.

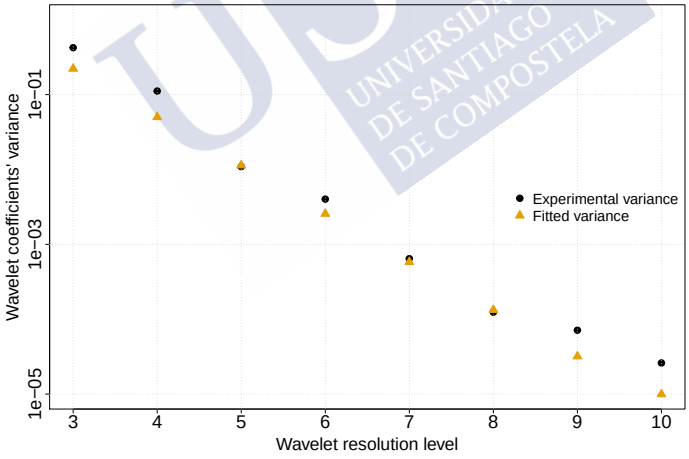

Figure 2.12: Variance depending on the wavelet resolution level for the 95 RON unleaded petrol data. The resulting Hurst estimate is $\hat{H} \approx 0.56$. Note the deviation from the theoretical variance in resolution levels 9 and 10.

marizes the mean price differences between Sunday-Monday and Tuesday-Monday obtained from the original data and from the deterministic component estimated through our method.

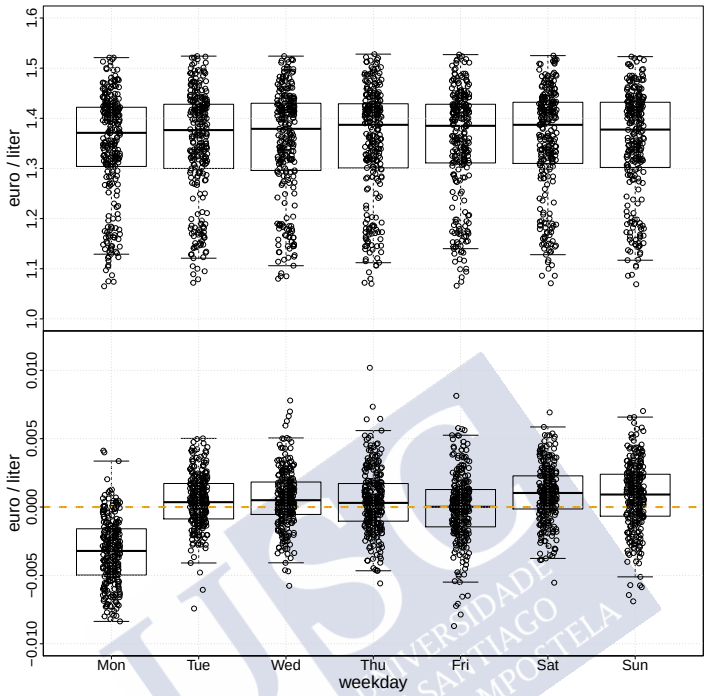

Figure 2.13: Upper panel: Boxplot showing the distribution of the original petrol price versus the weekday. Lower panel: Boxplot showing the distribution of the deterministic component of the petrol price versus the weekday. Note the decrease of the prices on Monday and the slight increase during the weekend.

Table 2.3 illustrates that the price differences in the deterministic component are smaller than the price differences obtained directly with the original data. This is a consequence of the assumptions of our model and the resulting shrinkage algorithm. Our model assumes that the petrol prices result from a linear superposition of a dfBm and a deterministic signal, and this last contribution is estimated after a shrinkage operation of the original time series. The shrinkage operation can be intuitively understood by considering that the algorithm distributes the original signal's energy between the two components assumed by our method.

Figure 2.13 and Table 2.3 illustrate how our proposal can be used for unveiling interesting patterns buried in complex time series even when starting with a very limited knowledge of the phenomena under study. Table 2.3 also illustrates that as a result of the assumptions of our model, the resulting deterministic estimates will always be smaller than the original data.

Mean €cents price difference	2011	2012
Det. Mon-Sun	-0.087 (0.325)	-0.260 (0.312)
Mon-sun	-0.189 (0.482)	-0.483 (0.626)
Det.Tue-Mon	0.223 (0.332)	0.420 (0.341)
Tue-Mon	0.413 (0.670)	0.868 (0.883)

Table 2.3: Mean differences between Sunday-Monday prices and Tuesday-Monday prices of the 95 RON unleaded petrol in Spain ("Mon-Sun" and "Tue-Mon", respectively) and its deterministic component ("Det. Mon-Sun" and "Det. Tue-Mon"). Standard errors are given in parentheses.

2.4 Discussion

We have presented an algorithm that, under the assumption that a simple superposition model holds, is able to characterize the fractal stochastic component and estimate the deterministic components present in a time series with long-term correlations. In this model, the fractal stochastic component is assumed to be a dfBm/dfGn and the deterministic signal is assumed to be band-limited. We further hypothesized that both components do not interact. The main contribution of our proposal is the observation that, since the energy of self-similar processes changes with the scale following a power-law relation, deviations from this behavior permit the estimation of the energy distribution across scales of a deterministic signal superimposed on $1/f$ noise, which at the same time enables its approximate reconstruction using well-known Bayesian estimation techniques. We tested the method on simulated time series and we compared its performance with Donoho's universal threshold, showing that our method is able to reconstruct the deterministic series with great accuracy.

It must be noted that the model proposed in this chapter is a generalization of the CGSA [136]. The CGSA model assumes stationary harmonic components for the deterministic signal, whereas we just require it to be band-limited, without any assumption about its stationarity. CGSA focuses on estimating the fractal spectrum, whereas our proposal focuses on providing an estimate of the deterministic signal in time. In stationary conditions and when the deterministic signal can be approximated by a sum of sinusoidal terms (when CGSA model holds), both algorithms can complement each other. One of the flaws of our method is the need of selecting the wavelet resolution levels to use in the regression procedure (Equation (2.3)), which is a subjective step that depends on the experience of the researcher. When CGSA assumptions hold, the selection of these levels may be avoided by means of the CGSA

algorithm. If those assumptions are not fulfilled, to the best of our knowledge, there is a lack of tools for objectively estimating the fractal properties of a time series that does not uniquely contain fractal stochastic components. The design of some mechanism to automatically select the wavelet resolution levels free from deterministic contributions should be considered in future improvements.

We have studied the evolution of the mean daily prices of the 95 RON unleaded petrol in the fuel stations of Spain as an example of the application of our method to real data, demonstrating its usefulness for discovering deterministic patterns and gaining insights about the system under study.

Although this example is suggestive, we should be aware of the main limitation of our model: it ignores the possible interactions between the deterministic and stochastic components. In relation with this topic, it would be interesting to consider the inclusion of some flexible interaction mechanism in our model to take into account the possible weak reinforcement-inhibition phenomena. However, note that with strong interactions our model $Y[n] = x[n] + B[n]$ will not be a valid approximation anymore, and thus the study of strong interactions will probably require a different approach. The most natural one would be considering the evolution of the system at each step: $dY(t) = dx(t) + dB(t)$, where we have used continuous notation for clarity in the following exposition. This equation suggests that, at each time step, the system evolves following some deterministic forcing $dx(t)$ but it is also subject to random perturbations $dB(t)$. The problem is now much more general and hence, it is usually simplified by assuming that the deterministic forcing depends on $Y(t)$: $dx(t) \triangleq f(Y(t))dt$. Furthermore, to permit richer interactions between the deterministic driving force and the noise perturbations, a multiplicative noise term can be used, which yields: $dY(t) = f(Y(t))dt + \sqrt{g(Y(t))}dB(t)$, which may remind us of the expression for a SDE.

Build on top of this informal discussion, the following chapter deals with the use of SDEs to model time series with deterministic-stochastic interactions. Although it is possible to use SDEs driven by fBms, we shall focus on another kind of noise (a Wiener process), which is more widely used and greatly simplifies the discussion of what the differential of a noise $dB(t)$ is.

CHAPTER 3

MODELING DETERMINISTIC-STOCHASTIC INTERACTIONS IN MARKOVIAN DYNAMICS

SDEs, also referred to as Langevin equations, provide an effective framework for modeling complex systems comprising a large number of subsystems which show irregular fast dynamics that can be treated as fluctuations or noise. Intuitively, SDEs couple a deterministic equation of motion with noisy fluctuations interfering in its dynamical evolution. They have demonstrated their usefulness in a wide range of applications: diffusion of grains in a liquid [84], drift of particles without flux [83], turbulence [40, 41], fluctuations in plasma [94], variations in quasar's optical flux [72], chemical reactions [54], traffic flow dynamics [91], quantitative finance [52], gene expression [102], electroencephalography analysis [5], etc. (see [42] for a complete review with applications). The only specific requirements that modeling through SDEs imposes are stationarity and Markovianity. Recent advances have enabled the application of SDEs to time series that do not meet some of these requirements. For example, SDE-modeling has been successfully applied to non-stationary conditions by means of additional measurable properties [86, 95] or by assuming a parametric model in which the parameters themselves evolve according to stationary stochastic processes [115]. On the other hand and concerning Markovianity, consider for example [119], in which Langevin-like equations have been properly estimated in the presence of observational noise spoiling the Markov properties of the experimental time series. We shall not exploit these advances, and we shall assume that the stationarity and Markovianity assumptions hold.

In this chapter, we consider a system that may be represented by a continuous-time uni-

variate Markov process $x(t)$ described by the SDE:

$$dx(t) = f\big(x(t)\big)dt + \sqrt{g\big(x(t)\big)}dW(t), \tag{3.1}$$

where $W(t)$ denotes a Wiener process. The Wiener process has independent Gaussian increments $W(t+\tau) - W(t)$ with zero mean and variance τ. Thus, we may intuitively think of $dW(t)$ as white noise, which is the source of randomness of the system. Note that the stationarity and Markovianity assumptions follow from the properties of the Wiener process. The function f defines a deterministic drift and g modulates the strength of the noise term. If the function g is constant, the resulting noise term $\sqrt{g\big(x(t)\big)}dW(t)$ is said to be additive. Otherwise, the noise is said to be multiplicative. The functions f and g are usually referred to as the *drift* and *diffusion* coefficients. Note that, although we could have denoted $x(t)$ with uppercase to highlight its stochastic nature, we decided not to do it to keep the notation uncluttered.

When studying complex dynamical systems, the large number of degrees of freedom and the nonlinear interactions between the subsystems involved in the dynamics usually hinders obtaining an exact knowledge of the functional forms of both the drift and diffusion coefficients. This leads to the problem of its non-parametric estimation from the observation of an experimental time series $\boldsymbol{x}$, which is usually a sampled version of the underlying continuous process $x(t)$ (the reason for using $N+1$ as the number of samples will be apparent at the beginning of Section 3.1).

The most widely used non-parametric estimation methods exploit the theoretical expressions for both the drift and diffusion terms [42]:

$$\begin{aligned} f(\xi) &= \lim_{\tau\to 0} \frac{1}{\tau}\mathbb{E}_{x(t+\tau)}\left[x(t+\tau) - x(t) \mid x(t) = \xi\right], \\ g(\xi) &= \lim_{\tau\to 0} \frac{1}{\tau}\mathbb{E}_{x(t+\tau)}\left[\big(x(t+\tau) - x(t)\big)^2 \mid x(t) = \xi\right], \end{aligned} \tag{3.2}$$

where $\mathbb{E}$ denotes the expectation operator. Equation (3.2) suggests the possibility of estimating the dynamical coefficients $f(\xi)$ and $g(\xi)$ by computing "local" means in a small neighborhood of ξ [42]. Typically, the local means are computed after binning the domain of $\boldsymbol{x}$ using bins of size ϵ, like in a histogram:

$$\begin{aligned} \hat{f}(\xi) &= \frac{1}{N_\xi} \sum_{x[i]\in B(\xi,\epsilon)} [x[i+1] - x[i]], \\ \hat{g}(\xi) &= \frac{1}{N_\xi} \sum_{x[i]\in B(\xi,\epsilon)} [x[i+1] - x[i]]^2, \end{aligned}$$

where $\hat{f}$ and $\hat{g}$ represent the estimates, $B(\xi, \epsilon)$ denotes the bin in which ξ falls and N_ξ is the number of points from $\boldsymbol{x}$ falling in that bin.

The most obvious limitation of this histogram-based approach is that the estimations highly depend on the choice of ϵ. Furthermore, it is not obvious how we should select the size of the bin. More sophisticated approaches rely on replacing the mean of the bins with the mean of the k-nearest neighbors [60]. However, the free parameter of this approach, k, must still be heuristically selected.

Another refinement of those methods grounded in Equation (3.2) is achieved by [82], which introduces a kernel-based (instead of an histogram-based) regression for the coefficients. Furthermore, they propose a method for the selection of the bandwidth of the kernel.

Recently, the use of orthogonal Legendre polynomials for approximating the functional form of the dynamical coefficients [110] was proposed. The weights of the polynomials are learned by minimizing the squared regression error that results after discretizing the SDE with the Euler-Maruyama scheme. Although this method is proposed as non-parametric we actually find that it is closer to a parametric method than to a non-parametric one, since the use of a small subset of any polynomial basis restricts the possible functional shapes of the estimates.

An alternative way for performing non-parametric regression that has become very popular among the machine learning community is based on the concept of Gaussian Process (GP) [111]. Instead of working in the weight-space that arises when using a set of basis functions (e.g., when using the Legendre polynomials), GPs permit working directly in the function space by placing a distribution over the functions. This enables a Bayesian treatment of the estimation process. The main advantage of this approach is that it yields probabilistic estimates, which permits the computation of robust confidence intervals. Furthermore, prior distributions modeling our prior beliefs about functions could also be included in the model. In this chapter, a GP-based method to reconstruct the SDE terms is proposed, with a focus on the computational challenges that common dataset sizes, $N \approx 10^3 - 10^5$, impose. A brief overview of the theory of GPs, as well as how they could be used for SDEs estimation is given in Section 3.1.

GPs have already been considered in the context of SDEs in the pioneering work of Ruttor et al. [117]. This work proposes a non-parametric estimation method of the drift function in systems of SDEs based on GP priors. It focuses on time series whose samples are separated by large time intervals. The authors develop an Expectation Maximization (EM) algorithm to

cope with the unobserved dynamics (i.e. latent) between observed samples. There are two main differences between [117] and our proposal: (1) we attempt to provide estimations in the case where we have a densely sampled time series (resulting in large series) whereas [117] focuses on the case of sparsely observed time series; and (2) we apply the GP approach to the estimation of both the drift and diffusion functions whereas [117] only deals with the drift.

The use of densely observed time series poses a challenge related with the demanding computations that GPs usually require. This is the main drawback that prevents GPs to be more widely utilized as non-parametric regression tool. In our proposal, we tackle the problem by providing a Sparse Gaussian Process (SGP) approximation (see [109] for an excellent overview of the subject), which is one of the main contributions of this chapter. The sparse approximation is developed in Section 3.2.

Section 3.3 details how to handle the mathematical difficulties that arise in the Sparse Gaussian Process (SGP) approximation due to the inclusion of the diffusion in the inference procedure. The estimation of non-constant diffusions has indeed become a major concern. Non-constant diffusions can be found in many physical systems (see, for example, [5, 40, 41, 52, 83, 91, 94]). Furthermore, it is well established that multiplicative noise can have surprising effects in the dynamics of the system. Some well-known examples of these effects are stochastic resonance [43], coherence resonance [107] and noise-induced transitions [65].

Section 3.4 discusses how to select the free parameters of the SGP and how to tune them for a better performance of the estimates. The resulting SGP method is validated in Sections 3.5 and 3.6-3.7 on simulated data and real data, respectively. In case of the simulated data, we compare our method with the kernel-based regression [82] and the polynomial method of [110]. In case of the real data, we apply our method to the study of financial data and climate transitions during the last glacial age. Finally, we discuss the main results of the chapter in Section 3.8.

It is worth noting that the main contributions of this chapter were published in [47]. Furthermore, with the aim of supporting reproducible research, we have also released the open-source `R` package `voila`. This package not only implements the SGP method, but also the methods proposed in [82] and [110]. A brief overview of `voila` is provided in Appendix B.

3.1 Gaussian Processes for SDE estimation

We consider the discrete-time signal $\boldsymbol{x}$ obtained from sampling the continuous Markov process $x(t)$. If the coefficients of the SDE are approximately constant over small time intervals $[t, t + \Delta t)$, the Euler-Maruyama discretization scheme yields [66, Chapter 2]

$$x(t + \Delta t) - x(t) \approx f(x(t))\Delta t + \sqrt{g(x(t))}(W(t + \Delta t) - W(t)),$$

which in discrete notation can be written as

$$\Delta x[i] = f_i \Delta t + \sqrt{g_i}(W[i + 1] - W[i]), \tag{3.3}$$

where we have denoted $\Delta x[i] = x[i + 1] - x[i]$, $f_i = f(x[i])$ and $g_i = g(x[i])$. Since the increments of the Wiener process $W[i + 1] - W[i]$ follow a Gaussian distribution $\mathcal{N}(0, \Delta t)$, Equation (3.3) can be used to approximate the discrete transition probabilities as follows:

$$p(x[i + 1] \mid x[i], f_i, g_i) = \frac{1}{\sqrt{2\pi g_i \Delta t}} \exp\left(-\frac{1}{2} \frac{(\Delta x[i] - f_i \Delta t)^2}{g_i \Delta t} \right).$$

Thus, when the stochastic process takes a value close to $x[i]$, it changes by an amount that is normally distributed, with expectation $f(x[i])\Delta t$ and variance $g(x[i])\Delta t$. Since the Wiener increments are independent between them, the log-likelihood of the path can be written as follows [66, Chapter 3]:

$$\log p(\boldsymbol{x} | f, g) = -\frac{1}{2} \sum_{i=1}^{N} \left[\frac{(\Delta x[i] - f_i \Delta t)^2}{g_i \Delta t} + \log\left(g_i\right) \right] - \frac{N}{2} \log\left(2\pi \Delta t\right) + \log p(x[1]). \tag{3.4}$$

It must be noted that this approximation is only valid if Δt is small. Concretely, since the Euler-Maruyama scheme has a strong order of convergence of $1/2$, the expected error between a real continuous path and the numerical approximation scales as $\Delta t^{1/2}$ [77]. Furthermore, we may only expect very accurate estimations for both f and g if the number of samples N is large. The time series' length used for SDE estimation depends on the study, but usual length requirements range from $N \approx 10^3$ [87, 110, 117] to $N \approx 10^5$ [60, 81, 108].

We would like to obtain estimates of f and g given a realization of the process $x(t)$ without any assumption on their form (non-parametric regression). GPs provide a powerful method for non-parametric regression and other machine learning tasks [111].

A GP is a collection of random variables indexed by some continuous set (e.g., time or space), which can be used to define a prior over a function $y(\boldsymbol{\xi})$, where $\boldsymbol{\xi}$ denotes a generic

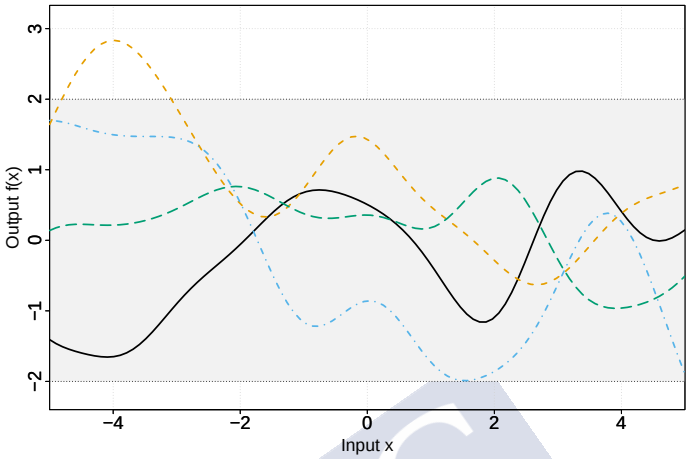

Figure 3.1: Four samples drawn from a GP distribution. The shaded region denotes the 95 % confidence interval at each input value x.

vector variable belonging to some multidimensional real space $\mathbb{R}^D$. A GP assumes that any finite number of function points $[y(\boldsymbol{\xi}_1), y(\boldsymbol{\xi}_2), \ldots, y(\boldsymbol{\xi}_n)]^T$ have a joint Gaussian distribution. Thus, $y(\boldsymbol{\xi}) \sim \mathcal{GP}(m(\boldsymbol{\xi}), k(\boldsymbol{\xi}, \boldsymbol{\xi}'))$ is fully specified by a mean function $m(\boldsymbol{\xi})$ and a covariance function $k(\boldsymbol{\xi}, \boldsymbol{\xi}')$, defined as follows:

$$m(\boldsymbol{\xi}) = \mathbb{E}[y(\boldsymbol{\xi})] \qquad \text{and} \qquad k(\boldsymbol{\xi}, \boldsymbol{\xi}') = \mathbb{E}\big[\big(y(\boldsymbol{\xi}) - m(\boldsymbol{\xi})\big)\big(y(\boldsymbol{\xi}') - m(\boldsymbol{\xi}')\big)\big].$$

By selecting a smooth covariance function $k(\boldsymbol{\xi}, \boldsymbol{\xi}')$ we can model smooth functions. Furthermore, the kernel determines almost all the properties of the resulting GP. For example, we can produce periodic processes by using a periodic kernel. Figure 3.1 illustrates the kind of functions that can be drawn from a GP distribution with a smooth kernel.

Since we are interested in non-parametric regression we shall avoid those kernels that impose any predetermined form on the final predictor, e.g., linear kernels or polynomial kernels. When using flexible kernels, GPs do not make strong assumptions about the nature of the function and, hence, they build their estimates from information derived from the data. Furthermore, even when lots of observations are used, there may still be some flexibility in the estimates. Thus, GPs are regarded as non-parametric methods [111, Chapter 1]. It should also be noted that the number of parameters of a GP model grows with the amount of training data, which is another feature of non-parametric methods [98, Section 1.4.1].

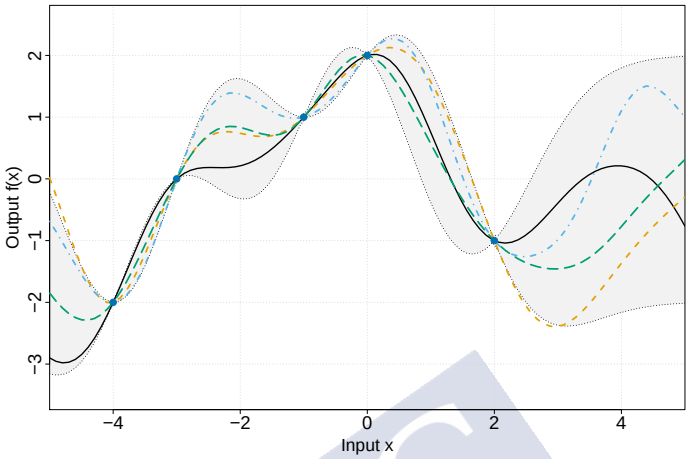

Figure 3.2: Four samples drawn from the posterior GP distribution, after several data points have been observed (dark blue dots). The shaded region denotes the 95 % confidence interval at each input value x.

In the SDE estimation problem, we shall use two different GPs for modeling our prior beliefs about the properties of the drift and diffusion terms. After observing the data $\boldsymbol{x}$, we shall update our knowledge about them. This updated knowledge is represented by the posterior distributions $p(\boldsymbol{f}^*|\boldsymbol{x})$ and $p(\boldsymbol{s}^*|\boldsymbol{x})$, where $\boldsymbol{f}^*$ and $\boldsymbol{s}^*$ represent the sets that result from evaluating $f(x)$ and $g(x)$ over a set of inputs, i.e., $\boldsymbol{f}^* = \{f(x) : x \in \boldsymbol{x}^*\}$ (a similar expression applies to $g(x)$). Figure 3.2 illustrates the result of updating the knowledge about the functions from Figure 3.1 after observing some data (that we consider as noiseless for simplicity). After observing the data, we would like to consider only those functions consistent with the acquired knowledge, i.e., we want to discard all the functions that do not pass through the observed points. Note that the uncertainty is lower near the observations. The combination of a prior GP distribution and the data results in a new posterior distribution over the functions. This intuitive inference procedure lies at the core of our approximation to the SDE estimation problem.

For modeling the drift function f, we shall use a GP with zero mean as prior. The zero mean arises from symmetry considerations and our lack of prior knowledge about f (there is no reason to assume positive values instead of negative ones, and vice versa). On the other hand, we must ensure that $g > 0$ (since it plays the role of a variance in Equation (3.4)). Thus,

we assume that $g(x) = \exp\big(s(x)\big)$, where $s(x)$ is a Gaussian process with a constant mean function $m(x) = v$. The v parameter is useful to control the scale of the noise process and possible numerical issues arising from the explosiveness of the exponential transformation. We shall use general covariance functions k_f and k_s for both processes, parametrized with the hyperparameters $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_s$, respectively. Hence, our complete model is

$$\log p(\boldsymbol{x}|f, s, v) \approx -\frac{1}{2}\sum_{i=1}^{N}\left[\frac{(\Delta x[i] - f_i\Delta t)^2}{\exp\big(s_i\big)\Delta t} + s_i\right] - \frac{N}{2}\log(2\pi\Delta t), \tag{3.5}$$
$$f(x)|\boldsymbol{\theta}_f \sim \mathcal{GP}\big(0, k_f(x, x', \boldsymbol{\theta}_f)\big),$$
$$s(x)|\boldsymbol{\theta}_s \sim \mathcal{GP}\big(v, k_s(x, x', \boldsymbol{\theta}_s)\big),$$

where we have ignored the distribution of $p(x[1])$ from Equation (3.4) (this is reasonable when $N >> 1$). Since $f(x)$ and $g(x)$ are GPs, the discrete vectors $\boldsymbol{f} = [f_1, f_2, \ldots, f_N]^T$ and $\boldsymbol{s} = [s_1, s_2, \ldots, s_N]^T$ must follow multivariate Gaussian distributions:

$$\boldsymbol{f} \mid \boldsymbol{\theta}_f \sim \mathcal{N}(\mathbf{0}^N, \boldsymbol{K}_{NN}) \qquad \boldsymbol{s} \mid \boldsymbol{\theta}_s \sim \mathcal{N}(\boldsymbol{v}^N, \boldsymbol{J}_{NN}),$$

where the entries of the covariance matrices are defined using

$$[\boldsymbol{K}_{NN}]_{ij} = k_f(x[i], x[j], \boldsymbol{\theta}_f), \qquad [\boldsymbol{J}_{NN}]_{ij} = k_s(x[i], x[j], \boldsymbol{\theta}_s), \tag{3.6}$$

and where $\mathbf{0}^N$ and $\boldsymbol{v}^N$ denote vectors of length N with all their entries set to 0 and v, respectively.

Although in Equations (3.5) and (3.6) we have explicitly written the dependencies on the hyperparameters (v, $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_s$) for the sake of completeness, we shall assume, for the moment, that they are known and fixed. Hence, we shall remove them from the equations in the next sections to keep the notation uncluttered.

As stated before, our aim is to compute the posteriors of any new set of new function points $\boldsymbol{f}^*$ (from $f(x)$) and $\boldsymbol{s}^*$ (from $s(x)$): $p(\boldsymbol{f}^*|\boldsymbol{x})$ and $p(\boldsymbol{s}^*|\boldsymbol{x})$. However, computing the posterior distribution of a model that involves GPs requires the calculation of inverse matrices, which usually scales as $\mathcal{O}(N^3)$ operations [111]. For large N, such as those used in the SDEs' literature, this approach is prohibitive. In the next section we discuss how to approximate the GP problem using only m points ($m << N$), which yields the so-called Sparse Gaussian Processes (SGP) [109, 111].

3.2 Approximation with Sparse Gaussian Processes

To overcome the intractable computations that a large dataset requires, many sparse methods construct an approximation to the GP using a small set of m inducing variables ($m << N$). Our inducing variables shall be the function points that result from evaluating $f(x)$ and $s(x)$ at m pseudo-inputs $\tilde{\boldsymbol{x}} = \{\tilde{x}_i\}_{i=1}^m$, i.e., $\tilde{\boldsymbol{f}} = \{f(x) : x \in \tilde{\boldsymbol{x}}\}$ and $\tilde{\boldsymbol{s}} = \{s(x) : x \in \tilde{\boldsymbol{x}}\}$. Note that, although we could have used a set of pseudo-inputs for $f(x)$ and another one for $s(x)$, we have opted for a single set for the sake of simplicity. The key idea is that, instead of using the N-dimensional posterior distribution $p(\boldsymbol{f} \mid \boldsymbol{x})$ to compute the predictions of the new function points $\boldsymbol{f}^*$, we could "summarize" the information that we may learn from the data about $f(x)$ in a m-dimensional distribution $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$, and then use it to make the predictions (a similar reasoning also applies to $s(x)$). Since $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{s}}$ represent the "reference points" that we shall use to make new predictions, it seems reasonable that $\tilde{\boldsymbol{x}}$ should be spread across the range of values of $\boldsymbol{x}$. Hence, in our problem, we shall require any pseudo-input to be contained in $[\min \boldsymbol{x}, \max \boldsymbol{x}]$. It must be noted that the pseudo-inputs $\tilde{\boldsymbol{x}}$ may be seen as hyperparameters of the complete model subject to optimization. However, for the moment we shall assume that they are known and fixed. To determine how the pseudo-inputs can be used to make predictions, we shall follow a similar approach to that introduced in [128], which used a variational formulation for learning the inducing variables of the SGP. The advantage of this approach is that variational inference naturally arises in our problem when trying to approximate the posterior distributions $p(\boldsymbol{f}|\boldsymbol{x})$ and $p(\boldsymbol{s}|\boldsymbol{x})$, as it will be discussed later.

Since $\tilde{\boldsymbol{f}}$ (or $\tilde{\boldsymbol{s}}$) and the N points that result from evaluating $f(x)$ (or $s(x)$) at $\{x[i]\}_{i=1}^N$ both sample the same GP, we assume:

$$\begin{bmatrix} \boldsymbol{f} \\ \tilde{\boldsymbol{f}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0}^N \\ \boldsymbol{0}^m \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{NN} & \boldsymbol{K}_{Nm} \\ \boldsymbol{K}_{mN} & \boldsymbol{K}_{mm} \end{bmatrix} \right),$$

$$\begin{bmatrix} \boldsymbol{s} \\ \tilde{\boldsymbol{s}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{v}^N \\ \boldsymbol{v}^m \end{bmatrix}, \begin{bmatrix} \boldsymbol{J}_{NN} & \boldsymbol{J}_{Nm} \\ \boldsymbol{J}_{mN} & \boldsymbol{J}_{mm} \end{bmatrix} \right),$$

where all the submatrices are computed analogously as done in Equation (3.6).

Since they will be used throughout the document, it is useful to write the expressions for the conditional distributions as follows:

$$\begin{aligned} \boldsymbol{f}|\tilde{\boldsymbol{f}} &\sim \mathcal{N}(\boldsymbol{A}\tilde{\boldsymbol{f}}, \boldsymbol{P}), \\ \boldsymbol{s}|\tilde{\boldsymbol{s}} &\sim \mathcal{N}(\boldsymbol{v}^N + \boldsymbol{B}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m), \boldsymbol{Q}), \end{aligned}$$

with

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{K}_{Nm}\boldsymbol{K}_{mm}^{-1}, & \boldsymbol{P} &= \boldsymbol{K}_{NN} - \boldsymbol{K}_{Nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mN}, \\ \boldsymbol{B} &= \boldsymbol{J}_{Nm}\boldsymbol{J}_{mm}^{-1}, & \boldsymbol{Q} &= \boldsymbol{J}_{NN} - \boldsymbol{J}_{Nm}\boldsymbol{J}_{mm}^{-1}\boldsymbol{J}_{mN}. \end{aligned} \tag{3.7}$$

Using this augmented model, the posterior distribution of the new function points $\boldsymbol{f}^*$, from $f(x)$, and $\boldsymbol{s}^*$, from $s(x)$, would be

$$\begin{aligned} p(\boldsymbol{f}^*, \boldsymbol{s}^* | \boldsymbol{x}) &= \int p(\boldsymbol{f}^*, \boldsymbol{s}^* | \boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}}) p(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}} | \boldsymbol{x}) d\boldsymbol{f}\, d\tilde{\boldsymbol{f}}\, d\boldsymbol{s}\, d\tilde{\boldsymbol{s}} \\ &= \int p(\boldsymbol{f}^* | \boldsymbol{f}, \tilde{\boldsymbol{f}}) p(\boldsymbol{s}^* | \boldsymbol{s}, \tilde{\boldsymbol{s}}) p(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}} | \boldsymbol{x}) d\boldsymbol{f}\, d\tilde{\boldsymbol{f}}\, d\boldsymbol{s}\, d\tilde{\boldsymbol{s}}, \end{aligned} \tag{3.8}$$

Following [128], we assume that $\tilde{\boldsymbol{f}}$ provides complete information for $\boldsymbol{f}^*$ in the sense that $p(\boldsymbol{f}^* | \tilde{\boldsymbol{f}}, \boldsymbol{f}) = p(\boldsymbol{f}^* | \tilde{\boldsymbol{f}})$. Similarly, we assume $p(\boldsymbol{s}^* | \tilde{\boldsymbol{s}}, \boldsymbol{s}) = p(\boldsymbol{s}^* | \tilde{\boldsymbol{s}})$. However, these assumptions do not prevent the computation of the GPs' posterior $p(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}} | \boldsymbol{x})$. To make the model computationally efficient we shall approximate this distribution by factorizing it in groups of $(\boldsymbol{f}, \tilde{\boldsymbol{f}})$ and $(\boldsymbol{s}, \tilde{\boldsymbol{s}})$, as it is usually done in the variational inference approach [11, 128]:

$$p(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}} | \boldsymbol{x}) \approx q(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}}) = p(\boldsymbol{f} | \tilde{\boldsymbol{f}}) q_{\tilde{f}}(\tilde{\boldsymbol{f}}) p(\boldsymbol{s} | \tilde{\boldsymbol{s}}) q_{\tilde{s}}(\tilde{\boldsymbol{s}}), \tag{3.9}$$

where $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$ and $q_{\tilde{s}}(\tilde{\boldsymbol{s}})$ denote unconstrained variational distributions over $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{s}}$. Under this assumption, Equation (3.8) becomes

$$\begin{aligned} p(\boldsymbol{f}^* \mid \boldsymbol{x}) &\approx \int p(\boldsymbol{f}^* \mid \tilde{\boldsymbol{f}}) q_{\tilde{f}}(\tilde{\boldsymbol{f}}) d\tilde{\boldsymbol{f}}, \\ p(\boldsymbol{s}^* \mid \boldsymbol{x}) &\approx \int p(\boldsymbol{s}^* \mid \tilde{\boldsymbol{s}}) q_{\tilde{s}}(\tilde{\boldsymbol{s}}) d\tilde{\boldsymbol{s}}. \end{aligned}$$

Given Equation (3.9), we may try to minimize the following Kullback-Leibler divergence to calculate the $q(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}})$ distribution:

$$\mathcal{KL}(q \mid p) = \int q(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}}) \log \frac{q(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}})}{p(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}} | \boldsymbol{x})} d\boldsymbol{f}\, d\tilde{\boldsymbol{f}}\, d\boldsymbol{s}\, d\tilde{\boldsymbol{s}}.$$

Taking into account the identity

$$\log p(\boldsymbol{x}) = \mathcal{L}(q) + \mathcal{KL}(q \mid p),$$

where we have defined

$$\mathcal{L}(q) = \int q(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}}) \log \frac{p(\boldsymbol{x}, \boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}})}{q(\boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{s}, \tilde{\boldsymbol{s}})} d\boldsymbol{f}\, d\tilde{\boldsymbol{f}}\, d\boldsymbol{s}\, d\tilde{\boldsymbol{s}},$$

we notice that minimizing the Kullback-Leibler divergence with respect to q is equivalent to maximizing the lower bound of the marginal log-likelihood $\mathcal{L}(q)$. Setting $\partial\mathcal{L}(q)/\partial q = 0$ we obtain the optimal solutions [11]:

$$\log q_{\tilde{f}}(\tilde{\boldsymbol{f}}) = \mathbb{E}_{q_{\text{-}\tilde{f}}}\left[\log\left(p(\boldsymbol{x}|\boldsymbol{f},\boldsymbol{s})p(\tilde{\boldsymbol{s}})p(\tilde{\boldsymbol{f}})\right)\right], \tag{3.10a}$$

$$\log q_{\tilde{s}}(\tilde{\boldsymbol{s}}) = \mathbb{E}_{q_{\text{-}\tilde{s}}}\left[\log\left(p(\boldsymbol{x}|\boldsymbol{f},\boldsymbol{s})p(\tilde{\boldsymbol{s}})p(\tilde{\boldsymbol{f}})\right)\right], \tag{3.10b}$$

where we have denoted:

$$q_{\text{-}\tilde{f}}(\boldsymbol{f},\boldsymbol{s},\tilde{\boldsymbol{s}}) = p(\boldsymbol{f}|\tilde{\boldsymbol{f}})p(\boldsymbol{s}|\tilde{\boldsymbol{s}})q_{\tilde{s}}(\tilde{\boldsymbol{s}}),$$
$$q_{\text{-}\tilde{s}}(\boldsymbol{f},\tilde{\boldsymbol{f}},\boldsymbol{s}) = p(\boldsymbol{f}|\tilde{\boldsymbol{f}})q_{\tilde{f}}(\tilde{\boldsymbol{f}})p(\boldsymbol{s}|\tilde{\boldsymbol{s}}),$$

to the density functions that result from ignoring the distributions $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$ and $q_{\tilde{s}}(\tilde{\boldsymbol{s}})$ from $q(\boldsymbol{f},\tilde{\boldsymbol{f}},\boldsymbol{s},\tilde{\boldsymbol{s}})$ (see Equation (3.9)), respectively.

Note that Equations (3.10) are not a closed-form solution of the variational inference problem, since both equations are coupled. However, they naturally suggest the use of a coordinate ascent algorithm to find a solution. The coordinate ascent algorithm iterates between holding $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$ to update $q_{\tilde{s}}(\tilde{\boldsymbol{s}})$ using Equation (3.10a) and holding $q_{\tilde{s}}(\tilde{\boldsymbol{s}})$ to update $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$ through Equation (3.10b).

To find the variational distribution for the drift, we start by expanding the expression inside the expectation operator from Equation (3.10a):

$$\begin{aligned}\log q_{\tilde{f}}(\tilde{\boldsymbol{f}}) =& \mathbb{E}_{q_{\text{-}\tilde{f}}}\left[\log\left(p(\boldsymbol{x}|\boldsymbol{f},\boldsymbol{s})p(\tilde{\boldsymbol{s}})p(\tilde{\boldsymbol{f}})\right)\right] \\ =& \mathbb{E}_{q_{\text{-}\tilde{f}}}\left[\log p(\boldsymbol{x}|\boldsymbol{f},\boldsymbol{s})\right] + \mathbb{E}_{q_{\text{-}\tilde{f}}}\left[\log p(\tilde{\boldsymbol{s}})\right] + \mathbb{E}_{q_{\text{-}\tilde{f}}}\left[\log p(\tilde{\boldsymbol{f}})\right].\end{aligned}$$

Given that the expectation operator does not affect $\log p(\tilde{\boldsymbol{f}})$ and that when applied to $\log p(\tilde{\boldsymbol{s}})$ results in an expression that does not depend on $\tilde{\boldsymbol{f}}$, we may write:

$$\log q_{\tilde{f}}(\tilde{\boldsymbol{f}}) = \mathbb{E}_{q_{\text{-}\tilde{f}}}\left[\log p(\boldsymbol{x}|\boldsymbol{f},\boldsymbol{s})\right] - \frac{1}{2}\tilde{\boldsymbol{f}}^T\boldsymbol{K}_{mm}^{-1}\tilde{\boldsymbol{f}} + \text{constant}, \tag{3.11}$$

where we have denoted all terms that do not depend on $\tilde{\boldsymbol{f}}$ as *constant*. It is convenient to work with the term *constant*, given that we can infer its value after identifying the distribution of $q_{\tilde{f}}$. In that case, *constant* corresponds to the normalizing constant required by the distribution $q_{\tilde{f}}$.

Expanding the expression inside the expectation operator from Equation (3.11) and joining the new constants yields

$$\begin{aligned}\log q_{\tilde{f}}(\tilde{\boldsymbol{f}}) = &-\frac{1}{2\Delta t}\sum_{i=1}^{N} \mathbb{E}_{q_{\tilde{s}}(\tilde{\boldsymbol{s}})p(\boldsymbol{s}|\tilde{\boldsymbol{s}})}\left[\exp(-s_i)\right]\mathbb{E}_{p(\boldsymbol{f}|\tilde{\boldsymbol{f}})}\left[(\Delta x_i - \Delta t f_i)^2\right] \\ &-\frac{1}{2}\tilde{\boldsymbol{f}}^T\boldsymbol{K}_{mm}^{-1}\tilde{\boldsymbol{f}} + \text{constant}.\end{aligned} \tag{3.12}$$

Using Fubini's rule of integration we may write the first expectation from Equation (3.12) as follows:

$$\mathbb{E}_{q_{\tilde{s}}(\tilde{\boldsymbol{s}})p(\boldsymbol{s}|\tilde{\boldsymbol{s}})}\left[\exp(-s_i)\right] = \mathbb{E}_{q_{\tilde{s}}}\left[\mathbb{E}_{p(\boldsymbol{s}|\tilde{\boldsymbol{s}})}\left[\exp(-s_i)\right]\right]. \tag{3.13}$$

It is possible to demonstrate that if $X \sim \mathcal{N}(\mu, \sigma^2)$, then $\mathbb{E}\left[\exp(-X)\right] = \exp(-\mu + \sigma^2/2)$. Hence, Equation (3.13) becomes

$$\mathbb{E}_{q_{\tilde{s}}}\left[\mathbb{E}_{p(\boldsymbol{s}|\tilde{\boldsymbol{s}})}\left[\exp(-s_i)\right]\right] = \mathbb{E}_{q_{\tilde{s}}}\left[\exp\left(-[\boldsymbol{v}^N + \boldsymbol{B}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)]_i + \frac{Q_{ii}}{2}\right)\right] = \zeta_i, \tag{3.14}$$

where we have introduced the new variable ζ_i and where we have used the definitions of $\mathbf{B}$ and $\mathbf{Q}$ from Equation (3.7). Introducing back Equations (3.13) and (3.14) into Equation (3.12) we finally arrive to

$$\begin{aligned}\log q_{\tilde{f}}(\tilde{\boldsymbol{f}}) &= -\frac{1}{2\Delta t}\sum_{i=1}^{N}\zeta_i \cdot \mathbb{E}_{p(\boldsymbol{f}|\tilde{\boldsymbol{f}})}\left[(\Delta x_i)^2 - 2\Delta t \Delta x_i f_i - (\Delta t)^2 f_i^2\right] \\ &\quad -\frac{1}{2}\tilde{\boldsymbol{f}}^T\boldsymbol{K}_{mm}^{-1}\tilde{\boldsymbol{f}} + \text{constant} \\ &= -\frac{1}{2\Delta t}\sum_{i=1}^{N}\zeta_i \cdot \left[(\Delta x_i)^2 - 2\Delta t \Delta x_i [\boldsymbol{A}\tilde{\boldsymbol{f}}]_i - (\Delta t)^2([\boldsymbol{A}\tilde{\boldsymbol{f}}]_i^2 + P_{ii})\right] \\ &\quad -\frac{1}{2}\tilde{\boldsymbol{f}}^T\boldsymbol{K}_{mm}^{-1}\tilde{\boldsymbol{f}} + \text{constant}\end{aligned} \tag{3.15}$$

Reordering Equation (3.15) and expressing it in vector form, we finally arrive to

$$\log q_{\tilde{f}}(\tilde{\boldsymbol{f}}) = -\frac{1}{2}\tilde{\boldsymbol{f}}^T\left[\boldsymbol{K}_{mm}^{-1} + \Delta t \boldsymbol{A}^T \text{diag}(\boldsymbol{\zeta})\boldsymbol{A}\right]\tilde{\boldsymbol{f}} + [\boldsymbol{\zeta} \odot \boldsymbol{\Delta x}]^T\boldsymbol{A}\tilde{\boldsymbol{f}} + \text{constant}, \tag{3.16}$$

where $\odot$ denotes the element-by-element multiplication of two vectors and $\text{diag}(\boldsymbol{\zeta})$ is the diagonal matrix constructed using the values of the vector $\boldsymbol{\zeta}$ as main diagonal.

Equation (3.16) implies that $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$ follows a Gaussian distribution, that we shall write as follows:

$$\begin{aligned} &q_{\tilde{f}}(\tilde{\boldsymbol{f}}) = \mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{F}), \text{ where} \\ &\boldsymbol{F} = \left[\boldsymbol{K}_{mm}^{-1} + \Delta t \boldsymbol{A}^T \text{diag}(\boldsymbol{\zeta})\boldsymbol{A}\right]^{-1}, \\ &\boldsymbol{\mu}_f = \boldsymbol{F}\boldsymbol{A}^T(\boldsymbol{\zeta} \odot \boldsymbol{\Delta x}). \end{aligned} \tag{3.17}$$

On the other hand, to find the variational distribution for the diffusion we start from Equation (3.10b). Proceeding similarly as before it is possible to arrive to

$$\begin{aligned} \log q_{\tilde{s}}(\tilde{\boldsymbol{s}}) = &-\frac{1}{2}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)^T \boldsymbol{J}_{mm}^{-1}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m) \\ &- \frac{1}{2\Delta t}\sum_{i=1}^{N} \mathbb{E}_{p(\boldsymbol{s}|\tilde{\boldsymbol{s}})}\left[\exp(-s_i)\right] \mathbb{E}_{q_{\tilde{f}}(\tilde{\boldsymbol{f}})p(\boldsymbol{f}|\tilde{\boldsymbol{f}})}\left[(\Delta x_i - \Delta t f_i)^2\right] \\ &- \frac{1}{2}\sum_{i=1}^{N} \mathbb{E}_{p(\boldsymbol{s}|\tilde{\boldsymbol{s}})}\left[s_i\right] + \text{constant}. \end{aligned}$$

The expectation of $\exp(-s_i)$ can be computed as in Equation (3.13), which results in

$$\begin{aligned} \log &q_{\tilde{s}}(\tilde{\boldsymbol{s}}) = \\ &-\frac{1}{2}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)^T \boldsymbol{J}_{mm}^{-1}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m) \\ &- \frac{1}{2\Delta t}\sum_{i=1}^{N} \exp\left(-[\boldsymbol{v}^N + \boldsymbol{B}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)]_i + \frac{Q_{ii}}{2}\right) \mathbb{E}_{q_{\tilde{f}}(\tilde{\boldsymbol{f}})p(\boldsymbol{f}|\tilde{\boldsymbol{f}})}\left[(\Delta x_i - \Delta t f_i)^2\right] \\ &- \frac{1}{2}\sum_{i=1}^{N}[\boldsymbol{B}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)]_i + \text{constant}. \end{aligned} \tag{3.18}$$

Using again Fubini's law, we may write:

$$\begin{aligned} \mathbb{E}_{q_{\tilde{f}}(\tilde{\boldsymbol{f}})p(\boldsymbol{f}|\tilde{\boldsymbol{f}})}\left[(\Delta x_i - \Delta t f_i)^2\right] &= \mathbb{E}_{q_{\tilde{f}}}\left[\mathbb{E}_{p(\boldsymbol{f}|\tilde{\boldsymbol{f}})}\left[(\Delta x_i - \Delta t f_i)^2\right]\right] \\ &= \mathbb{E}_{q_{\tilde{f}}}\left[\Delta x_i^2 - 2\Delta t \Delta x_i [\boldsymbol{A}\tilde{\boldsymbol{f}}]_i + (\Delta t)^2\left([\boldsymbol{A}\tilde{\boldsymbol{f}}]_i^2 + P_{ii}\right)\right] \\ &= \psi_i, \end{aligned}$$

where we have introduced the definition of the new variable ψ_i. Introducing ψ_i into Equa-

tion (3.18), we finally arrive to

$$\begin{aligned}\log q_{\tilde{s}}(\tilde{\boldsymbol{s}}) = &-\frac{1}{2\Delta t}\sum_{i=1}^{N}\psi_i \exp\left(-[\boldsymbol{v}^N + \boldsymbol{B}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)]_i + \frac{Q_{ii}}{2}\right)\\ &-\frac{1}{2}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)^T \boldsymbol{J}_{mm}^{-1}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m) - \frac{1}{2}\sum_{i=1}^{N}[\boldsymbol{B}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)]_i + \text{constant}.\end{aligned} \tag{3.19}$$

Unlike the distribution for $q_{\tilde{f}}(\tilde{\boldsymbol{f}})$, we cannot identify the distribution that appears in Equation (3.19). This is not surprising, since the updates in a variational inference problem are only available in closed-form when using conditionally conjugate distributions. As a consequence, we are forced to use approximate variational inference.

3.3 Laplace Variational Inference for the Estimation of the Diffusion

Laplace approximations use a Gaussian to approximate intractable density functions. In the context of variational inference, it has already been considered in [134], in order to handle non-conjugate models. We shall use this approach to handle Equation (3.19). Let $\hat{\boldsymbol{s}}_m$ be the maximum of the right hand side from Equation (3.19), which may be found using numerical optimization techniques. In our implementation of the method, we have used the L-BFGS-B algorithm [17], although any other method could have been used. A Taylor expansion around $\hat{\boldsymbol{s}}_m$ gives

$$\log q_{\tilde{s}}(\tilde{\boldsymbol{s}}) \approx \frac{1}{2}(\tilde{\boldsymbol{s}} - \hat{\boldsymbol{s}}_m)^T \boldsymbol{H}_{\log q}(\hat{\boldsymbol{s}}_m)(\tilde{\boldsymbol{s}} - \hat{\boldsymbol{s}}_m) + \text{constant},$$

where $\boldsymbol{H}_{\log q}(\hat{\boldsymbol{s}}_m)$ is the Hessian matrix of $\log q_{\tilde{s}}(\tilde{\boldsymbol{s}})$ evaluated at $\hat{\boldsymbol{s}}_m$. In our case, we find that

$$[\boldsymbol{H}_{\log q}(\hat{\boldsymbol{s}}_m)]_{kq} = -\frac{1}{2\Delta t}\sum_{i=1}^{N}\psi_i \cdot \exp\left(-v + \frac{Q_{ii}}{2}\right) B_{ik}B_{iq}\exp\left(-[\boldsymbol{B}(\hat{\boldsymbol{s}}_m - \boldsymbol{v}^m)]_i\right) - [\boldsymbol{J}_{mm}^{-1}]_{kq}.$$

Thus, the approximate update for $q_{\tilde{s}}(\tilde{\boldsymbol{s}})$ to be used in the coordinate ascent algorithm is a Gaussian distribution:

$$\begin{aligned}&q_{\tilde{s}}(\tilde{\boldsymbol{s}}) \approx \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{S}), \text{ where}\\ &\boldsymbol{\mu}_s = \hat{\boldsymbol{s}}_m, \qquad \boldsymbol{S} = -[\boldsymbol{H}_{\log q}(\hat{\boldsymbol{s}}_m)]^{-1}.\end{aligned} \tag{3.20}$$

Taking into account that both the distributions of $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{s}}$ are Gaussians, we can write ψ_i and ζ_i as follows:

$$\begin{aligned}\zeta_i &= \exp\left[-[v + \boldsymbol{B}(\boldsymbol{\mu}_s - \boldsymbol{v}^m)]_i + \frac{1}{2}(Q_{ii} + \boldsymbol{B}_{i,\cdot}^T \boldsymbol{S} \boldsymbol{B}_{i,\cdot})\right],\\ \psi_i &= (\Delta x[i])^2 - 2\Delta t \Delta x[i][\boldsymbol{A}\boldsymbol{\mu}_f]_i + (\Delta t)^2[\boldsymbol{P} + \boldsymbol{A}(\boldsymbol{\mu}_f \boldsymbol{\mu}_f^T + \boldsymbol{F})\boldsymbol{A}^T]_{ii},\end{aligned}$$

where $\boldsymbol{B}_{i,\cdot}$ denotes the i-th row of the $\boldsymbol{B}$ matrix. It must be noted that the values of $\boldsymbol{\psi}$ and $\boldsymbol{\zeta}$ should be updated with each step of the coordinate ascent algorithm. After each step, the convergence of the algorithm must be assessed by computing the lower bound $\mathcal{L}(q)$:

$$\begin{aligned}\mathcal{L}(q) =& \mathbb{E}_q\left[\log p(\boldsymbol{x}|f,s)\right] + \mathbb{E}_{q_{\tilde{f}}}\left[\log p(\tilde{\boldsymbol{f}})\right] + \mathbb{E}_{q_{\tilde{s}}}\left[\log p(\tilde{\boldsymbol{s}})\right]\\ &- \mathbb{E}_{q_{\tilde{f}}}\left[\log q_{\tilde{f}}(\tilde{\boldsymbol{f}})\right] - \mathbb{E}_{q_{\tilde{s}}}\left[\log q_{\tilde{s}}(\tilde{\boldsymbol{s}})\right]\\ =& -\frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_q\left[\frac{(\Delta x[i] - \Delta t f_i)^2}{\Delta t \exp(s_i)} + s_i\right] - \frac{N}{2}\log(2\pi\Delta t)\\ &- \frac{1}{2}\mathbb{E}_{q_{\tilde{f}}}\left[\tilde{\boldsymbol{f}}^T \boldsymbol{K}_{mm}^{-1}\tilde{\boldsymbol{f}}\right] - \frac{1}{2}\log|\boldsymbol{K}_{mm}| - \frac{m}{2}\log 2\pi\\ &- \frac{1}{2}\mathbb{E}_{q_{\tilde{s}}}\left[(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)^T \boldsymbol{J}_{mm}^{-1}(\tilde{\boldsymbol{s}} - \boldsymbol{v}^m)\right] - \frac{1}{2}\log|\boldsymbol{J}_{mm}| - \frac{m}{2}\log 2\pi\\ &+ \mathbb{H}_{q_{\tilde{f}}}\left[\tilde{\boldsymbol{f}}\right] + \mathbb{H}_{q_{\tilde{s}}}\left[\tilde{\boldsymbol{s}}\right],\end{aligned} \tag{3.21}$$

where $\mathbb{H}$ is the entropy of a distribution. Taking the expectations in Equation (3.21) yields

$$\begin{aligned}\mathcal{L}(q) =& -\frac{1}{2\Delta t}\sum_{i=1}^{N}\psi_i\zeta_i - \frac{1}{2}\sum_{i=1}^{N}[\boldsymbol{v}^N + \boldsymbol{B}(\boldsymbol{\mu}_s - \boldsymbol{v}^m)]_i - \frac{N}{2}\log(2\pi\Delta t)\\ &- \frac{1}{2}\left[\mathrm{tr}(\boldsymbol{K}_{mm}^{-1}\boldsymbol{F}) + \boldsymbol{\mu}_f^T \boldsymbol{K}_{mm}^{-1}\boldsymbol{\mu}_f\right] - \frac{1}{2}\log|\boldsymbol{K}_{mm}| - \frac{m}{2}\log 2\pi\\ &- \frac{1}{2}\left[\mathrm{tr}(\boldsymbol{J}_{mm}^{-1}\boldsymbol{S}) + (\boldsymbol{\mu}_s - \boldsymbol{v}^m)^T \boldsymbol{J}_{mm}^{-1}(\boldsymbol{\mu}_s - \boldsymbol{v}^m)\right] - \frac{1}{2}\log|\boldsymbol{J}_{mm}| - \frac{m}{2}\log 2\pi\\ &+ \frac{1}{2}\log\left((2\pi e)^m|\boldsymbol{F}|\right) + \frac{1}{2}\log\left((2\pi e)^m|\boldsymbol{S}|\right) + \text{constant},\end{aligned} \tag{3.22}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. In addition to checking the convergence, computing the lower bound permits checking the correctness of the implementation since it should always increase monotonically; and since it is an approximation to the marginal likelihood, it can be used for Bayesian model selection. For example, we can use the lower bound to select the best kernel among a set of possible ones or to select the number of inducing points m. However, there is a subtle detail that must be addressed. Although the variational inference

framework approximates the posterior distribution, it only does it around one of the local modes. With m pseudo-inputs, there are $m!$ equivalent modes due to the lack of identifiability of the pseudo-inputs (the different modes only differ through a relabeling of the $\tilde{\boldsymbol{x}}$ vector). A simple approximate solution that takes into account the multi-modality is using

$$\mathcal{L}' \approx \mathcal{L} + \log(m!), \tag{3.23}$$

for model selection [11].

3.4 Hyperparameter Optimization

So far, we have assumed that the "variance" parameter v, the hyperparameters of the covariance functions, $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_s$, and the pseudo-inputs $\tilde{\boldsymbol{x}}$ were known and fixed. However, Equation (3.22) does depend on all these hyperparameters, i.e., $\mathcal{L}(q) \triangleq \mathcal{L}(q, v, \boldsymbol{\theta}_f, \boldsymbol{\theta}_s, \tilde{\boldsymbol{x}}) \triangleq \mathcal{L}(q, \boldsymbol{\theta}_{\text{all}})$, with $\boldsymbol{\theta}_{\text{all}} = [v, \boldsymbol{\theta}_f, \boldsymbol{\theta}_s, \tilde{\boldsymbol{x}}]^T$. Hence, further maximization of the lower bound could be achieved. Note that this optimization permits the automatic selection of the inducing-inputs $\tilde{\boldsymbol{x}}$ and the kernel hyperparameters starting from some reasonable initial values. In our implementation, we have interleaved the updates of the variational distributions with the numerical optimization of the lower bound with respect to the hyperparameters (since the analytical optimization is intractable). This permits the slow adaptation of the hyperparameters to the variational distributions. The resulting algorithm may be compared with a Generalized Expectation Maximization (GEM) algorithm [93]. In what we may identify as the E step, the variational distributions are updated. First, the distribution parameters $\boldsymbol{\mu}_f$ and $\boldsymbol{F}$ are modified according to Equation (3.17) using the last values obtained for $\boldsymbol{\mu}_s$ and $\boldsymbol{S}$ to compute any expectation involving the random variable $\tilde{\boldsymbol{s}}$. These new values are then used to compute the expectations involving $\tilde{\boldsymbol{f}}$ and updating $\boldsymbol{\mu}_s$ and $\boldsymbol{S}$ through Equation (3.20). In the M step, the lower bound given by Equation (3.22) is further optimized with respect to the hyperparameters while keeping the distribution parameters ($\boldsymbol{\mu}_f, \boldsymbol{F}, \boldsymbol{\mu}_s$, and $\boldsymbol{S}$) fixed. Given that finding a maximum may have a slow convergence, instead of aiming to maximize the lower bound we sought to change the hyperparameters in a way that increases: $\mathcal{L}(q, \boldsymbol{\theta}_{\text{all}}^{n+1}) > \mathcal{L}(q, \boldsymbol{\theta}_{\text{all}}^{n})$. This may be interpreted as a "partial" M step. In our implementation, we just limited the number of iterations of a L-BFGS-B algorithm [17], although any other numerical method could have been used. Changing from the maximization of the objective to simply searching for an increase of it is what makes our method similar to the GEM algorithm instead of the standard

EM algorithm. The E and M steps are then repeated until the convergence of the lower bound $\mathcal{L}$.

3.4.1 Hyperparameter Initialization and Kernel Selection

The lower bound in a variational problem is usually a non-convex function and hence, the proposed GEM-like algorithm is only guaranteed to converge to a local maximum, which can be sensitive to initialization [12]. Thus, several trials with randomly selected initial values of the hyperparameters should be run. The final estimate can be selected using Equation (3.23). However, it should be noted that, experimentally, solutions stacked in a clearly suboptimal local maxima happen infrequently.

Given that SGPs provide a Bayesian framework, the kernels and the initial values for their hyperparameters should be selected to model the prior beliefs about the behavior of the drift and diffusion functions. Choosing a proper kernel requires some knowledge about the properties of covariance functions [111, Chapter 4] and experience to combine them to model functions with different kinds of structure [34, Chapter 2]. Reasonable choices commonly used in the GP literature when no prior information is available are the squared exponential kernel (or Gaussian kernel) and the rational quadratic kernel [111], although any kernel could be used within our method. The squared exponential kernel is one of the most widely used covariance functions in the field of GP regression since it is infinitely differentiable and hence it yields very smooth processes [111, Chapter 2]. Its main hyperparameter is the length-scale l, i.e., the variation necessary in the input variable for the function values to appreciably change. On the other hand, the rational quadratic kernel can be seen as an infinite sum of squared exponential covariance functions with different length-scales. It has two main hyperparameters, a mean length-scale l and a parameter controlling the mixing of the different squared exponential kernels (derived from a Gamma distribution) [111, Chapter 4].

The amplitude of a kernel function can be interpreted as the prior belief about the variance of the drift/diffusion term. Hence, large amplitudes can be used when no prior information is available. The selection of the amplitude hyperparameter for the diffusion requires further discussion since we have to link the amplitude of the kernel modeling $s(x)$, A_s, with our prior belief about the variance of $g(x) = \exp\big(s(x)\big)$, A_g. Furthermore, it also requires selecting an initial value for v. Since a lognormal random variable $Z \sim \log\mathcal{N}(\mu = v, \sigma^2 = A_s)$ fulfills

$$\mathbb{E}[Z] = e^{v+\frac{A_s}{2}}, \qquad \mathrm{Var}[Z] = (e^{A_s} - 1)e^{(2v+A_s)},$$

we find the proper parameters v and A_s from the prior belief A_g and the data itself, $\boldsymbol{x}$, using

$$\begin{aligned} A_s &= \log\left(1 + \frac{A_g}{(\mathrm{Var}\left[\Delta\boldsymbol{x}\right]/\Delta t)^2}\right), \\ v &= \log\left(\frac{\mathrm{Var}\left[\Delta\boldsymbol{x}\right]}{\Delta t}\right) - \frac{A_s}{2}. \end{aligned} \tag{3.24}$$

As argued in Section 3.2, $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{s}}$ may be interpreted as "reference points" used to infer the shape of $f(x)$ and $s(x)$. Hence, we may expect $\tilde{\boldsymbol{x}}$ to be spread across the range of values of $\boldsymbol{x}$ so that the function shapes can be properly modeled in the whole range of x. It is also reasonable to assume that the inducing points should be more concentrated in those regions where $f(x)$ or $s(x)$ change their curvature. However, in our non-parametric approach, we cannot presume any prior knowledge about these regions. Thus, a simply strategy for selecting the initial values of the pseudo-inputs would be to uniformly spread $\tilde{\boldsymbol{x}}$ between $\min \boldsymbol{x}$ and $\max \boldsymbol{x}$. It is possible to design another approach based on the inducing points concentrating in the regions with low uncertainty about the function shape. This is due to the fact that the inducing points permit reducing the variance around their "region of influence", which enables accurately modeling the low-uncertainty true posterior and hence reducing the Kullback-Leibler divergence. Further evidence about this will be given in Section 3.5. Thus, we propose to initialize the inducing points with the result of applying the quantile function to the values $\{0/(m-1), 1/(m-1), ..., (m-1)/(m-1)\}$, since this approach concentrates the inducing points in the region where more evidence for inferring confident estimates is available. We will later refer to this approach as the "percentile initialization". It must be noted that, when performing several runs of the estimation algorithm, random noise can be added to each value of $\tilde{\boldsymbol{x}}$ to obtain slightly different starting points. In practical applications, we also add the restriction that, after adding the noise, the $\tilde{\boldsymbol{x}}$ vector should remain ordered and that $\min \tilde{\boldsymbol{x}} \geq \min \boldsymbol{x}$ and $\max \tilde{\boldsymbol{x}} \leq \max \boldsymbol{x}$.

The selection of the number of inducing points m is the most challenging, since SGP usually get better approximations to the full GP posterior when using more points (larger $\mathcal{L}'$), at the cost of greater computational time [111]. When taking into account both factors, there is not an unique way of defining which is the optimum value of m and hence, the final choice can be subjective. Rasmussen et al. suggest to perform runs with small values of m and compare the resulting estimates between them while getting a feeling on how the running time scales [111]. Since most kernels use a length-scale parameter l we suggest using

$$m = \lfloor (\max \boldsymbol{x} - \min \boldsymbol{x})/l \rfloor \tag{3.25}$$

Table 3.1: Models used for the validation with synthetic data.

Model	$f(x)$	$\sqrt{g(x)}$
M_1	$-(x-3)$	$\sqrt{2}$
M_2	$-(x^3-x)$	1
M_3	$-x^3$	$0.2+x^2$
M_4	$-0.7(x-0.5)$	$\sqrt{0.7x(1-x)}$
M_5	$-(x-0.225)$	$0.5\sqrt{x}$
M_6	$-x+\sin(3.5x)\exp(-x^2)$	0.431

as a rule of thumb for obtaining an estimate of a proper number of inducing points. This rule uses only a few inducing points when the function varies very smoothly (large l) and a large number of them when the function wiggles quickly (small l).

3.5 Validation on synthetic data

To assess the validity of the SGP method, we shall compare its performance with the kernel-based method [82] and with a version of the orthonormal polynomials method [110] using a set of simulated SDEs. From now on, we shall refer to these methods as the Kernel Based Regression (KBR) and the POLY method (since it is based on orthonormal polynomials), respectively. We have also used these tests to further investigate the impact of the number of pseudo-inputs m on the estimates.

For the validation, we consider the generic SDE described by Equation (3.1) parametrized with the drift and diffusion functions summarized in Table 3.1. It must be noted that some of these tests have been inspired by some well-known models. M_1 is the celebrated Ornstein-Uhlenbeck model, which describes the motion of a Brownian particle in velocity space [84]. M_4 is the Jacobi diffusion process, which has an invariant distribution that is uniform on $(0, 1)$ [66]. A Jacobi-based model was used in [85] to model exchange rates in target zone. M_5 is the Cox-Ingersoll-Ross model. Despite it was introduced to model population growth, it has become popular after its proposal for studying short-term interest rates in finance [23]. Although M_2 and M_6 do not receive any particular name, they are interesting models since they are able to generate time series with a bimodal density. Finally, M_3 was used to test dynamical systems with nonlinear drift and diffusion functions and just a single stable point.

For each of these models, 100 time series with a length of 10^4 samples were generated. The

Euler-Maruyama scheme with an integration step $\Delta t = 0.001$ was used for the simulations. The quality of the estimations obtained for the i-th simulation of Model M_j was assessed by the weighted integrated absolute error:

$$\mathcal{E}(M_j, i) = \int_{-\infty}^{\infty} |F_j(x) - \hat{F}_j(x)| \cdot p_{ij}(x) dx,$$

where F_j can be either f_j or g_j, $\hat{F}_j$ denotes its estimate and $p_{ij}(x)$ is the probability density function of the i-th simulation of the M_j model. In practice, $p_{ij}(x)$ is approximated using a kernel density estimate with a Gaussian kernel. The bandwidth of the kernel is selected using Silverman's rule of thumb [121, Page 48, Equation 3.31].

To select a proper bandwidth for the KBR method, the selection algorithm described in [82] was implemented. Regarding the POLY method, the parameter estimation was performed with polynomials of orders $R = 1, 2, \ldots, 5$ and $L = 0, 1, \ldots, 3$ for the drift and the diffusion terms, respectively. Instead of using the Legendre polynomials as in [110], the orthonormal polynomials described in [73] were employed for easiness of implementation. Our tests indicate that the use of these polynomials instead of the Legendre polynomials do not undermine the expressive power of the method. Three different model selection methods were tested within the POLY framework: the simulation-based method proposed in [110], a cross-validation method and a stepwise regression method. Since the latter yielded the best results, we shall focus on it. The stepwise regression method that we have implemented uses a bidirectional elimination approach. It starts with no predictors for the drift function. Then, at each step until convergence, it adds or removes an orthonormal polynomial term by comparing the Akaike Information Criterion (AIC) improvement that results from each possible decision. The procedure stops when no more predictors can be added or removed from the model. The method is then repeated for the diffusion term.

Regarding the SGP method, the same kernel was selected for estimating both the drift and diffusion terms:

$$k(\boldsymbol{\xi}, \boldsymbol{\xi}', A, \boldsymbol{\theta}) = \theta_0 \exp\left[-\frac{\theta_1}{2}\|\boldsymbol{\xi} - \boldsymbol{\xi}'\|^2\right] + (A - \theta_0). \tag{3.26}$$

The kernel k is a linear combination of a squared exponential kernel (first term in the Right-hand side (RHS)) and a constant kernel (second term in the RHS). Note that the hyperparameter θ_1 determines the characteristic length-scale of the GP ($l^2 = 1/\theta_1$). The constant covariance function was included since a constant diffusion term is often used in the literature. It must be noted that we have not treated the parameter A as an hyperparameter subject to optimization

(we have not included it into the hyperparameter vector $\boldsymbol{\theta}$). We prefer to keep it fixed so that the total amplitude of the diagonal of the covariance matrices that result from k always sum up to A. In this way, A can be interpreted as the prior belief about the variance of the drift/diffusion term. This eases the comparison between several optimization runs using Equation (3.23), since all the estimates share the same prior belief about the range in which the dynamic terms may lie. Note that in order to fulfill $k(\boldsymbol{\xi}, \boldsymbol{\xi}', A, \boldsymbol{\theta}) \in [0, A]$ we must perform a box-constrained optimization of θ_0 ($\theta_0 \in [0, A]$), which originally motivated the use of L-BFGS-B as the optimization algorithm.

Since it is usual to get ill-conditioned covariance matrices when working with GPs we slightly modified Equation (3.26). To regularize the covariance matrices a small value on the principal diagonals was added. In general, any type of kernel $\mathcal{Q}$ can be modified to improve stability as follows:

$$\mathcal{Q}'(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}, \epsilon) = \mathcal{Q}(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}) + \epsilon\delta(\boldsymbol{x} - \boldsymbol{x}'), \tag{3.27}$$

where we did not state the dependencies of $\mathcal{Q}$ that are not treated as hyperparameters (e.g., A in Equation (3.26)). When using the modified squared exponential kernel k', we did not optimize on the ϵ parameter to avoid creating large discontinuities in the covariance function.

Since all the models used for testing have very smooth functions and they generate time series with a range of the order of 1, we may expect good estimates with only a few inducing-points. For example, the drift function of M_2 has three roots at -1, 0 and 1 and, therefore, a reasonable estimate for its length-scale would belong to $[0.5, 1]$. A typical trajectory of M_2 would probably lie in the interval $x(t) \in [-2, 2]$ and hence, an estimate of the m based on Equation (3.26) would yield $m = 4/0.5 = 8$. To verify our intuitions, we have followed Rasmussen's approach [111] (see Section 3.4.1). We have calculated the integrated error of the drift function for M_6 on a small subset of simulations while testing how the computation time scales with m. For this exploratory analysis we used $m = 5, 10, 15, 20$ and 30. The drift function for M_6 was selected for the test since it is probably the most complex one. Figure 3.3 shows that there are not big differences in the integrated errors for $m \geq 10$, whereas the time per iteration quickly increases.

Based on these exploratory results, we finally run our SGP method on the whole experimental dataset using $m = 2, 5, 10$ and 15. It should be noted that $m << N$ and hence, we could have used larger m without compromising the computational tractability of the problem. Also note that although Figure 3.3 suggests that we could stop searching at $m = 10$, we have also included $m = 15$. This was done to compare both estimates and further investigate the impact

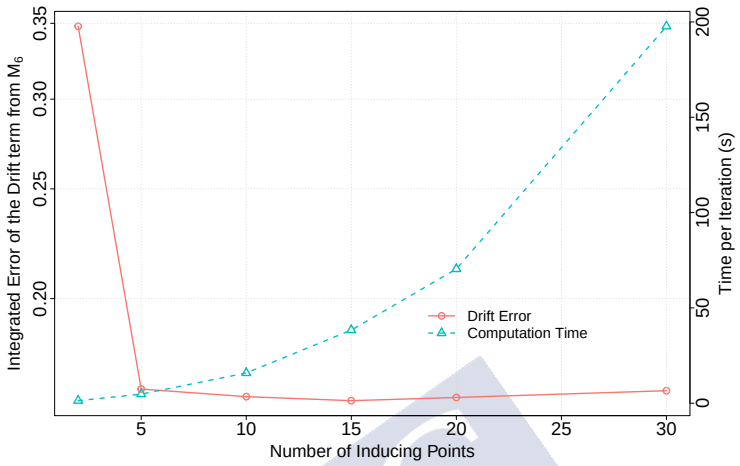

Figure 3.3: Drift's integrated error and computational time per iteration depending on m for a small subset of M_6 simulations.

of m in the lower bound. Furthermore, Figure 3.3 was obtained using a small subset of the data from a single model and, therefore, there may be simulations for which the use of $m = 15$ may yield much better estimates.

For each value of m, several trials with randomly selected initial values of the hyperparameters were run. The length-scales of both drift and diffusion kernels were restricted to the interval $l \in [0.25, 2]$, based again on the fact that all the time series have a range of the order of 1. The value A_f was set to 25 (equivalent to a standard deviation of 5) and the initial value of $\theta_{f,0}$ was randomly initialized into the interval $[0, A_f]$. The selection of ν and the amplitude hyperparameter for the diffusion process was made using Equation (3.24) and $A_g = 25$ for all the models present in the simulated set. The starting values for the pseudo-inputs were selected using the percentile initialization. The final model for each of the time series was selected by using the modified lower bound (Equation (3.23)).

Table 3.2 summarizes the mean values of the integrated errors for all the models from Table 3.1. The best result for each model is marked in bold (smaller is better). Additionally, a star (*) points those best-results with statistically significant differences with respect to the other two methods. The differences between methods were tested using the Nemenyi post-hoc test [100]. The results in Table 3.2 show that our proposal has a good performance, specially in the drift estimates, where it performs better than KBR and POLY for all the models but one.

Table 3.2: Integrated absolute errors of the methods KBR and POLY and our proposal (denoted as SGP), using different test models with length $N = 10^4$.

	Drift estimates			Diffusion Estimates		
Model	KBR	POLY	SGP	KBR	POLY	SGP
M_1	0.6863	0.7896	**0.4992***	0.03963	0.03426	**0.02684***
M_2	**0.5073***	0.6267	0.5760	0.01915	0.01878	**0.01511***
M_3	0.1501	0.2731	**0.1232***	0.05293	0.02711	**0.007465***
M_4	0.1244	0.1519	**0.1128***	0.02585	**0.002054***	0.0045
M_5	0.09035	0.1613	**0.08256***	**0.001338***	0.001771	0.002667
M_6	0.2289	0.2618	**0.2256**	0.002751	0.002972	**0.002323***

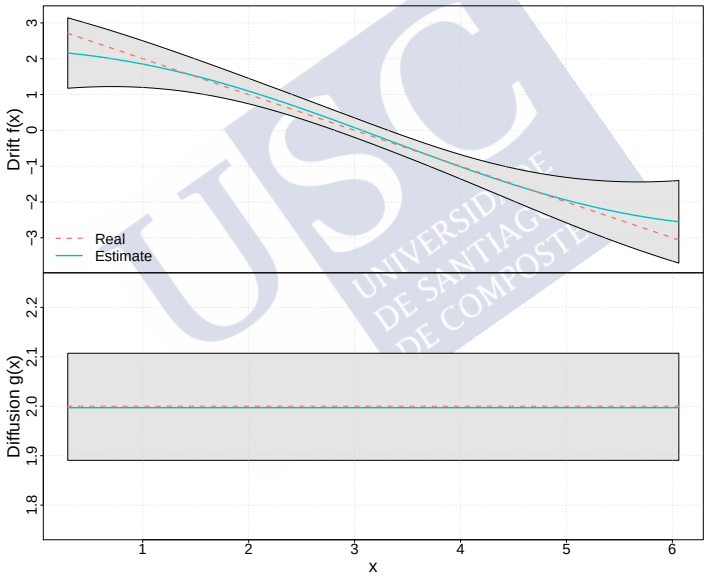

Figure 3.4: Drift and diffusion estimates obtained from a single trajectory of M_1.

The results for the diffusions are also good, but the SGP method has the largest mean error for the M_5 model. The reason for this shall be discussed later in this section.

Figures 3.4-3.9 illustrate the kind of estimates that the SGP method yields for the drift and diffusion terms from a single realization of the simulated models. The shaded area represents the 95% confidence region. Note that these confidence intervals usually increase when x takes extreme values. This is due to the fact that the regions where x takes extreme values are only

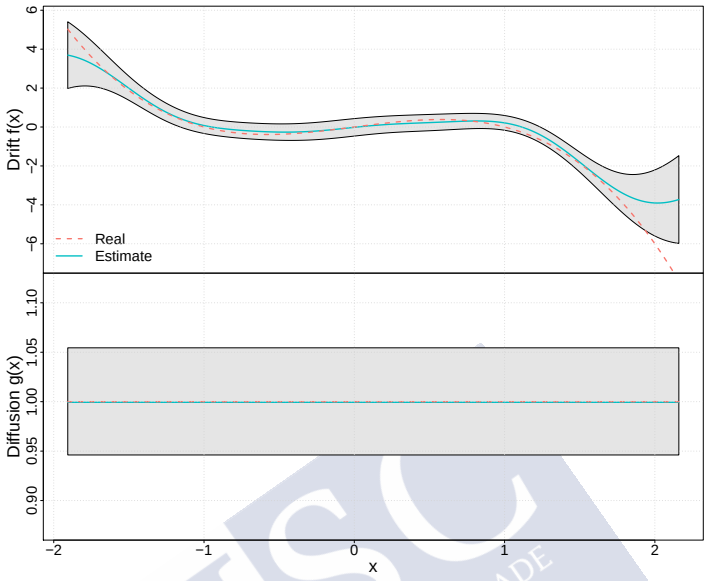

Figure 3.5: Drift and diffusion estimates obtained from a single trajectory of M_2.

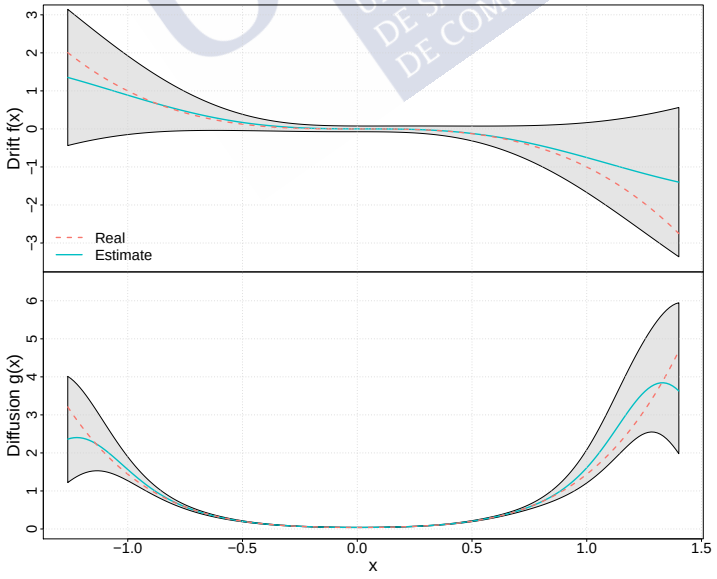

Figure 3.6: Drift and diffusion estimates obtained from a single trajectory of M_3.

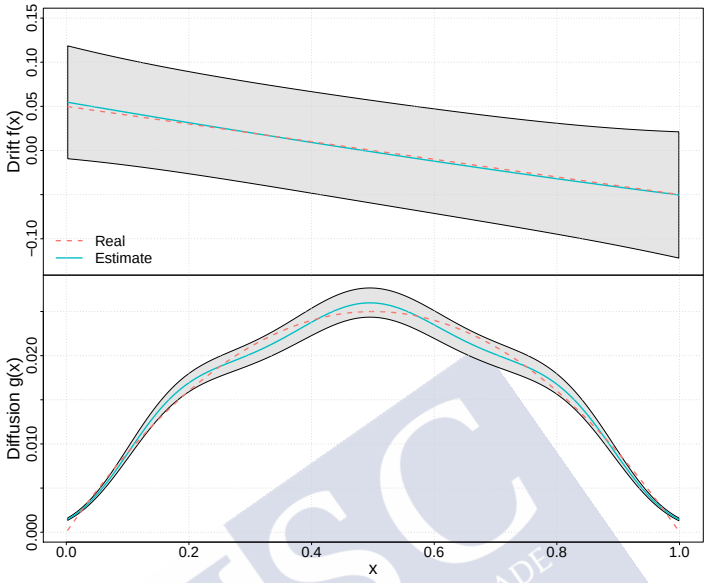

Figure 3.7: Drift and diffusion estimates obtained from a single trajectory of M_4.

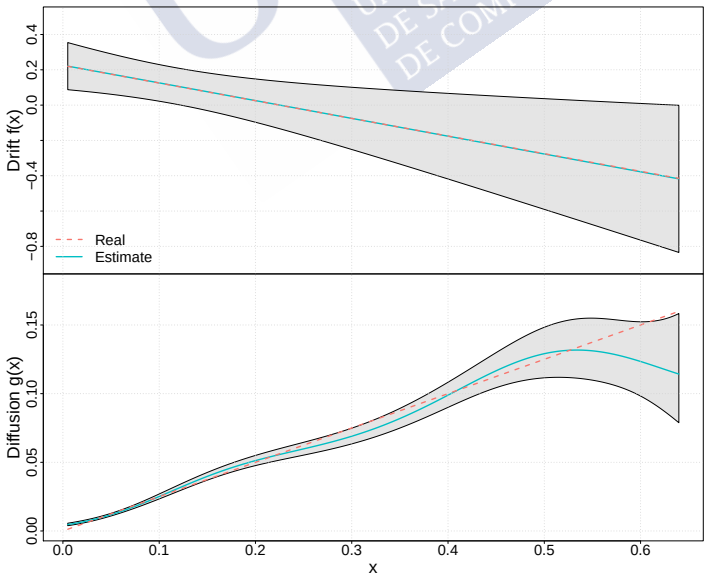

Figure 3.8: Drift and diffusion estimates obtained from a single trajectory of M_5.

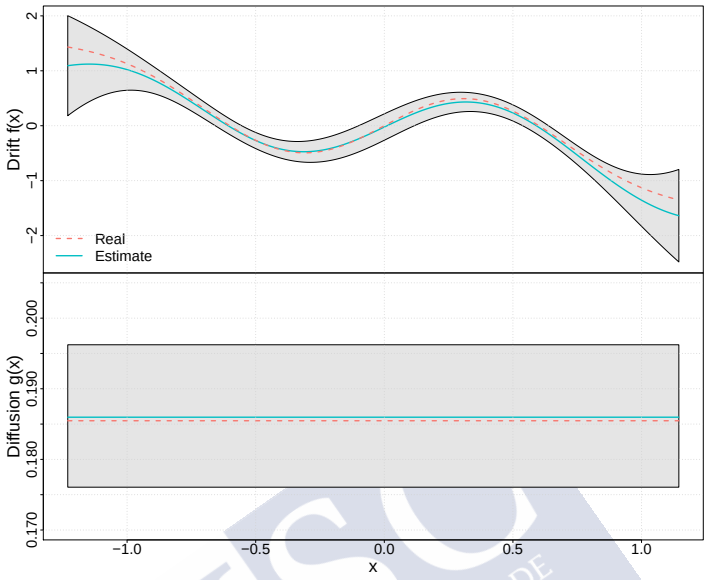

Figure 3.9: Drift and diffusion estimates obtained from a single trajectory of M_6.

visited a few times during any simulated trajectory and hence only a few points are available for the estimation. Since there is little data at these regions, the priors have strong influence and the estimates tend to curve towards the prior means. This effect is particularly remarkable for the drift estimates, which curve towards zero, and the diffusion for M_5. This is probably the reason why the SGP method does not perform as well as expected for the diffusion for M_5 and the drift for M_2.

Concerning the selected number of pseudo-inputs m, the general trend is that $\mathcal{L}'$ (see Equation (3.23)) increases with m, as we might have expected (see Section 3.4.1). Hence, all the selected models use $m = 15$ inducing points. However, it is not always worth to increase m in terms of the integrated error versus the running time, which scales as $\mathcal{O}(2^m)$ due to the use of the L-BFGS-B algorithm (see Figure 3.3). This can be understood by looking at Figure 3.10. The figure shows two estimates of the M_5's diffusion term obtained using a different number of inducing-points, which are also represented in the plot. As noted with Figure 3.10 the width of the confidence intervals (gray regions), depends on the number of points available for the estimation, illustrated with the point cloud. The similarity between both estimates over the high-density region results in an almost identical weighted integrated error. However, the $\mathcal{L}'$

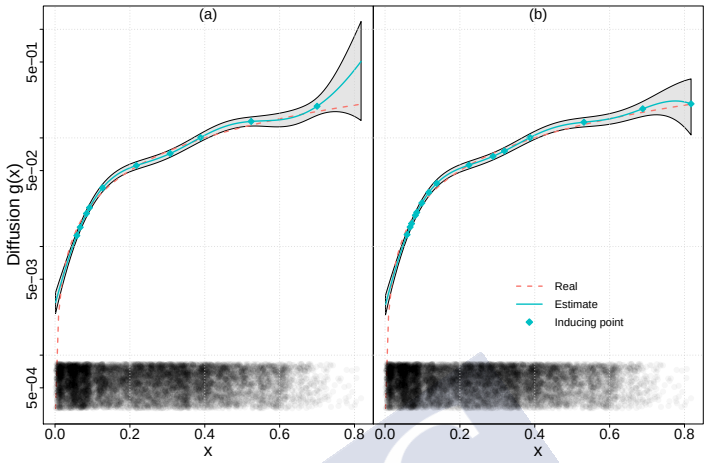

Figure 3.10: Diffusion estimates obtained using (a) $m = 10$ and (b) $m = 15$ inducing points. The density of the point cloud at the bottom of the figure represents the number of points available for the estimation at each x.

is larger for $m = 15$ than for $m = 10$, mostly because the confidence interval significantly increases in the low-density region for $m = 10$. The use of additional inducing-points in the case $m = 15$ permits a better control of the estimates and the confidence interval, which results in a larger $\mathcal{L}'$ although the weighted integrated error is very similar. Hence, the $\mathcal{L}'$-based selection criteria is not optimal for the purpose of minimizing the weighted integrated error without wasting computational resources. From these experimental results about the impact of m in $\mathcal{L}'$ we conclude that the selection of m should not be based solely on the lower bound, since it monotonically increases with m at a cost of greater computation times. Therefore, we suggest adopting Rasmussen's heuristic (Section 3.4.1) in combination with $\mathcal{L}'$, using Equation (3.25) as an initial guess for the value of m.

3.6 Application to financial data

In this section, we show the application of our method to a real dataset from econophysics with the aim of illustrating the applicability of SDEs to non-stationary problems and the role that non-constant diffusions play in complex dynamics. We study the daily fluctuations in the oil price in the period 1982/01/02-2017/05/30, which results in a time series $p[n]$ of length

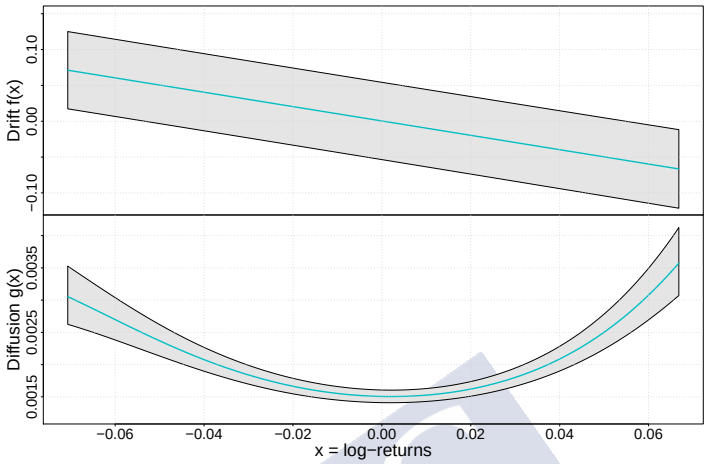

Figure 3.11: Drift and diffusion estimates obtained with the SGP method on the oil price log-returns.

$N \approx 10^4$ [130]. Following [52], we constructed the daily logarithmic increments of the oil price $x[n] = \log p[n+1]/p[n]$, where $p[n]$ is the price of the day n, to obtain a stationary time series. The SGP method was then applied using $m = 10$ inducing points (randomly started using the percentile initialization) and two squared exponential kernels. The numerical stability of the kernels was improved using Equation (3.27). The amplitudes of the kernels were selected to match a standard deviation of 5 for both the drift and diffusion functions. The method was run several times with random initial values for the length-scales. The final estimates selected using the lower bound are shown in Figure 3.11. These estimates are in good agreement with those reported in [52] (although this work focused in a smaller period). Similar estimates are also obtained using the KBR and POLY methods.

The drift and diffusion functions shown in Figure 3.11 can be approximated by $\hat{f}(x) \approx -x$ and $\hat{g}(x) \approx D + \gamma x^2$, which yields the SDE of a quadratic-noise Ornstein-Uhlenbeck process [53, Chapter 3]. This process is an illustrative example of the effects that multiplicative noise may have in the dynamics of a system. The stationary distribution of a quadratic-noise Ornstein-Uhlenbeck process is a non-standardized Student's distribution, which is a heavy-tailed distribution that permits the occurrence of large values in the log-returns series. Furthermore, this stationary distribution is more closely confined to the origin in comparison with the standard Ornstein-Uhlenbeck noise, which implies that the stable state is narrower

in the quadratic case. This is an example of noise-enhanced stability [53, Chapter 3] and illustrates the importance that the non-parametric estimation of the diffusion may have in the study of complex dynamics.

3.7 Application to paleoclimatology data

In this Section, we apply our estimation method to a dataset related to paleoclimatology. Climate records from the Greenland ice cores have played a central role in the study of the Earth's past climate in the Northern hemisphere. Among other interesting phenomena, these records show abrupt rapid climate fluctuations that occurred during the last glacial period, which ranges from approximately 110 Ky (1 Ky = 1000 years) to 12 Ky before present. These abrupt climate changes are usually referred to as Dansgaard-Oeschger (DO) events. Although there seems to be a general agreement that DO events are transitions between two quasi-stationary states (the glacial or stadial and the interstadial states), the nature of the phenomena triggering the transitions is still actively debated. It has been argued that the DO events occur quasi-periodically with a recurrence time of approximately 1.47 Ky [120]. However, recent studies support that the DO events are probably noise induced [28, 29, 79].

We applied our method to the δ^{18}O record during the last glacial period obtained from the North Greenland Ice Core Project (NGRIP) [3]. The δ^{18}O is a measure of the ratio of the stable isotopes oxygen-18 and oxygen-16 which is commonly used to estimate the temperature at the time that each small section of the ice core was formed. It is measured in "permil" (‰, parts per thousand) and its formula is:

$$\delta^{18}\mathrm{O} = \left(\frac{\left[\frac{^{18}\mathrm{O}}{^{16}\mathrm{O}}\right]_{\text{sample}}}{\left[\frac{^{18}\mathrm{O}}{^{16}\mathrm{O}}\right]_{\text{reference}}} - 1 \right) \cdot 1000\ ‰,$$

where *reference* defines a well-known isotopic composition.

Figure 3.12 shows the oxygen isotopic composition from the NGRIP ice core. We considered the period ranging from 70 Ky to 20 Ky before present as in [79], since it is dominated by the DO events, as it can be clearly observed.

We applied our method using different kernels to illustrate that different covariance functions can be used and combined to create different models, and that Equation (3.23) can be used to select the best among them. Within our method, testing different kernels is important because we usually do not have enough information about the drift and diffusion terms to

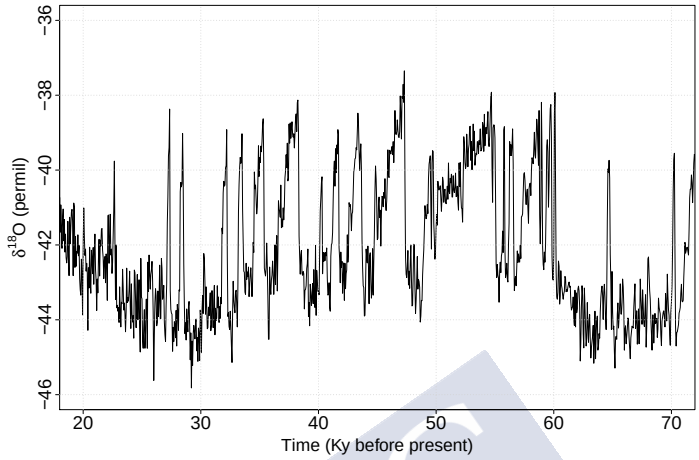

Figure 3.12: DO transitions during the last glacial period.

decide among them. Furthermore, the performance of GPs depends almost exclusively on the suitability of the chosen kernel to capture the features of the modeled function. Consider the following illustrative example: a function with fast quasi-periodic oscillations superimposed on a linear trend. A squared exponential kernel with a large length-scale can capture the behavior of the linear slope and make reasonable predictions of the trend for unobserved values, but it won't be able to model the quick wiggles. On the other hand, a squared exponential kernel with a small length-scale will be able to accurately fit all the data but, since the distance from the training points rapidly increases, it won't be able to make good predictions for unobserved values, not even for the trend. Furthermore, the uncertainty of the unobserved values will also scale fast. A better covariance choice could make use of a sum of exponential kernels with different length-scales, which would permit to accurately fit the data and make good predictions for the trend. More complex kernel choices are also possible. For a complete example on the impact of the kernel in the modeling capabilities of a GP, see [111, Chapter 5]. For our illustrative example on the paleoclimate data, we used the kernel specified in Equation (3.26), a sum of two exponential kernels with different length-scales, and a rational quadratic kernel. All these kernels were modified adding a small value to their main diagonals as in Equation (3.27).

For each possible kernel, the method was started with random values for the hyperparameters. The number of the pseudo-inputs was set to $m = 15$, based on the good results achieved

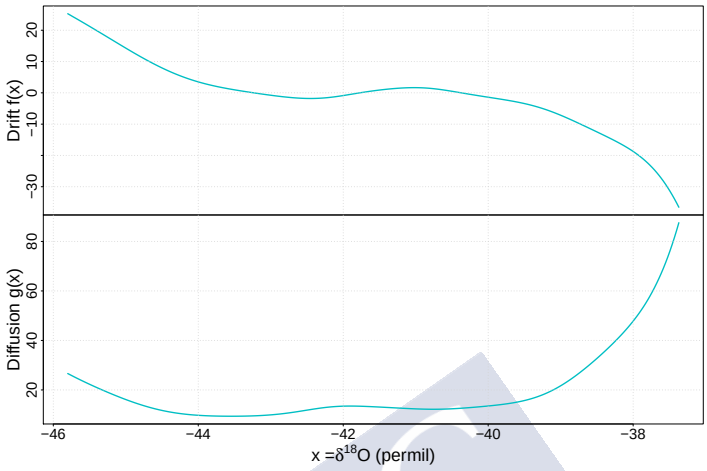

Figure 3.13: Best drift and diffusion estimates using the KBR method.

in Section 3.5. The amplitudes of the kernels were selected so that they were compatible with a standard deviation of 30 for both the drift and diffusion functions. The estimates selected based on the value of the modified lower bound (Equation (3.23)) are illustrated in Figure 3.14. The drift term was obtained using a rational squared kernel whereas the diffusion term was estimated using the kernel from Equation (3.26). Note that, as expected, the drift function presents two stable points: one corresponding to the stadial state and the other corresponding to the interstadial state. Integrating the drift function yields the potential function, which indicates that the stadial state corresponds to a stable state of the system since it has the lowest energy. On the other hand, the interstadial state corresponds to a metastable state.

The SGP estimate supports the use of a state-dependent diffusion rather than the widely-used constant term. The use of a state-dependent diffusion for the DO events was first proposed in [79], who suggested

$$f(x,\boldsymbol{\theta}) = \sum_{i=0}^{3} \theta_j x^j; \qquad g(x,\boldsymbol{\theta}) = \begin{cases} \theta_4 \text{ if } x < \theta_6 \\ \theta_5 \text{ if } x \geq \theta_6 \end{cases}, \tag{3.28}$$

while testing their framework for parametric inference and model selection for SDEs [79]. The authors discussed the model from Equation (3.28) since it is able to accurately predict the histogram of the DO events, although the final parametrization that results from their model selection criteria proposes a constant diffusion term. However, our non-parametric

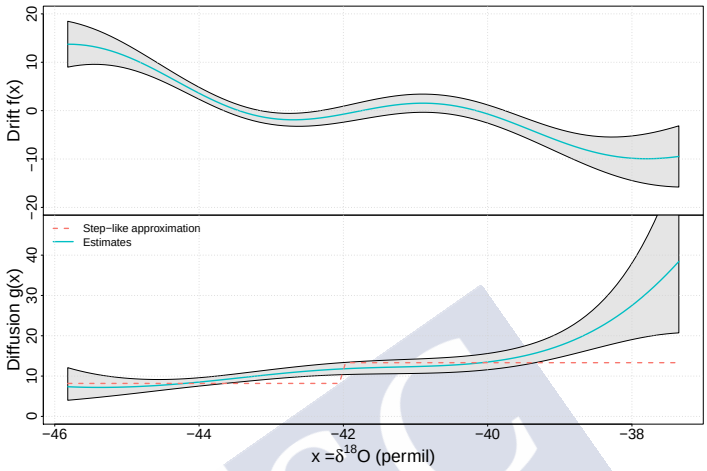

Figure 3.14: Best drift and diffusion estimates using the SGP method.

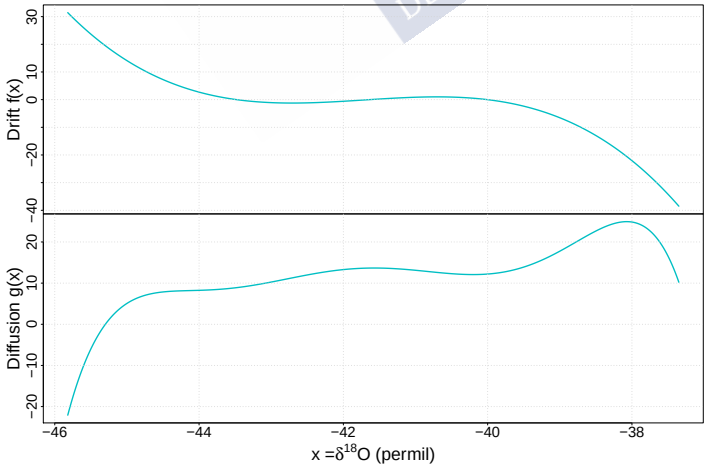

Figure 3.15: Best drift and diffusion estimates using the POLY method.

methodology suggests that the state dependent diffusion is indeed preferable. Despite it is possible to approximate the SGP diffusion's estimate using a step function (as can be appreciated in Figure 3.14), there exists a linear increasing region for $x > -40$ that does not match Krumscheid's model. To compare the diffusion model in Ref. [79] with the SGP's diffusion model, a Lasso estimate [24] was applied to the diffusion term while keeping the drift term fixed. Lasso penalties are very useful in regression analysis since they are able to set coefficients to zero, eliminating unnecessary variables. Hence, by using the diffusion term:

$$g(x, \boldsymbol{\zeta}) = \begin{cases} \zeta_1 & \text{if } x < -42 \\ \zeta_2 & \text{if } -42 \leq x < -40 \\ \zeta_2 + \zeta_3(x + 40) & \text{if } x \geq -40 \end{cases},$$

we can compare the model proposed in [79] (which corresponds with setting $\zeta_3 = 0$) and our estimate. The Lasso estimate provides evidence in favor of the model obtained through our method, since the ζ_3 is not eliminated. Note, however, that this evidence it is not conclusive since the time series used for the estimates is quite short ($N \approx 10^3$) and there is a lack of points for $x > -39$.

We have also used the SGP estimates to compute the distribution of the time between DO events. 1000 new time series were generated using the Euler-Maruyama scheme. The initial points were sampled with replacement from the real paleoclimate series. To robustly identify the DO states, we fitted a Hidden Markov Model (HMM) with three states and Gaussian response to the real data. The aim of the three states is to clearly identify the stadial state, the interstadial state and a "transition state". We identified the states of each of the simulated time series using this HMM by means of the Viterbi algorithm [132]. The resulting mean time between DO events was 1.50 Ky, in good agreement with the generally accepted value of 1.47 Ky [120]. However, it must be noted that this value was obtained on basis of a quasi-periodical model, whereas our value is based on a stochastic model.

Using the KBR (Figure 3.13) and POLY (Figure 3.15) methods result in similar drift estimates, compared with the SGP one. Furthermore, both diffusion estimates also support the use of a state-dependent model. However, the SGP estimate provides confidence intervals based on the Bayesian formalism while the others methods do not. Also, the KBR method presents an unlikely increment in the diffusion for $x < -44$. The POLY method approximates the shape of the state-dependent diffusion using a high order polynomial, which cannot properly capture the plateau for $x < -44$. Additionally, the polynomial fit results in negative values for

$x < -45$ which make no sense for a diffusion term.

3.8 Discussion

In this chapter, we have presented a non-parametric estimation method for SDEs from densely observed time series based on GPs. The only assumptions made on the data are stationarity, Markovianity, and that the sampling period is small enough so that the Euler-Maruyama discretization holds. From the point of view of the adoption of GPs to the estimation of SDEs, the main contributions of this chapter are: (1) providing estimates for any type of diffusion function and (2) proposing a sparse approximation to the true GP posterior that permits to handle efficiently the typical experimental time series size of $N \approx 10^3 - 10^5$.

To cope with the computational complexity of calculating the posterior distribution of the GPs (which scales as $\mathcal{O}(N^3)$), we approximate the GPs using the evidence provided by the data in only a small set of function points: the inducing variables. The inducing variables are learned by minimizing the Kullback-Leibler divergence between the true posterior GP distribution and the approximated one. The minimization problem is approached using the standard techniques from the variational inference framework, which usually yields a coordinate-ascent optimization to approximate the posterior. However, our approach makes use of a non conjugate model due to the inclusion of the diffusion function, which prevents the direct use of variational inference methods. To tackle the problem, a Laplace approximation was used to compute the distribution modeling the diffusion. It must be noted that, although we have developed our estimation approach bearing in mind the computational challenges that a large N imposes, our proposal can also handle small time series without any further adjustment. Also, although the SGP approximation permits handling large experimental time series, N cannot increase without limit. Variational inference algorithms require a full pass through the whole dataset for updating the distributions (once each iteration). Thus, the larger the dataset, the slower our algorithm updates the parameters, and the longer it may take until convergence (even when using sparse techniques like the proposed one). Furthermore, variational inference may be unfeasible if the whole dataset does not fit in main memory. Scaling up variational inference can be done using stochastic gradient optimization, which yields stochastic variational inference [64]. In stochastic gradient optimization, the parameters are frequently updated using small batches of data, which usually results in a faster convergence. This enables the use of variational inference with large datasets, even those that do not fit in

memory. Since large datasets are increasingly common, the use of this kind of techniques is considered in Chapter 4.

The performance of the SGP estimates was evaluated using simulated data from different SDE models and compared with the kernel-based method [82] and the polynomial-based method [110]. The results show that the SGP approach is able to provide very accurate estimates, specially for the drift term. The main advantage of the SGP method with respect to [82] and [110] is that it permits a Bayesian treatment of the estimation problem; this enables obtaining probabilistic predictions and computing robust confidence intervals. Furthermore, the prior information about the drift/diffusion is expressed in a function-space view, i.e., the SGP method permits specifying the prior directly over functions instead of working with weights of some basis expansion. In our view, this is a more natural way of working with functions. Another major advantage of the proposed method is its versatility. Although we have focused on very flexible kernels, any type of kernel (or even combinations of them) can be used, which may completely change the properties of the posterior estimates. For example, using polynomial kernels would yield similar estimates to those of [110], but with the aforementioned advantages of the Bayesian framework and without the possibility of obtaining negative values for the diffusion (see Section 3.7).

We applied the SGP method to a real problem in econophysics with the aim of illustrating the importance of non-constant diffusions in the behavior of a system and, hence, the importance of its non-parametric estimation. This example also emphasizes the applicability of the SDE framework to non-stationary time series by means of transformations commonly used in time series analysis (e.g., the log transformation).

The proposed method was also applied to a real paleoclimate time series: the NGRIP core data showing the DO events occurring during the last glacial period. The SGP method accurately captures the relevant physical states of the time series (the stadial and interstadial states) and yields a mean transition time between DO events that it is close to the accepted value in the literature (which is calculated under the assumption of a deterministic periodic model). This demonstrates its ability to capture the behavior of real data with complex dynamics. Furthermore, the SGP estimates provide evidence supporting a novel state dependent diffusion model for the DO events. This diffusion model is similar to the step-like function proposed in [79] for a wide range of the diffusion's support, but it also adds a linear term for the region corresponding to the most extreme values of the DO events. These results should be viewed with caution, since the estimates were made using small amounts of data. Further research to

assess the physical meaning of the model should be made.

An interesting line of future work would be to generalize the kind of noises driving the SDE so that fBm could be used as stochastic force. Although it is very appealing since it would directly connect the present chapter and Chapter 2, we shall not consider it in the present thesis. On the one hand, the mathematical foundations required to formalize the meaning of the differential of a dfBm $dW^H(t)$ are quite sophisticated [80]. On the other hand, if we take an intuitive approach and we interpret $dW^H(t)$ as a fGn, then the methods developed in this chapter would not significantly change and therefore, we would not add any meaningful contribution to the framework already developed.

The following chapter tries to overcome the main restriction that the use of SDEs imposes: the Markovianity assumption (see the introduction of this chapter). As discussed in Chapter 2, many natural occurring systems have long-term correlations and therefore, they cannot be accurately described by a SDE. If the time series under study $y[n]$ only had significant dependencies with a few temporal lags, say $y[n-1]$ and $y[n-2]$, we could easily overcome the issue by constructing a new multidimensional time series $\boldsymbol{y}[n] = [y[n-2], y[n-1], y[n]]$ and then proceeding with the method presented in the current chapter. The method developed on Chapter 4 builds on this intuition that we can transform a non-Markovian time series in a high-dimensional Markovian representation. However, instead of concatenating lagged samples and then applying some SDE estimation method, we integrate novel deep learning techniques with the SGP method of this chapter to *simultaneously* learn a Markovian representation of the data and its dynamics.

CHAPTER 4

ADDRESSING NON-MARKOVIANITY THROUGH VARIATIONAL AUTOENCODERS

The modeling of complex dynamical systems often relies on the concept of state space or phase space. In a deterministic system, the phase space consists of all the possible states of a dynamical system, with each state of the system corresponding to a unique point in the phase space. The system state at time t contains all the information that is needed to determine the future system states for any instant $> t$. For a system that can be mathematically modeled, the phase space is known from the dynamic equations. However, as discussed in Chapter 1, the mathematical description of many complex natural systems is usually not accurate. Furthermore, what we observe in an experimental setting is not a phase space but a set of time series representing the temporal evolution of different measurable aspects of the system. This is true for both simple and complex systems. An added difficulty is that, in some cases, some information may be missing. For example, for very complex systems, which we may think of as requiring an infinite set of equations for its description, many state variables cannot be measured directly. An illustrative example of this kind of systems is the cardiovascular system, which is usually characterized through indirect measurements such as heart rate variability [50]. Even for simpler dynamical systems, some signals may not be measured due to limitations in the available instrumentation. These considerations lead to the important problem of phase space reconstruction from a set of measurements $y(t)$. A simple idea, following the principles of classical dynamics, would be to use the time series itself and its derivatives $\dot{y}(t)$, $\ddot{y}(t)$, ..., to build a phase space. This idea is not usually feasible due to the

fact that the measurement noise gets amplified with each derivative and hence even second order derivatives will look like noise.

The phase space reconstruction problem was theoretically solved by the Takens' theorem [124]. Takens' theorem states that if we are able to observe a single scalar quantity $y(t)$ that depends on the current state of the system $\boldsymbol{x}(t)$,

$$y(t) \triangleq y(\boldsymbol{x}(t)), \tag{4.1}$$

then, under quite general conditions, the structure of the multivariate phase space can be unfolded from this set of scalar measurements by means of the so-called delay embedding. A delay embedding in d dimensions is formed by using delay coordinates:

$$\boldsymbol{y}(t) = [y(t), y(t-\tau), y(t-2\tau), \ldots, y(t-(d-1)\tau)]^T,$$

where the time distance τ between adjacent coordinates is usually referred to as the time lag or time delay. Takens' theorem guarantees that the new geometrical object formed by $\boldsymbol{y}(t)$ is topologically equivalent to the original phase space if the embedding dimension d is sufficient large. Specifically, the delay map should use $d > 2D_F + 1$ dimensions, being D_F the number of the active degrees of freedom of the system [118]. According to the Takens' theorem, the value of the lag τ is arbitrary. However, in practice the proper choice of τ is important. If τ is too small the delay vectors will be very similar between them, and therefore they will tend to cluster around the bisectrix of the phase space. On the other hand, if τ is too large, the delay vectors will be almost uncorrelated, resulting in a very complex phase space. Several methods have been proposed to select the time lag τ. A simple yet effective strategy is to select τ as the first minimum of the autocorrelation function or the average mutual information [71].

Takens' theorem only guarantees the preservation of the attractor's topology, i.e., there is a unique mapping between the original phase space and its reconstruction which is given by a unique invertible smooth map. The geometry of the attractor is not necessarily preserved, since bending and stretching are allowed mappings. Therefore close points on the original attractor may end up far in the reconstructed phase space. This makes Takens' theorem sensible to noise, since small fluctuations could have large effects on the delay reconstruction [137].

Furthermore, although the noise may be small enough so that it can be ignored, the assumption that the experimental data can be described by a deterministic differential equation is often unrealistic, as mentioned in Chapter 1. Complex systems usually involve a number of degrees of freedom much larger than what an experiment can resolve. In this case, considering

a dual deterministic-stochastic model in which noise represents the unresolved deterministic dynamics may be more appropriate than a pure deterministic one.

The stochastic version of an equation of motion is the SDE or Langevin equation, presented in Chapter 3. The SDE for a d-dimensional state vector $\boldsymbol{x}(t)$ reads

$$dx_i(t) = f_i\big(\boldsymbol{x}(t)\big)dt + \sum_{j=1}^{d} \sqrt{g_{ij}\big(\boldsymbol{x}(t)\big)}\, dW_j(t), \qquad i = 1, 2, \ldots, d \tag{4.2}$$

where $\{W_j(t)\}_{j=1}^{d}$ denote d independent Wiener processes (see Section 3.1). Equation (4.2) is also concerned with the same issues previously discussed when analyzing experimental data. That is, the SDE approach only works if the full state vector $\boldsymbol{x}(t)$ can be measured. However, this is an unrealistic assumption in most experimental settings, where only a few scalar time series are available.

To cope with the complexity of analyzing time series arising from stochastic systems Markov process are often used, which naturally incorporate the concept of phase space. In a Markov process, each state determines the probability of the transitions to the next state. Indeed, a deterministic system driven by a differential equation may be seen as a special Markov process in which the future distributions follow a Dirac's delta distribution. More generally, a Markov process of order n uniquely determines the transition probabilities given the n previous states of the Markov process.

Since a SDE generates a Markov process, we could expect that a time series from a single observable could be described by a Markov process of some order, probably larger than the order of the complete SDE Markov process but somewhat related, in analogy with the Takens' theorem. Unfortunately, this assumption is wrong and no stochastic-embedding theorem exists [71, Chapter 12]. The reason for this is that scalar time series originated from a continuous Markov process is not longer Markovian since it has infinite memory [114]. However, in most cases the memory decays exponentially fast and hence a Markov process can effectively approximate the time series under study. When using a Markov process of order n for analyzing the scalar time series $y[k] = y(k \cdot \Delta t)$ the dynamics of the time series are determined by the transition probabilities $p\left(y[k] \mid y[k-1], y[k-2], \ldots, y[k-n]\right)$. This may be interpreted as a embedding where $d = n$ and $\tau = \Delta t$, although there is no theorem that guarantees that it will have nice properties for the analysis of the underlying dynamical system. Note that, in this context, the new embedding space may be seen as the result of a transformation that is able to represent a scalar time series as a vector Markov process. Having such a transformation

will overcome the need of the Markovian property to be able to use the SDE-based techniques developed in Chapter 3.

In this chapter, we propose a unifying framework to find a proper embedding for both deterministic and deterministic-stochastic time series and, at the same time, learn a SDE-like equation describing the dynamics in the embedding space. Although we have introduced the problem with scalar signals, for the sake of generality, we shall consider vector time series $\boldsymbol{y}[n]$. This permits us to study the issue of incomplete experimental data, where only a few of the time series that define the systems are available (including the case of a single scalar time series); or, in general, any non-Markovian vector signal.

The method should be able to cope with the possible infinite memory of the original time series by finding a robust transformation into a new vector space. The new vectors should comprise all the dynamical information to determine the evolution of the system in a Markov space, enabling its description by a SDE, which also needs to be discovered. Finally, a new transformation should permit recovering the original time series. In this way, the resulting embedding description could also be used as a generative model. That is, it would be possible to generate new synthetic data by sampling from the probability distribution induced by the method.

To tackle the problem, we shall use of a Structured Variational Autoencoder (SVAE) [69] with the SGP-based estimation method presented in Chapter 3. SVAEs enable the combination of graphical models, which permit to express the Markovian probability distributions of the dynamics, with deep learning methods able to learn encoding and decoding transformations.

To get an intuitive understanding of SVAEs, we shall first start from a simple pair of neural networks, and we shall add parts to them step by step. The two networks work as a pair of encoding and decoding mechanisms. Ideally, the encoding network is able to disentangle the state vector $\boldsymbol{x}(t)$ from the observed scalar time series $y(t)$ (see Equation (4.1)). In practice, the true state vector $\boldsymbol{x}(t)$ cannot be exactly recovered from $y(t)$, but the "code" generated by the encoding network should hold all the relevant dynamic information. To keep things simple, we will also refer to this code as $\boldsymbol{x}(t)$ (the state vector). From our assumptions, the original observations $y(t)$ can be recovered from $\boldsymbol{x}(t)$. The decoding network is responsible for translating $\boldsymbol{x}(t)$ to $y(t)$. The cascade of the encoding and decoding networks is analogous to the structure of an autoencoding neural network or autoencoder [14, 62]. An autoencoder is a neural network able to compress the data from the input layer into a short code, and then reconstruct the input from this representation. This is illustrated in Figure 4.1 using images as

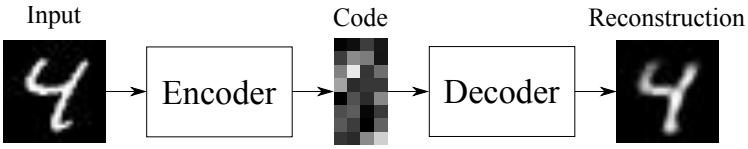

Figure 4.1: An image autoencoder. The code acts as a compressed representation of the input image, from which it is possible to obtain an approximate reconstruction. Note that the reconstructed image misses some of the details from the input.

data. This short code is usually referred to as latent vector. When applied to time series, this latent vector will play the role of a phase space vector and therefore, we shall use both terms interchangeably.

The main limitation of autoencoders is that they cannot generate (at least in a simple way), new data from what they have learned, since we do not know how to create the codes other than encoding them from already seen inputs. Furthermore, the encoded vectors may lie in a non-continuous space, which prevents extrapolation from the learned codes. Variational Autoencoders (VAEs) solve this issue by forcing the latent vectors to approximately follow a standardized Gaussian distribution [76, 113]. This simple trick makes creating generative models easy: all what we have to do is sampling from a standardized Gaussian distribution and passing the resulting latent vector to the decoder. Note that this procedure highlights a strong assumption made by VAEs: each observation entering the VAE is independent from the others.

SVAEs generalize VAEs by permitting the latent space to have a richer structure, usually described by structured graphical models (hence, its name). In our case, a VAE would permit forcing the phase space to approximately have Markovian dynamics described by a SDE. However, we formulate a SVAEs geared for describing SDE-like dynamics in Sections 4.2 and 4.3, with the aim of simplifying the discussion and future extensions of the model.

Both VAEs and SVAEs permit leveraging graphical models (either simple or structured) with deep learning methods. However, using probabilistic models with highly nonlinear methods such as neural networks comes with a cost. Inference cannot be done exactly and, therefore, variational approximations must be used (which motivates the "variational" part of their name). Section 4.4 describes the variational approximation derived for our model.

The parameters of the model should then be tuned to accurately describe the dynamics of the data. This learning procedure is complex. The model should not only learn the terms

of a SDE, but it should also learn a encoding transformation able to summarize the relevant dynamics, and a decoding transformation able to translate the codes from the encoder. Hence, it is reasonable to assume that we shall need a large amount of data. As noted in Section 3.8, variational inference requires a full-pass through the whole dataset at each iteration, which quickly becomes quite inefficient or even intractable for datasets that do not fit in main memory. A key advantage of the SVAE framework is that it naturally makes use of stochastic optimization methods to perform variational inference. This technique, known as Stochastic Variational Inference (SVI) [64], permits effectively scaling up variational inference to large datasets, since there is no need to keep the whole data in memory and because the updates of the parameters are more frequent. Section 4.5 describes the learning procedure used to tune the SVAE.

Finally, the derived SVAE is tested on synthetic data in Section 4.6, and some conclusions are given in Section 4.7.

Code implementing the methods discussed in this chapter is available at [46]. In this case, the code is not as mature as in other chapters and therefore it is not distributed as a package, but as a collection of scripts.

4.1 Background

Although we prefer to introduce concepts and tools as needed, this section provides a brief overview of the key ideas on which SVAEs are built. Albeit not directly used, an intuitive understanding of these concepts may help the reader to follow the exposition.

4.1.1 Variational autoencoders

As discussed in the introduction, the Variational Autoencoder (VAE) is a recent model that leverages generative graphical models and neural network autoencoders through approximate inference and gradient-based learning methods. Given a dataset $\{\boldsymbol{y}_n\}_{n=1}^N$, which may consist on pictures or text, the VAE models each observation $\boldsymbol{y}_n$ as if it was generated by some stochastic latent variable $\boldsymbol{x}_n$. To generate a sample from the model, the VAE draws a sample $\boldsymbol{x}^*$ from some prior distribution $p(\boldsymbol{x})$. Then, a new value $\boldsymbol{y}^*$ is generated from the conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x} = \boldsymbol{x}^*, \boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$. To easily generate samples, $p(\boldsymbol{x})$ is usually selected as a standardized Gaussian. On the other hand, to permit rich expressivity, the

conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$ is parametrized through a neural network:

$$\begin{aligned} \boldsymbol{x}_n &\sim \mathcal{N}(\boldsymbol{x}_n \mid \mathbf{0}, \boldsymbol{I}), \qquad \text{for } n = 1, 2, \ldots, N \\ \boldsymbol{y}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta} &\sim \mathcal{N}\Big(\boldsymbol{y}_n \mid \boldsymbol{\mu}(\boldsymbol{x}_n, \boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{x}_n, \boldsymbol{\theta})\Big), \end{aligned}$$

where $\boldsymbol{\mu}(\boldsymbol{x}_n, \boldsymbol{\theta})$ and $\boldsymbol{\Sigma}(\boldsymbol{x}_n, \boldsymbol{\theta})$ are the outputs of a neural network with weights $\boldsymbol{\theta}$ and input $\boldsymbol{x}_n$. For a shorthand notation, we shall use

$$\boldsymbol{\mu}(\boldsymbol{x}_n, \boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{x}_n, \boldsymbol{\theta}) \triangleq \text{NNet}(\boldsymbol{x}_n, \boldsymbol{\theta}).$$

The resulting generative model is shown in Figure 4.2a. Figure 4.2a follows the usual conventions of graphical models. Nodes correspond to random variables whereas edges represent statistical dependencies between these variables. Figure 4.2a is a directed graphical model (or Bayesian network) since all the edges are directed. In directed graphical models, an arrow represents a conditional distribution. The arrow starts at the conditioning variables and ends at the random variable of the distribution. Therefore, the arrows pointing from $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$ to $\boldsymbol{y}_n$ are representing the distribution $p(\boldsymbol{y}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})$. The box labelled with N in Figure 4.2a is called a plate, and indicates that the nodes within the box are repeated N times. Therefore, the graphical model is representing the distribution

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{x}_n) p(\boldsymbol{y}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}),$$

where we have noted $\boldsymbol{y} = \{\boldsymbol{y}_n\}_{n=1}^N$ and $\boldsymbol{x} = \{\boldsymbol{x}_n\}_{n=1}^N$.

Finally, observed variables are noted with a colored gray node (see $\boldsymbol{y}_n$) and deterministic variables will be denoted with small solid circles (not shown in Figure 4.2a). In an undirected graphical model, the edges have a slightly different meaning, which will be summarized when necessary.

Regarding the distribution induced by Figure 4.2a, we would like to approximate the posterior distribution $p(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y})$ after observing the data. However, exact inference is intractable and therefore, a variational approximation must be used

$$p(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y}) \approx q(\boldsymbol{\theta}) q(\boldsymbol{x} \mid \boldsymbol{y}) = q(\boldsymbol{\theta}) \prod_{n=1}^{N} q(\boldsymbol{x}_n \mid \boldsymbol{y}_n),$$

where $q(\cdot)$ denotes the variational approximations to the posterior distributions and therefore $q(\boldsymbol{x} \mid \boldsymbol{y}) \approx p(\boldsymbol{x} \mid \boldsymbol{y})$ and $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta} \mid \boldsymbol{y})$. This is illustrated in Figure 4.2b. Note that in

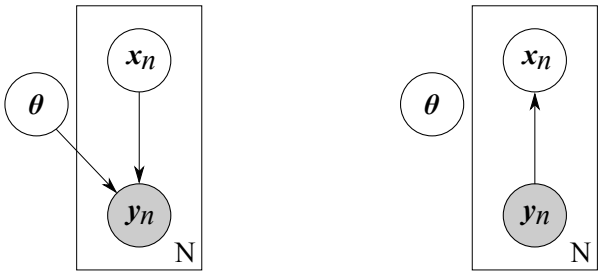

Figure 4.2: (a) VAE generative model and (b) VAE variational approximation.

variational inference $q(\boldsymbol{x} \mid \boldsymbol{y})$ is usually noted as $q(\boldsymbol{x})$, since the information provided by the data $\boldsymbol{y} = \{\boldsymbol{y}_n\}_{n=1}^N$ is absorbed by the parameters of the distribution. However, a key insight in the VAE is to permit the parameters of the conditional variational distribution $q(\boldsymbol{x}_n \mid \boldsymbol{y}_n)$ to depend on the corresponding data point $\boldsymbol{y}_n$. Note that, with this dependence, there are only local latent variables, i.e., each datapoint $\boldsymbol{y}_n$ has its own latent variable $\boldsymbol{x}_n$. This permits expressing the variational lower bound as a sum where each term depends on a single datapoint. As before, we can model $q(\boldsymbol{x}_n \mid \boldsymbol{y}_n)$ as a Gaussian distribution depending on a neural network with parameters $\boldsymbol{\eta}$,

$$q(\boldsymbol{x}_n \mid \boldsymbol{y}_n) = \mathcal{N}\Big(\boldsymbol{x}_n \mid \boldsymbol{\mu}(\boldsymbol{y}_n, \boldsymbol{\eta}), \boldsymbol{\Sigma}(\boldsymbol{y}_n, \boldsymbol{\eta})\Big),$$
$$\boldsymbol{\mu}(\boldsymbol{y}_n, \boldsymbol{\eta}), \boldsymbol{\Sigma}(\boldsymbol{y}_n, \boldsymbol{\eta}) = \text{NNet}(\boldsymbol{y}_n, \boldsymbol{\eta}).$$

Note that the variational distribution $q(\boldsymbol{x}_n \mid \boldsymbol{y}_n)$ takes on the role of a stochastic encoder, whereas the $p(\boldsymbol{y}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})$ distribution introduced by the generative model acts as a stochastic decoder.

4.1.2 Stochastic variational inference and natural gradients

Stochastic Variational Inference (SVI) [64] scales up variational inference for large datasets by using stochastic natural gradient ascent on a mean field variational approximation. Therefore, we first review natural gradients before summarizing the key aspects of SVI. We finally review how to use GP models within the SVI framework.

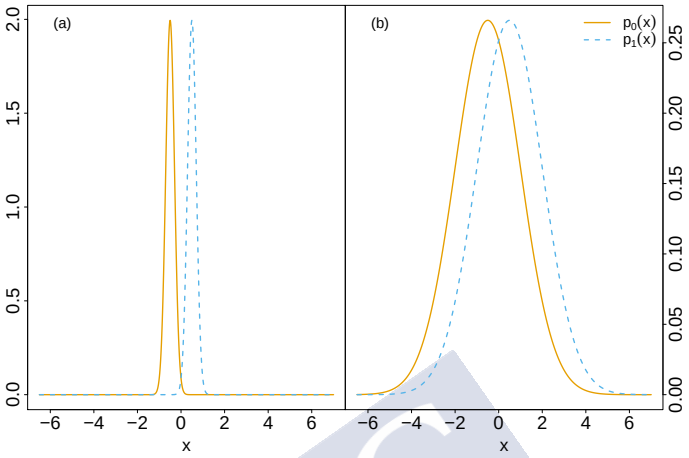

Figure 4.3: Two different settings where the Euclidean distance between p_0 and p_1 is 1. While the distributions from (a) (left panel) are very different, the distributions from (b) (right panel) are similar. To assess this, consider how difficult would it be to distinguish if some sample was drawn from p_0 or p_1. The distributions have the parameters (a) $p_0 \sim \mathcal{N}(-0.5, 0.2^2)$ and $p_1 \sim \mathcal{N}(0.5, 0.2^2)$, and (b) $p_0 \sim \mathcal{N}(-0.5, 1.5^2)$ and $p_1 \sim \mathcal{N}(0.5, 1.5^2)$.

Natural gradients

Optimization using gradient descent methods is straightforward. First, we compute the gradients of the objective function with respect to the optimizable parameters. These gradients represent the direction in which we should update the parameters to obtain the biggest change in the objective function. Hence, the parameters can be tuned by performing *small* updates in the direction of the gradient. However, the definition of *small* is problem dependent and the standard metric used by most algorithms, the Euclidean metric, may not be the most appropriate for measuring the distances between parameters. To illustrate this, let's consider two Gaussian distributions p_0 and p_1 with parameters (μ_0, σ_0^2) and (μ_1, σ_1^2). Intuitively, if the distance between both distributions is small, they should be quite similar. In our example, this intuition is not always correct when using the Euclidean distance, as illustrated in Figure 4.3.

From Figure 4.3, it is clear that the Euclidean distance between $[\mu_0, \sigma_0^2]^T$ and $[\mu_1, \sigma_1^2]^T$ fails to capture the structure of the objects that underlie these parameters. Therefore, to compare statistical objects, we must define some sort of statistical distance. For example, one

of the simplest ones is the symmetrized Kullback-Leibler divergence:

$$\mathcal{KL}_{symm}(p_0 \mid p_1) = \frac{1}{2}\Big(\mathcal{KL}(p_0 \mid p_1) + \mathcal{KL}(p_1 \mid p_0)\Big).$$

The use of the Kullback-Leibler divergence as a distance defines a "distribution space" with special geometrical properties. To study this properties, the mathematics of Riemannian geometry must be employed. In Riemannian geometry, the distance between two points $\boldsymbol{x}$ and $\boldsymbol{x} + d\boldsymbol{x}$ is defined in terms of the Riemannian metric tensor $\boldsymbol{G}(\boldsymbol{x})$:

$$\|d\boldsymbol{x}\|^2 = \sum_{ij} G_{ij}(\boldsymbol{x}) dx_i dx_j.$$

The Riemannian metric tensor defines the intrinsic curvature of a particular manifold in the geometrical space. In our distribution space, a manifold is a family of density functions $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$. The Riemannian metric of this manifold is given by the Fisher information matrix

$$\boldsymbol{F}_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{x}}\left[\Big(\nabla \log p(\boldsymbol{x} \mid \boldsymbol{\theta})\Big)\Big(\nabla \log p(\boldsymbol{x} \mid \boldsymbol{\theta})\Big)^T\right].$$

Given a cost function depending on $\boldsymbol{\theta}$, $\mathcal{L}(\boldsymbol{\theta})$, the natural gradient tries to move along the manifold of the distributions $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ by correcting the gradient of $\mathcal{L}(\boldsymbol{\theta})$ according to the curvature of the manifold itself [2],

$$\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta}) \triangleq \boldsymbol{F}_{\boldsymbol{\theta}}^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}),$$

where we have noted the natural gradient with $\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta})$.

As practical corollary, for gradient-based optimization of functions depending on statistical parameters, we can replace standard gradients with natural gradients. Furthermore, the natural gradient can be obtained from the standard gradient by multiplying it by the inverse of the Fisher information matrix.

Stochastic variational inference

SVI permits scaling up variational inference in graphical models of the form

$$p(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{y}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) p(\boldsymbol{x}_n \mid \boldsymbol{\theta}),$$

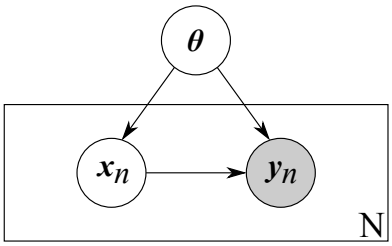

Figure 4.4: Probabilistic graphical model suited for SVI.

which includes global latent variables $\boldsymbol{\theta}$, local latent variables $\boldsymbol{x} = \{\boldsymbol{x}_n\}_{n=1}^N$, and the observed data $\boldsymbol{y} = \{\boldsymbol{y}_n\}_{n=1}^N$. The graphical representation of the model is given in Figure 4.4. Again, we would like to approximate the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{x} \mid \boldsymbol{y})$ with a mean field approximation

$$p(\boldsymbol{\theta}, \boldsymbol{x} \mid \boldsymbol{y}) \approx q(\boldsymbol{\theta})q(\boldsymbol{x}) = q(\boldsymbol{\theta}) \prod_{n=1}^{N} q(\boldsymbol{x}_n).$$

As discussed in Section 3.2, the variational approximation can be found by maximizing the marginal likelihood lower bound

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{p(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})}{q(\boldsymbol{\theta})q(\boldsymbol{x})} \right]. \tag{4.3}$$

SVI optimizes Equation (4.3) using stochastic natural gradients with respect to the global parameters $\boldsymbol{\theta}$. Furthermore, the natural gradients can be computed without calculating the Fisher matrix if $p(\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{\theta})$ is a member of the exponential family and $p(\boldsymbol{\theta})$ is its natural conjugate pair, also belonging to the exponential family [61, 64, 69]. Any variable $\boldsymbol{z}$ belonging to the exponential family may be written as

$$\log p(\boldsymbol{z}) = \boldsymbol{\eta}^T \boldsymbol{t}(\boldsymbol{z}) - \log Z(\boldsymbol{\eta}),$$

where $\boldsymbol{t}(\cdot)$ and $\log Z(\cdot)$ are two functions whose expressions depend on the specific distribution being considered (for example, whether it is a Gaussian or a Gamma distribution), and $\boldsymbol{\eta}$ is a vector of free variables parametrizing the distribution. These terms may seem arbitrary, but they play a fundamental role in determining the final form of the distribution. The variables from $\boldsymbol{\eta}$ are usually referred to as the natural parameters. The function $\boldsymbol{t}(\cdot)$ defines the sufficient statistics of the distribution. Therefore, for any dataset $\mathcal{X}$, it is possible to estimate the natural

parameters from $\boldsymbol{t}(\mathcal{X})$. $\log Z(\cdot)$ is referred to as the log-partition function since it is the logarithm of the normalization factor of the distribution. Furthermore, the moments of the sufficient statistics can be derived by differentiating the log-partition function [9].

Hence, if we assume that $p(\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$ are members of the exponential family, we may write them as

$$\log p(\boldsymbol{\theta}) = \boldsymbol{\eta}_\theta^T \boldsymbol{t}_\theta(\boldsymbol{\theta}) - \log Z_\theta(\boldsymbol{\eta}_\theta)$$
$$\log p(\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{\theta}) = \boldsymbol{\eta}_{xy}^T \boldsymbol{t}_{xy}(\boldsymbol{x}_n, \boldsymbol{y}_n) - \log Z_{xy}(\boldsymbol{\eta}_{xy}),$$

where we have used the subscripts (θ or xy) to distinguish the natural parameters, sufficient statistics and log-partition functions of both distributions. From the conjugacy assumption it follows that [9]

$$\boldsymbol{t}_\theta(\boldsymbol{\theta}) = [\boldsymbol{\eta}_{xy}(\boldsymbol{\theta}), -\log Z_{xy}(\boldsymbol{\eta}_{xy}(\boldsymbol{\theta}))]^T.$$

Furthermore, the conjugate exponential family structure implies that the optimal variational distribution has the same form as the prior, and therefore

$$\log q(\boldsymbol{\theta}) = \tilde{\boldsymbol{\eta}}_\theta^T \boldsymbol{t}_\theta(\boldsymbol{\theta}) - \log Z_\theta(\tilde{\boldsymbol{\eta}}_\theta),$$

for some variational parameter $\tilde{\boldsymbol{\eta}}_\theta$.

With these assumptions, the natural gradient of the lower bound $\mathcal{L}$ with respect to the natural parameters can be written in terms of local expected sufficient statistics, without the need of computing the Fisher matrix [64]:

$$\tilde{\nabla}_{\tilde{\boldsymbol{\eta}}_\theta} \mathcal{L} = \boldsymbol{\eta}_\theta - \tilde{\boldsymbol{\eta}}_\theta + \sum_{n=1}^{N} \mathbb{E}_{q^*(\boldsymbol{x}_n)} \Big[[\boldsymbol{t}_{xy}(\boldsymbol{x}_n, \boldsymbol{y}_n), 1]^T \Big],$$

where $q^*(\boldsymbol{x}_n)$ is a locally optimal mean field factor computed using standard variational inference given $\tilde{\boldsymbol{\eta}}_\theta$.

Stochastic variational inference for GPs

SVI can only be applied to probabilistic models that factorize as illustrated in Figure 4.4. However, not all graphical models follow the structure imposed by Figure 4.4. For example, consider the case of a GP regression problem: given $\{\boldsymbol{y}_n\}_{n=1}^N$ and $\{\boldsymbol{x}_n\}_{n=1}^N$, determine $f(\boldsymbol{x})$ according to the model

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n, \qquad n = 1, 2, \ldots, N$$

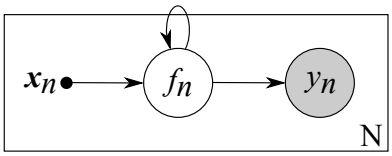

Figure 4.5: Graphical model of a GP regression problem. The loop represents the dependencies between the values of $f_n = f(\boldsymbol{x}_n)$ induced by the kernel of the GP. The $\{\boldsymbol{x}_n\}_{n=1}^N$ variables have been noted with a small solid circle since they are deterministic and known.

with

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k_f(\boldsymbol{x}, \boldsymbol{x}')) \qquad \text{and} \qquad \epsilon_n \sim \mathcal{N}(0, \sigma^2).$$

The resulting graphical model is shown in Figure 4.5. Due to the covariance structure imposed by the kernel of the GP, all the values of $f_n = f(\boldsymbol{x}_n)$ depend on each other. This prevents using the GP regression model within the SVI framework.

Fortunately, by introducing a set of inducing variables to approximate the GPs an appropriate model for using SVI can be obtained [61]. The key idea is that the inducing variables work as the global variables of the SVAE. The resulting graphical model, which is compatible with the use of SVI, is shown in Figure 4.6.

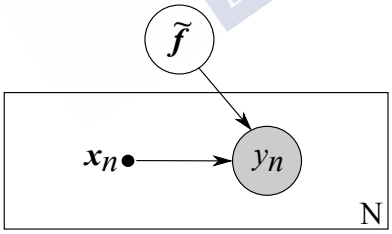

Figure 4.6: SGP graphical model with the inducing points $\tilde{\boldsymbol{f}}$ acting as global variables. This enables the use of SVI in the GP regression problem.

4.2 A generative model based on SDEs

In this section we develop a SVAE based on SDEs and the techniques presented in Chapter 3. We consider a collection of discrete-time signals obtained from sampling different realizations

of a continuous process with

$$\boldsymbol{y}_t^{(r)} = \boldsymbol{y}^{(r)}(t \cdot \Delta t), \qquad r = 1, 2, \ldots, R,$$
$$t = 1, 2, \ldots, N,$$

being Δt the sampling period, R the number of different realizations of the process, and N the total number of samples. To keep the notation uncluttered, we shall eliminate the superscript (r) from the formulas, as if $R = 1$, unless necessary. On the other hand, the different realizations will be represented in the graphical models for completeness. Note that we have preferred the "vector-like" notation $\boldsymbol{y}_t$ instead of $\boldsymbol{y}[t]$. This is because, in our exposition, we shall use distributions conditioned on several past values, which can be easily noted using the vector-like notation. For example, the conditional distribution $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}, \cdots \boldsymbol{y}_1) = p(\boldsymbol{y}[t] \mid \boldsymbol{y}[t-1], \boldsymbol{y}[t-2], \cdots \boldsymbol{y}[1])$ will be noted as $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1})$, which is significantly shorter. Following this notation, we will note the whole time series as $\boldsymbol{y}_{1:N}$.

We assume that the observations $\boldsymbol{y}_{1:N}$ are generated from a d-dimensional latent phase space $\boldsymbol{x}_{1:N}$. Furthermore, we also assume that this phase space can be modelled as the sampled version of a process $\boldsymbol{x}(t)$ described by a SDE (see Equation (4.2)). To simplify the phase space, we assume that the diffusion matrix from Equation (4.2) is diagonal and constant, i.e., g_{ii} does not depend on $\boldsymbol{x}(t)$:

$$dx_i(t) = f_i\big(\boldsymbol{x}(t)\big)dt + \sqrt{g_{ii}}\, dW_i(t) \qquad i = 1, 2, \ldots, d \tag{4.4}$$

The constant diffusion assumption may be unexpected given the attention paid to the non-constant diffusions in Chapter 3. This premise is guided by the existence of a transformation that is able to leave the diffusion term independent of the state $\boldsymbol{x}(t)$: the so called Lamperti transformation [96]. The general applicability of the Lamperti transformation is limited by the fact that it is constructed using the diffusion term. Consider, for example, a one-dimensional stochastic process given by

$$dx(t) = f\big(x(t)\big)dt + \sqrt{g\big(x(t)\big)}dW(t).$$

The Lamperti transformation is

$$z(t) = \int \frac{1}{\sqrt{g(x)}} dx,$$

which yields a new stochastic process with unit diffusion:

$$dz(t) = \tilde{f}(z(t))dt + dW(t).$$

When the diffusion term is not known, the Lamperti transformation cannot be applied. However, in our current approach we are building a new phase space $\boldsymbol{x}_{1:N}$ from scratch. Since, under quite general conditions, the Lamperti transformation guarantees that a representation with a state independent diffusion exists, we can select this representation as our phase space without loss of generality.

Equation (4.4) implies that the Euler-Maruyama discretization scheme (see Section 3.1) results in d equations for each sample $\boldsymbol{x}_t$:

$$\Delta x_{i,t} = f_i(\boldsymbol{x}_t)\Delta t + \sqrt{g_{ii}}(W^i_{t+1} - W^i_t), \qquad i = 1, 2, \ldots, d \tag{4.5}$$

where $x_{i,t}$ denotes the i-th dimension of the t-th sample and therefore $\Delta x_{i,t} = x_{i,t+1} - x_{i,t}$. Assuming again that Δt is small enough (see Section 3.1), the discrete transition probabilities can be approximated with a single multivariate Gaussian with diagonal covariance matrix:

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{f}, \boldsymbol{g}) = \mathcal{N}\left(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t + \boldsymbol{f}(\boldsymbol{x}_t)\Delta t, \mathrm{diag}(\boldsymbol{g})\Delta t\right), \tag{4.6}$$

where $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_d(\boldsymbol{x})]^T$ and $\boldsymbol{g} = [g_{11}, g_{22}, \ldots, g_{dd}]^T$.

Note that, since the phase space is not observed, both $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}$ are not known and must be learned. To enable rich dynamics, we parametrize both $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}$ with a flexible model, following Chapter 3. However, the assumption of constant diffusion does not only simplify the probability distribution of the transition probabilities (Equation (4.6)), but it also permits a simpler model for the diffusion term compared to the GP model developed in Chapter 3. Note that g_{ii} plays the role of the variance in Equation (4.6). Since the Normal-Gamma distribution is the conjugate prior of a Gaussian distribution with unknown mean and precision, we propose the use of a GP-Gamma distribution for modeling the drift and diffusion terms. We note

$$f(\boldsymbol{x}), \lambda \mid \boldsymbol{\theta} \sim \text{GP-Gamma}(f(\boldsymbol{x}), \lambda \mid m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'), \alpha, \beta; \boldsymbol{\theta})$$

implying that

$$\begin{aligned} \lambda &\sim \Gamma(\lambda \mid \text{shape} = \alpha, \text{rate} = \beta), \\ f(\boldsymbol{x})|\boldsymbol{\theta}, \lambda &\sim \mathcal{GP}\left(m(\boldsymbol{x}), \frac{1}{\lambda}k(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta})\right). \end{aligned}$$

Therefore, we model the drift and diffusion terms as

$$\begin{aligned} g_{ii} &= \frac{1}{\lambda_i}, \qquad i = 1, 2, \ldots, d, \\ f_i(\boldsymbol{x}), \lambda_i \mid \boldsymbol{\theta}_i &\sim \text{GP-Gamma}(f_i(\boldsymbol{x}), \lambda_i \mid 0, k_i(\boldsymbol{x}, \boldsymbol{x}'), \alpha_i, \beta_i; \boldsymbol{\theta}_i), \end{aligned} \tag{4.7}$$

where we have used a zero prior for the mean of f_i for the same symmetry reasons why we chose it in Section 3.1. For convenience, we shall denote $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_d]^T$.

Although in principle we could use any kernel k_i, some choices are better suited for deriving the learning procedure of the SVAE, since they permit to exactly integrate some expressions (see Section 4.5.2). Thus, in this chapter we use squared exponential covariance functions

$$k_i(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}_i) = A_i \exp\Big(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{\Lambda}_i^{-1} (\boldsymbol{x} - \boldsymbol{x}')\Big), \tag{4.8}$$

where $\boldsymbol{\Lambda}_i = \operatorname{diag}([(l_1^i)^2, (l_2^i)^2, \ldots, (l_d^i)^2])$ is a diagonal matrix of squared length-scales and $\boldsymbol{\theta}_i = [A_i, \operatorname{diag}(\boldsymbol{\Lambda}_i)]^T$.

It should be noted that the drift and diffusion terms appearing in Equation (4.6) include the Δt factor. However, this term can be easily absorbed by the parameters of the GP-Gamma distribution since

$$f(\boldsymbol{x})\Delta t, \lambda/\Delta t \sim \text{GP-Gamma}\big(f(\boldsymbol{x})\Delta t, \lambda/\Delta t \mid m(\boldsymbol{x})\Delta t, k(\boldsymbol{x}, \boldsymbol{x}')\Delta t, \alpha, \beta\Delta t; \boldsymbol{\theta}\big)$$

if

$$f(\boldsymbol{x}), \lambda \sim \text{GP-Gamma}\big(f(\boldsymbol{x}), \lambda \mid m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'), \alpha, \beta; \boldsymbol{\theta}\big).$$

Therefore, to simplify the notation we shall let the parameters of the GP-Gamma distribution absorb Δt, which shall subsequently be ignored in our exposition.

Each sample from the phase space $\boldsymbol{x}_t$ generates an observation $\boldsymbol{y}_t$. To enable an accurate modeling of complex time series, a Gaussian emission model parametrized by a feed-forward neural network can be used:

$$\begin{aligned} \boldsymbol{y}_t \mid \boldsymbol{x}_t, D &\sim \mathcal{N}\left(\boldsymbol{y}_t \mid \boldsymbol{\mu}(\boldsymbol{x}_t, D), \boldsymbol{\Sigma}(\boldsymbol{x}_t, D)\right), \\ \boldsymbol{\mu}(\boldsymbol{x}_t, D), \boldsymbol{\Sigma}(\boldsymbol{x}_t, D) &= \text{NNet}(\boldsymbol{x}_t, D). \end{aligned} \tag{4.9}$$

The parameters of the network, D, have been noted with a mnemonic for *D*ecoder. The generative model that results from these assumptions is shown in Figure 4.7.

SVAEs make use of SVI to handle large datasets and therefore, they can only be applied to graphical models which have a set of global variables. Figure 4.8 shows an example of a simple model consistent with the expected SVAE factorization.

Comparing Figures 4.7 and 4.8, it is immediately clear that our graphical model within the SVAE framework suffers from the same issue as the GP-regression model within the SVI framework: GPs do not have global variables and do not exhibit the expected factorization.

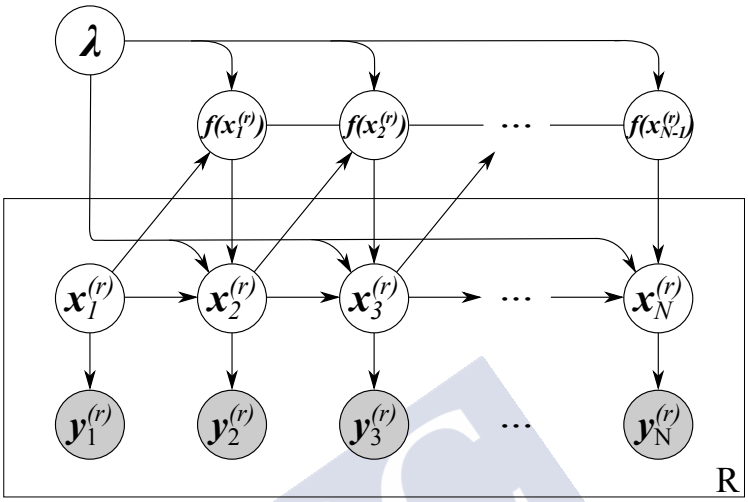

Figure 4.7: Generative graphical model for a phase space with SDE-dynamics.

Following [61], the key to tackle this issue is to introduce a set of inducing points in our model to act as global variables, as introduced in Section 4.1.2. This is convenient since it not only solves the SVI factorization problem, but it also connects with our formulation in Chapter 3.

4.3 Global variables for SVAE

As in Chapter 3, our inducing variables shall be the function points that result from evaluating $\boldsymbol{f}(\boldsymbol{x})$ at m pseudo-inputs $\tilde{\boldsymbol{X}} = \left\{\tilde{\boldsymbol{x}}_j : \tilde{\boldsymbol{x}}_j \in \mathbb{R}^d\right\}_{j=1}^{m}$. We have noted the inducing points with uppercase to highlight that they are a collection of points in $\mathbb{R}^d$, and because we can arrange them as a matrix with m rows and d columns. Although a more general treatment is possible, for the sake of simplicity we assume that the d GPs modeling the vector function $\boldsymbol{f}(\cdot)$ share the same set of pseudo-inputs. Since Equation (4.5) decomposes the equations of motion in d independent equations, we shall study each component of $\boldsymbol{f}(\cdot)$ independently. The i-th component of the inducing points shall be noted as $\tilde{\boldsymbol{f}}_i = \{f_i(\tilde{\boldsymbol{x}}_j) : \tilde{\boldsymbol{x}}_j \in \tilde{\boldsymbol{X}}\}$, and it is a vector of length m. From our previous assumptions, it follows that the i-th component of the inducing-points only depends on the values of the i-th GP. Furthermore, since $\tilde{\boldsymbol{f}}_i$ is derived

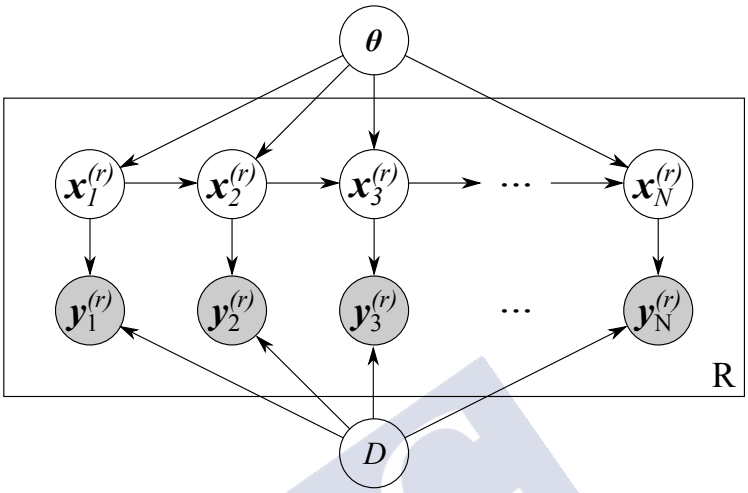

Figure 4.8: A simple model consistent with the expected SVAE factorization. The dynamics of the hidden space is parametrized with $\boldsymbol{\theta}$, whereas the observation model is defined by D.

from the same GP as $f_i(\cdot)$, the following conditional distribution holds:

$$\begin{bmatrix} f_i(\boldsymbol{x}) \\ \tilde{\boldsymbol{f}}_i \end{bmatrix} \mid \lambda_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \boldsymbol{0} \end{bmatrix}, \frac{1}{\lambda_i} \begin{bmatrix} \boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{x}) & \boldsymbol{K}_i(\boldsymbol{x},\tilde{\boldsymbol{X}}) \\ \boldsymbol{K}_i(\tilde{\boldsymbol{X}},\boldsymbol{x}) & \boldsymbol{K}_i(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}) \end{bmatrix} \right) \qquad i = 1, 2, \cdots, d,$$

where $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{x})$ is a 1×1 matrix, and the matrices $\boldsymbol{K}_i(\boldsymbol{x},\tilde{\boldsymbol{X}})$, $\boldsymbol{K}_i(\tilde{\boldsymbol{X}},\boldsymbol{x})$ and $\boldsymbol{K}_i(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}})$ should be read as the $1 \times m$, $m \times 1$ and $m \times m$ matrices that result from using Equation (3.6) on each of the points belonging to $\tilde{\boldsymbol{X}}$. Note that $f_i(\cdot)$ is a scalar function and it should not be confused with the i-th dimension of the inducing-points $\tilde{\boldsymbol{f}}_i$. The latter is the result of evaluating $f_i(\cdot)$ on a fixed set of points (the pseudo-inputs $\tilde{\boldsymbol{X}}$) and hence, it is possible to interpret it as a multivariate Gaussian random variable. The distribution of $f_i(\boldsymbol{x})$ conditioned on $\tilde{\boldsymbol{f}}_i$ and λ_i can be written as

$$\begin{aligned} & f_i(\boldsymbol{x}) \mid \tilde{\boldsymbol{f}}_i, \lambda_i \sim \mathcal{N}\Big(f_i(\boldsymbol{x}) \mid \boldsymbol{A}_i(\boldsymbol{x})\tilde{\boldsymbol{f}}_i, \boldsymbol{P}_i(\boldsymbol{x})/\lambda_i \Big) \qquad \text{with} \\ & \boldsymbol{A}_i(\boldsymbol{x}) = \boldsymbol{K}_i(\boldsymbol{x},\tilde{\boldsymbol{X}}) \cdot \boldsymbol{K}_i^{-1}(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}), \\ & \boldsymbol{P}_i(\boldsymbol{x}) = \boldsymbol{K}_i(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{K}_i(\boldsymbol{x},\tilde{\boldsymbol{X}}) \cdot \boldsymbol{K}_i^{-1}(\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}) \cdot \boldsymbol{K}_i(\tilde{\boldsymbol{X}},\boldsymbol{x}). \end{aligned} \tag{4.10}$$

Note that we have used a bold notation for $\boldsymbol{P}_i(\boldsymbol{x})$ because we compute its value through multiplications of matrices. However, the final $\boldsymbol{P}_i(\boldsymbol{x})$ from Equation (4.10) is a 1×1 matrix

and should be read as a scalar.

To "eliminate" the GP from the model shown in Figure 4.7, we need to replace the role of the conditional distribution involving the GP, $p(\boldsymbol{x}_{1:N} \mid \boldsymbol{f}(\cdot), \boldsymbol{\lambda}, \boldsymbol{\theta})$, by the conditional distribution involving the inducing-points, $p(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})$. Theoretically, the latter distribution could be obtained by marginalizing $\boldsymbol{f}(\cdot)$. However, in our setting $\boldsymbol{x}_{1:N}$ is latent, which hinders this option. Even in those cases where the marginalization is possible, the resulting expression has an unfeasible computational complexity [61]. Instead of computing the exact conditional distribution, we lower bound it by using Jensen's inequality:

$$\begin{aligned}\log p(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) &= \log \mathbb{E}_{p(\boldsymbol{f} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})} \left[p(\boldsymbol{x}_{1:N} \mid \boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{\theta}) \right] \\ &\geq \mathbb{E}_{p(\boldsymbol{f} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})} \log \left[p(\boldsymbol{x}_{1:N} \mid \boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{\theta}) \right] \\ &\triangleq \log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})\end{aligned}$$

It is possible to calculate $\log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})$ using calculations similar to those from Chapter 3, which yield

$$\begin{aligned}\log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \log p(\boldsymbol{x}_1) &+ \sum_{t=1}^{N-1} \sum_{i=1}^{d} \mathcal{N}\left(x_{i,t+1} \mid x_{i,t} + \boldsymbol{A}_i(\boldsymbol{x}_t) \tilde{\boldsymbol{f}}_i, \lambda_i^{-1} \right) \\ &- \frac{1}{2} \sum_{t=1}^{N-1} \sum_{i=1}^{d} \boldsymbol{P}_i(\boldsymbol{x}_t),\end{aligned}$$

where, just like in Equation (4.10), $\boldsymbol{P}_i(\boldsymbol{x}_t)$ should be read as a scalar. Just like in Chapter 3, we shall ignore the distribution of the initial point $p(\boldsymbol{x}_1)$, since its effect is negligible when N is large:

$$\log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) \approx \sum_{t=1}^{N-1} \sum_{i=1}^{d} \mathcal{N}\left(x_{i,t+1} \mid x_{i,t} + \boldsymbol{A}_i(\boldsymbol{x}_t) \tilde{\boldsymbol{f}}_i, \lambda_i^{-1} \right) - \frac{1}{2} \sum_{t=1}^{N-1} \sum_{i=1}^{d} \boldsymbol{P}_i(\boldsymbol{x}_t).$$

Following [61], we can use this lower bound in the SGP version of our original generative model as if it was $\log p(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})$, resulting in the graphical model shown in Figure 4.9. Note that the inducing variables work now as the global variables required by the SVAE approximation. Therefore, we shall consider the graphical model from Figure 4.9 as our complete generative model, instead of the one shown in Figure 4.7.

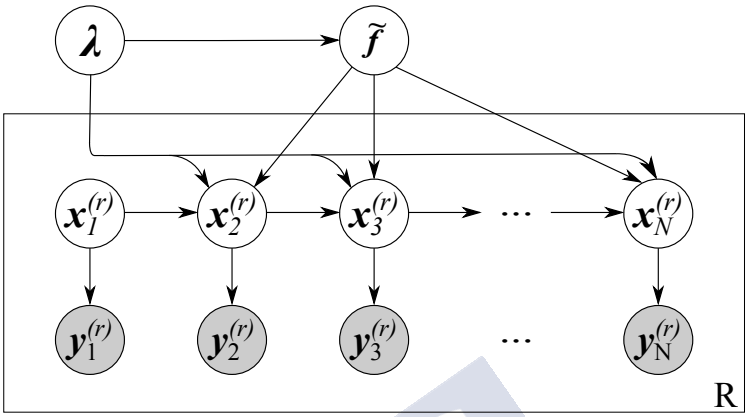

Figure 4.9: Generative model with inducing points.

4.4 Variational approximation of the SDE-based model

As usual with mean field variational inference, we start by breaking the posterior dependencies of the latent variables to achieve a tractable distribution:

$$p(\boldsymbol{x}_{1:N}, \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}, D \mid \boldsymbol{y}_{1:N}) \approx q(\boldsymbol{x}_{1:N}, \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}, D) = q(\boldsymbol{x}_{1:N}) q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) q(D). \tag{4.11}$$

We consider a simple mean field approximation that just factorizes the distributions distinguishing between the terms directly modeling the drift and diffusion, $\tilde{\boldsymbol{f}}$ and $\boldsymbol{\lambda}$, and other global variables $\boldsymbol{\theta}$:

$$q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}) q(\boldsymbol{\theta}) \approx p(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta} \mid \boldsymbol{y}_{1:N}). \tag{4.12}$$

Note that the variational distributions of $\tilde{\boldsymbol{f}}$ and $\boldsymbol{\lambda}$ may be affected by $\boldsymbol{\theta}$. Think for example, in the length-scale parameters of the Gaussian kernels. To simplify the variational approximation, we take the variational factor $q(\boldsymbol{\theta})$ to be a Dirac distribution $q(\boldsymbol{\theta}) = \delta_{\boldsymbol{\theta}^*}(\boldsymbol{\theta})$. Equation (4.12) becomes

$$q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) \delta_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}). \tag{4.13}$$

Since $p(\boldsymbol{f}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$ was chosen as a conjugate prior in Equation (4.7), the optimal $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$ is also in the same exponential family and therefore, it is the product of d multivariate Gaussian-

Gamma distributions:

$$q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) = \prod_{i=1}^{d} \mathcal{NG}(\tilde{\boldsymbol{f}}_i, \lambda_i \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \beta_i), \qquad \text{where}$$

$$\mathcal{NG}(\tilde{\boldsymbol{f}}_i, \lambda_i \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \beta_i) \triangleq \mathcal{N}(\tilde{\boldsymbol{f}}_i \mid \boldsymbol{\mu}_i, \lambda_i^{-1}\boldsymbol{\Sigma}_i) \cdot \Gamma(\lambda_i \mid \text{shape} = \alpha_i, \text{rate} = \beta_i).$$

Therefore, to find the variational approximation $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$ we only need to determine which are the optimal parameters of the Gaussian-Gamma distributions. The specific parametrization used for the Gaussian-Gamma distributions is further discussed in Section 4.5.3. For the moment, let us assume that each $\tilde{\boldsymbol{f}}_i$ has well-defined variational parameters $[\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \beta_i]^T$.

After studying the variational approach to the SDE model, we can plug in Equation (4.13) into Equation (4.11) to find the complete variational approximation:

$$\begin{aligned} p(\boldsymbol{x}_{1:N}, \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}, D \mid \boldsymbol{y}_{1:N}) &\approx q(\boldsymbol{x}_{1:N}, \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}, D) \\ &= q(\boldsymbol{x}_{1:N}) q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}) q(D) \\ &= q(\boldsymbol{x}_{1:N}) q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) \delta_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}) q(D). \end{aligned}$$

We can further simplify the variational approximation by assuming that the decoding parameters are singular $q(D) = \delta_{D^*}(D)$ and therefore:

$$p(\boldsymbol{x}_{1:N}, \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}, D \mid \boldsymbol{y}_{1:N}) \approx q(\boldsymbol{x}_{1:N}) q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) \delta_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}) \delta_{D^*}(D). \tag{4.14}$$

A key insight of the SVAE is that Equation (4.14) does not break the dependencies of $\boldsymbol{x}_{1:N}$ across time, which is intended to permit an accurate and flexible representation of the dynamics of phase space, whatever it may be [69]. In our case, since the phase space is modeled through a SDE, the optimal factor graph $\boldsymbol{x}_{1:N}$ is a Markov chain. In order to capture the relevant dynamical information from $\boldsymbol{y}_{1:N}$ an encoding network is used. This permits obtaining probabilistic guesses of the latent vector $\boldsymbol{x}_t$ from the observation $\boldsymbol{y}_t$. For the moment, we shall assume that such an encoding network exists, and we postpone the discussion of the details until Section 4.4.1.

To incorporate both the Markovian dynamics and the encoding information into the variational distribution $q(\boldsymbol{x}_{1:N})$, [69] proposes to parametrize it as a conditional random field. A conditional random field is a special undirected graphical model. The edges of a undirected graphical model represent statistical dependencies between the nodes (the random variables). However, unlike a directed graphical model, they do not represent a conditional distribution,

but rather the affinity between two variables. This affinity can be quantified by using a factor $\psi(\cdot)$, which is a non-negative function of the variables connected by an edge. That is, if we have an edge connecting variables A and B, these variables show some affinity; and this affinity can be measured using $\psi(A, B)$. Furthermore, in a conditional random field these factors can be conditioned on input (observed) features. Note that a factor is only a contribution to the overall joint distribution. Factors can also be associated with complete subgraphs, i.e., a subset of the graph such that there exist an edge between all pairs of nodes belonging to the subset. A complete subgraph is also referred to as clique. A maximal clique is a clique such that it is not possible to include any other node without it ceasing to be a clique. The joint distribution induced by the Markov field is written in term of the maximal cliques

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C} \psi_C(\boldsymbol{x}_C),$$

being $\boldsymbol{x}_C$ the set of variables that belong to the clique C.

In our case, the conditional Markov field can be written as

$$q(\boldsymbol{x}_{1:N}) \triangleq q(\boldsymbol{x}_{1:N}, E) \propto \prod_{t=1}^{N-1} \psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) \prod_{t=1}^{N} \psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E), \tag{4.15}$$

where the encoding potentials $\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)$ are Gaussian factors whose means and covariances depend on $\boldsymbol{y}_t$ through an encoder neural network:

$$\begin{aligned} &\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E) \propto \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{\mu}(\boldsymbol{y}_t, E), \boldsymbol{\Sigma}(\boldsymbol{y}_t, E)), \\ &\boldsymbol{\mu}(\boldsymbol{y}_t, E), \boldsymbol{\Sigma}(\boldsymbol{y}_t, E) = \mathrm{NNet}(\boldsymbol{y}_t, E). \end{aligned}$$

Note that the global variational distribution has been rewritten as $q(\boldsymbol{x}_{1:N}, E)$ to take into account the dependency on the encoder parameters. The resulting variational approximation is shown in Figure 4.10.

After selecting an approximation to the true posterior from some tractable family, variational inference tries to make this approximation as good as possible by reducing the inference procedure to an optimization problem in which the lower bound of the marginal log-likelihood $\mathcal{L}$ is maximized. Hence, the description of the variational approximation cannot be complete without giving the lower bound $\mathcal{L}$. Furthermore, to explicitly write the lower bound permits us to highlight the role of $\tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta})$ in our approximation (see Figure 4.9). We write $\mathcal{L}$

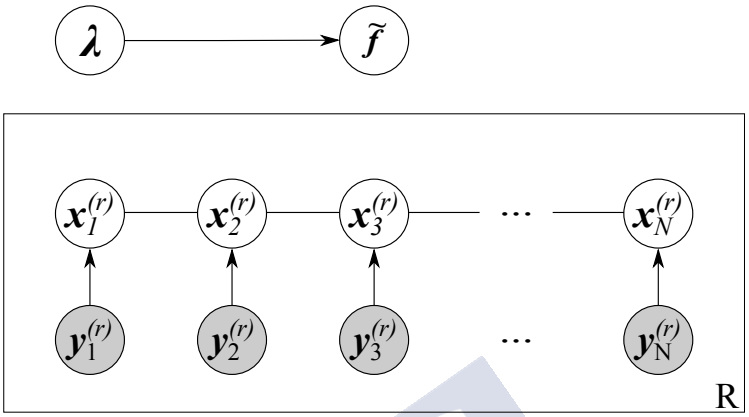

Figure 4.10: Variational approximation to the generative model.

as follows:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{x}_{1:N},E)q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log\left(\frac{\tilde{p}(y_{1:N},\boldsymbol{x}_{1:N},\tilde{\boldsymbol{f}},\boldsymbol{\lambda},\boldsymbol{\theta}^*,D^*)}{q(\boldsymbol{x}_{1:N},E)q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)}\right)\right] \\ &= \mathbb{E}_{q(\boldsymbol{x}_{1:N},E)q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log\left(\frac{p(y_{1:N}\mid\boldsymbol{x}_{1:N},D^*)\tilde{p}(\boldsymbol{x}_{1:N}\mid\tilde{\boldsymbol{f}},\boldsymbol{\lambda},\boldsymbol{\theta}^*)p(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)}{q(\boldsymbol{x}_{1:N},E)q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)}\right)\right],\end{aligned} \tag{4.16}$$

Note that the lower bound $\mathcal{L}$ makes use of the approximate conditional distribution $\tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}},\boldsymbol{\lambda},\boldsymbol{\theta}^*)$ to follow the graphical model shown in Figure 4.9.

Equation (4.16) can be expanded as follows:

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\log p(y_{1:N}\mid\boldsymbol{x}_{1:N},D^*)\right]+\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\mathbb{E}_{q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log\tilde{p}(\boldsymbol{x}_{1:N}\mid\tilde{\boldsymbol{f}},\boldsymbol{\lambda},\boldsymbol{\theta}^*)\right]\right] \\ &-\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\log q(\boldsymbol{x}_{1:N},E)\right]+\mathbb{E}_{q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log\frac{p(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)}{q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)}\right] \\ =&\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\log p(y_{1:N}\mid\boldsymbol{x}_{1:N},D^*)\right]+\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\mathbb{E}_{q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log\tilde{p}(\boldsymbol{x}_{1:N}\mid\tilde{\boldsymbol{f}},\boldsymbol{\lambda},\boldsymbol{\theta}^*)\right]\right] \\ &+\mathbb{H}\big(q(\boldsymbol{x}_{1:N},E)\big)-\mathcal{KL}\Big(q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)\mid p(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}\mid\boldsymbol{\theta}^*)\Big).\end{aligned} \tag{4.17}$$

For a better understanding of how the lower bound leads to a proper reconstruction of $\boldsymbol{y}_{1:N}$, we briefly discuss each of the terms of $\mathcal{L}$ from Equation (4.17). Using Equation (4.9), the first

term is just

$$\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\log p(y_{1:N} \mid \boldsymbol{x}_{1:N}, D^*)\right] = \mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\sum_{t=1}^{N} \log \mathcal{N}\left(\boldsymbol{y}_t \mid \boldsymbol{\mu}(\boldsymbol{x}_t, D^*), \boldsymbol{\Sigma}(\boldsymbol{x}_t, D^*)\right)\right],$$

which is basically measuring the expected reconstruction error (using a squared error) when decoding the phase space to predict $\boldsymbol{y}_{1:N}$. Therefore, the first term from Equation (4.17) is encouraging the decoder to fit as close as possible $\boldsymbol{y}_{1:N}$, given a variational distribution of the phase space, $\boldsymbol{x}_{1:N}$. This term may also influence the phase space itself. For instance, it may induce the separation of two points that lie close in the phase space, say $\boldsymbol{x}_{t_1}$ and $\boldsymbol{x}_{t_2}$, if they are decoded into two samples $\boldsymbol{y}_{t_1}$ and $\boldsymbol{y}_{t_2}$ with very different values.

To gain a deeper insight of the second term from Equation (4.17) we expand the expression inside the first expectation as follows:

$$\begin{aligned}\mathbb{E}_{q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}^*)\right] &= \sum_{t=1}^{N-1}\sum_{i=1}^{d} \log \mathcal{N}\left(x_{i,t+1} \mid x_{i,t} + \boldsymbol{A}_i(\boldsymbol{x}_t)\boldsymbol{\mu}_i, \beta_i/\alpha_i\right) \\ &\quad - \frac{1}{2}\sum_{t=1}^{N-1}\sum_{i=1}^{d}\left[\boldsymbol{P}_i(\boldsymbol{x}_t) + \operatorname{tr}\left(\boldsymbol{A}_i(\boldsymbol{x}_t)\boldsymbol{\Sigma}_i\boldsymbol{A}_i^T(\boldsymbol{x}_t)\right)\right] \\ &\quad + \frac{N-1}{2}\sum_{i=1}^{d}\left[\psi^0(\alpha_i) - \log(\alpha_i)\right], \end{aligned} \tag{4.18}$$

where $\psi^0(\cdot)$ is the digamma function.

The first term from Equation (4.18) is responsible for adjusting $\{\boldsymbol{A}_i(\boldsymbol{x}_t)\}_{i=1}^d$ and $\{\boldsymbol{\mu}_i\}_{i=1}^d$ so that $x_{i,t+1}$ can be accurately predicted using $\boldsymbol{x}_t$. Furthermore, it also fits $\{\beta_i/\alpha_i\}_{i=1}^d$ to the variance of the residuals that result from these predictions. The second term from Equation (4.18) is more difficult to interpret. The terms $\{\boldsymbol{P}_i(\boldsymbol{x}_t)\}_{i=1}^d$ come from Equation (4.10), and represent the uncertainty about the predictions of $f_i(\boldsymbol{x}_t)$ using the inducing-points $\tilde{\boldsymbol{f}}$. Hence, Equation (4.18) encourages the minimization of these predictive uncertainties, which can be achieved by either tuning the hyperparameters of the kernel or changing the phase space (Note that $\{\boldsymbol{P}_i(\boldsymbol{x}_t)\}_{i=1}^d$ depends on $\boldsymbol{x}_t$). The term involving $\left\{\operatorname{tr}\left(\boldsymbol{A}_i(\boldsymbol{x}_t)\boldsymbol{\Sigma}_i\boldsymbol{A}_i^T(\boldsymbol{x}_t)\right)\right\}_{i=1}^d$ cannot be interpreted in isolation, since it would yield $\boldsymbol{\Sigma}_i \to \boldsymbol{0}$. This does not occur due to the Kullback-Leibler divergence from Equation (4.17), which tries to keep each of the $\boldsymbol{\Sigma}_i$ as close as possible to its prior value, say $\boldsymbol{\Sigma}_i^{(0)}$. Therefore, the maximization of $\mathcal{L}$ requires a compromise, which (usually) results in values of $\boldsymbol{\Sigma}_i$ comprised between $\boldsymbol{0}$ and $\boldsymbol{\Sigma}_i^{(0)}$. The third term from Equation (4.18) favours large values of α, since $\psi(\alpha_i) - \log(\alpha_i)$ is strictly increasing

for $\alpha_i > 0$. This is consistent with the update rule of the Gaussian-Gamma posterior for Gaussians [97], where the α parameter is increased by $n/2$ after the observation of n samples. In our setting, this could be problematic, since we shall repeatedly update the parameters while learning the distributions. Again, the solution $\alpha_i \to \infty$ is avoided by means of the Kullback-Leibler divergence from Equation (4.17).

We have highlighted the key influence of the Kullback-Leibler divergence for the parameters $\{\boldsymbol{\Sigma}_i\}_{i=1}^d$ and $\{\alpha_i\}_{i=1}^d$, but it also has an effect on the parameters $\{\boldsymbol{\mu}_i\}_{i=1}^d$ and $\{\beta_i\}_{i=1}^d$, forcing a compromise between fitting the data and the prior distribution.

The last term from Equation (4.17) to be discussed is the entropy term (third term). This term helps preventing the optimization of $\mathcal{L}$ from simply collapsing the whole phase space to a random walk with small variance. In this case, the second term from Equation (4.17) can be maximized by letting $\boldsymbol{\mu}_i \to \mathbf{0}$ and $\beta_i/\alpha_i \to 0$ in Equation (4.18), although the Kullback-Leibler term does not permit the latter to be arbitrary small. Note that $\boldsymbol{\mu}_i \to \mathbf{0}$ matches the drift's prior, and implies that $\boldsymbol{x}_t$ does not provide information for predicting $\boldsymbol{x}_{t+1}$, i.e., a random walk. The entropy term precludes the collapse, since it favours a phase space that is as spread as possible. Furthermore, the first term from Equation (4.17) also penalizes the random walk collapse, since it would result in a low log-likelihood due to the inability of the phase space to provide valuable information to reproduce $\boldsymbol{y}_{1:N}$. Therefore, this term favours phase spaces having some structure ($\boldsymbol{\mu}_i \neq \mathbf{0}$), which enables the prediction of $\boldsymbol{y}_{1:N}$ from its trajectories.

4.4.1 The encoding network

We would like to obtain an encoding network able to discover the state vector $\boldsymbol{x}_t$ from $\boldsymbol{y}_t$. However, it is unrealistic to assume that this encoder will be able to output a good latent vector from a single measurement $\boldsymbol{y}_t$. What $\boldsymbol{y}_t$ lacks is the "context" of the time series (i.e., its previous samples), from which it would be possible to extract the relevant dynamics.

Since we assume that $\boldsymbol{y}_{1:N}$ has been generated by some high-dimensional stationary dynamical system, we could build a delay embedding:

$$\boldsymbol{Y}_t = \left[\boldsymbol{y}_t^T, \boldsymbol{y}_{t-\tau}^T, , \ldots, \boldsymbol{y}_{t-(l-1)\tau}^T\right]^T \qquad \text{for } t = 1 + (l-1)\tau, 2 + (l-1)\tau, \ldots, N,$$

where the vectors $\boldsymbol{y}_t$, $\boldsymbol{y}_{t-\tau}$, ..., are concatenated together in a single vector, and where τ and l should be selected in such a way that the embedding contains all the information required for determining the future of the time series. In an ideal scenario, we could set $\tau = 1$ and $l \to \infty$.

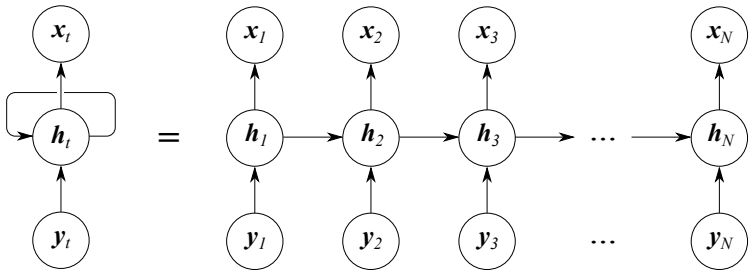

Figure 4.11: Schematic representation of a Recurrent Neural Network (RNN) (left) and its unrolled version for N timesteps (right).

What this embedding implies is that, for predicting the next sample of the sequences, all the past samples of the time series should be used. Hence, this new phase space is redundant and provides no insight about the dynamics of the system, although it does contain the information required for predicting the next step of the original time series. Therefore, we could feed a Multilayer Perceptron (MLP) with $\boldsymbol{Y}_t$ (instead of $\boldsymbol{y}_t$) to calculate the phase state at time t.

A more elegant solution to this issue, that we shall adopt for our proposal, is using a Recurrent Neural Network (RNN) as encoder [57]. RNNs are a family of neural networks specialized in processing sequential data. They are called recurrent because they have feedback loops that permit each output to depend on all previous computations. A schematic representation of a RNN is shown in Figure 4.11. In theory, these loops enable RNNs to connect previous relevant information with the present task, therefore implementing some sort of memory. In our case, this would permit sequentially processing $\boldsymbol{y}_t$, outputting a new latent state $\boldsymbol{x}_t$ after reading each sample while still taking into account the previous relevant dynamic information from the previous time steps. Note that the hidden state $\boldsymbol{h}_t$, as named in the RNN literature, should not be confused with our latent state vector $\boldsymbol{x}_t$, which is the one that will be forced to follow SDE-like dynamics.

In practice, RNNs may struggle with learning long-term dependencies, when the gap between the past relevant information and the point where it is needed becomes large [103]. For an intuitive understanding of this fact, consider a simple RNN whose hidden state $\boldsymbol{h}_{t+1}$ just depends on the previous hidden state $\boldsymbol{h}_t$, completely ignoring the input $\boldsymbol{y}_{t+1}$:

$$\boldsymbol{h}_{t+1} = \boldsymbol{F}(\boldsymbol{h}_t, \boldsymbol{y}_t, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{h}_t.$$

After k time steps, the resulting state $\boldsymbol{h}_{t+k}$ depends on $\boldsymbol{h}_t$ through the matrix $\boldsymbol{W}^k$. If $\boldsymbol{W}$ has the eigendecomposition $\boldsymbol{W} = \boldsymbol{V}\text{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$, it follows that

$$\boldsymbol{W}^k = \boldsymbol{V}\text{diag}(\boldsymbol{\lambda})^k\boldsymbol{V}^{-1}.$$

When k is large, the eigenvalue λ_i will either explode if its absolute value is greater than 1, or vanish if its absolute value is smaller than 1. The exploding and vanishing gradient problems are due to the fact that gradients scale according to $\text{diag}(\boldsymbol{\lambda})^k$ when backpropagating. Vanishing gradients hamper optimizing the RNN through gradient descent, since they are so small that it is difficult to know in which direction the network weights should move. On the other hand, exploding gradients result in large updates of the parameters, which can make learning unstable.

There are several approaches that may help with vanishing and exploding gradients ranging from simple tricks to fundamental architectural changes in the RNN design. Among the latter, the Long Short-Term Memory (LSTM) [63] and the Gated Recurrent Unit (GRU) [21] architectures have become widely used. Although we shall adopt the GRU architecture for our experiments, the method proposed in this chapter is independent of the specific RNN architecture. Therefore any other RNN design could be used.

Among the tricks that facilitate learning long-term correlations, gradient clipping and Truncated Backpropagation Through Time (TBPTT) stand out. Gradient clipping prevents gradients from blowing up by thresholding them before applying gradient descent. TBPTT refers to the truncation of the backpropagation of errors to a maximum of N time steps. It is worth noting that the problem of backpropagating gradients through time will also appear in our model due to the Markovian dynamics imposed in the latent space.

In practice, TBPTT is usually implemented by splitting the original sequence into subsequences, and only backpropagating on the latter. We briefly review how this affects the organization of our data, which in Section 4.2 was assumed to consist of R realizations of a time series of a specific length which we now note as N', to discuss the case where the length of the data does not match the length of TBPTT, N. That is, our data is $\{\boldsymbol{y}_{1:N'}^{(r)}\}_{r=1}^{R}$. However, we can illustrate how TBPTT affects the organization of data considering the case $R = 1$.

We shall organize the time series $\boldsymbol{y}_{1:N'}$ in shorter ones, which are then subsequently split in segments and grouped to form several batches. Figure 4.12 illustrates how we could arrange a single time series of length 36: $\boldsymbol{y}_{1:36}$. The original time series is split in three shorter ones of length 12, which are identified by a superscript. These time series are then split in segments

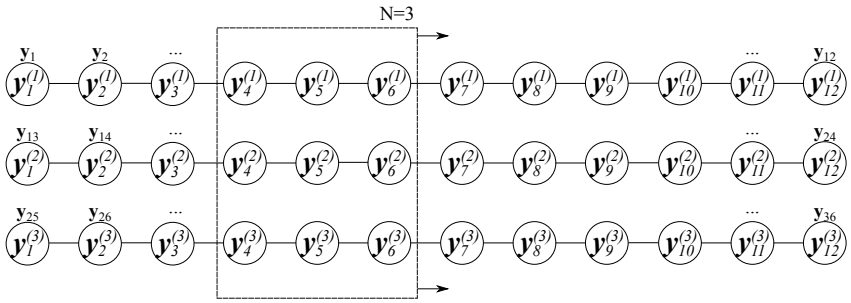

Figure 4.12: Organization of the training data for a time series $\boldsymbol{y}_{1:36}$. The original samples of the time series are noted over the circles whereas the final training data is noted inside them. The time series $\boldsymbol{y}_{1:36}$ is split into three shorter time series so that the batch size is $R = 3$. The length of the shorter time series should be large enough so that they are almost uncorrelated. TBPTT is then implemented by studying the series in small windows of length $N = 3$. A batch size $R > 1$ provides diversity in the examples for the learning procedure.

of length $N = 3$, which truncate the backpropagation of errors. During training, a single batch of data consisting of $R = 3$ segments of length $N = 3$ is used to update the weights of the model. An epoch of the training procedure comprises a full pass over the data, which in the example from Figure 4.12 consists of 4 batches. Note that the use of a batch size larger than one (splitting the original time series rather than using it as a whole) provides more diversity in examples in a single step of the training procedure. A similar arrangement could be used if we had three time series of length 12 ($R = 3$) instead of a single time series of length 36.

Having stressed the importance of the data organization to implement TBPTT, we now focus on the actual learning phase. Although we briefly recovered the notation from Section 4.2, in the following sections we shall drop again the superscript from the formulas to keep the notation uncluttered. Therefore, we will work with the time series $\boldsymbol{y}_{1:N}$, as if we were using $R = 1$ and $N = N'$.

4.5 Learning phase

In this section, we explain the learning procedure and derive the stochastic gradients required to learn the dynamics of our graphical model and the encoding-decoding transformations. These stochastic gradients can be used to learn the variational distributions by means of a stochastic optimization routine such as gradient descent, Adam [75] or Adadelta [138].

The computation of the gradients require to derive the lower bounds from Equation (4.17). However, the expectation from the term $\mathbb{E}_{q(\boldsymbol{x}_{1:N},E)}\left[\log p(y_{1:N} \mid \boldsymbol{x}_{1:N}, D^*)\right]$ is impossible to compute. The dependency on $\boldsymbol{x}_{1:N}$ comes from the fact that the mean and covariance of $p(y_{1:N} \mid \boldsymbol{x}_{1:N}, D^*)$ are the output of the decoding network, whose inputs are precisely $\boldsymbol{x}_{1:N}$. Computing the expectation through the neural network is unfeasible. Furthermore, although theoretically possible, the computation of other terms involving expectations with respect to $\boldsymbol{x}_{1:N}$ are also difficult.

To tackle this issue, a Monte Carlo approximation shall be used. This requires being able to generate samples from the phase space $\boldsymbol{x}_{1:N}^{(s)}$ where $\boldsymbol{x}_{1:N}^{(s)} \sim q(\boldsymbol{x}_{1:N}, E) \approx p(\boldsymbol{x}_{1:N} \mid \boldsymbol{y}_{1:N})$. The superscript (s) is a number that identifies one of the S samples used for the Monte Carlo approximation. Note that to be able to differentiate through the Monte Carlo approximation, the samples should keep the dependencies with the parameters of $q(\boldsymbol{x}_{1:N}, E)$, i.e., E. This can be achieved through the reparametrization trick [76]. Hence, we shall use the notation for the samples as $\boldsymbol{x}_{1:N}^{(s)}(E)$. The Monte Carlo approximation reads

$$\begin{aligned}\mathcal{L} \approx & \frac{1}{S}\sum_{s=1}^{S} \log p\big(y_{1:N} \mid \boldsymbol{x}_{1:N}^{(s)}(E), D^*\big) \\ & + \frac{1}{S}\sum_{s=1}^{S} \mathbb{E}_{q(\tilde{\boldsymbol{f}},\boldsymbol{\lambda}|\boldsymbol{\theta}^*)}\left[\log \tilde{p}\big(\boldsymbol{x}_{1:N}^{(s)}(E) \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}^*\big)\right] \\ & + \mathbb{H}\big(q(\boldsymbol{x}_{1:N}, E)\big) - \mathcal{KL}\Big(q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) \mid p(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)\Big).\end{aligned} \tag{4.19}$$

Having stressed the need of generating samples from the phase space, a complete overview of the algorithm can be provided. The algorithm for computing the necessary gradients is summarized, for the case where $R = 1$, in Algorithm 1 [69]. The algorithm receives as main argument the input data $\boldsymbol{y}_{1:N}$. The SVAE tries to transform this time series into a phase space representation that captures the main dynamical features of the signal. To do that, it combines information from the encoder, represented with $\{\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)\}_{t=1}^{N}$ in Algorithm 1, with the "constraints" $\{\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\}_{t=1}^{N-1}$ imposed by the fact that the phase space should be consistent with a system of SDEs. These constraints have to be updated each time the COMPUTE_GRADIENTS function is called, since they depend on the variational distribution of the SDE, $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$, which is subject of optimization through the variational parameters $\tilde{\boldsymbol{\eta}}$. It must be noted that the $\tilde{\boldsymbol{\eta}}$ parameters are not selected to be natural parameters of $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$ (see Section 4.5.3). Once the information from the encoder $\{\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)\}_{t=1}^{N}$ and the dynamical

information $\{\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\}_{t=1}^{N-1}$ are available, they are used to draw samples from the phase space. This can be efficiently done using the so called forward-backward algorithm, grounded on the Kalman filtering literature [70] (see Section 4.5.2). In a nutshell, the forward-backward algorithm performs a forward pass through the Markov chain defining the embedding space to collect all the information from $\{\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)\}_{t=1}^{N}$ and $\{\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\}_{t=1}^{N-1}$. Then, it uses this information in a backward pass to generate samples of $\boldsymbol{x}_{1:N}^{(s)}$. Furthermore, it is also possible to use the backward pass to compute the entropy of $\boldsymbol{x}_{1:N}$, $\mathbb{H}\big(q(\boldsymbol{x}_{1:N}, E)\big)$. The samples are used to feed the decoder, which outputs several estimates of the original time series (one for each sample). The quality of the reconstructions and how well the SDEs describe the phase space are evaluated with the lower bound term $\mathcal{L}$. To improve it through stochastic gradient ascent, its gradients with respect to the parameters D^*, E and $\boldsymbol{\theta}^*$, and its natural gradient with respect to $\tilde{\boldsymbol{\eta}}$ are computed. Note that, since the $\tilde{\boldsymbol{\eta}}$ are not natural parameters of the variational distribution $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$, Algorithm 1 has to explicitly compute the Fisher information matrix to calculate the natural gradients. The reasons for this are detailed in Section 4.5.3.

The remainder of the section provides detailed information about the steps of Algorithm 1 that we have just sketched.

4.5.1 Inference of the variational factors

To compute the variational objectives from Equation (4.19) and to draw samples from the phase space, we should first infer the optimal variational factors $q(\boldsymbol{x}_{1:N}, E)$. As introduced in Section 3.2, the optimal variational distribution $q(\cdot)$ in a local mean field approximation can be found solving $\partial\mathcal{L}(q)/\partial q = 0$. The general expression for the solution of this equation requires taking expectations with respect to the other variational factors [11]. In our case, finding the variational factor $q(\boldsymbol{x}_{1:N}, E)$ requires fixing the SDE parameters $\tilde{\boldsymbol{f}}$ and $\boldsymbol{\lambda}$, the hyperparameters $\boldsymbol{\theta}$ and the encoding potentials $\{\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)\}_{t=1}^{N}$ to compute the expectation:

$$\log q(\boldsymbol{x}_{1:N}, E) \propto \mathbb{E}_{q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)} \left[\log p(y_{1:N} \mid \boldsymbol{x}_{1:N}, D^*) + \log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}^*)\right],$$

which, as a result of our hypothesis about $q(\boldsymbol{x}_{1:N}, E)$ in Equation (4.15), implies that

$$\log q(\boldsymbol{x}_{1:N}, E) \propto \mathbb{E}_{q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)} \left[\log \tilde{p}(\boldsymbol{x}_{1:N} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}^*)\right]. \tag{4.20}$$

The expectation in Equation (4.20) results in the conditional Gaussian factors

$$\begin{aligned} &\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \mathcal{N}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t + \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t), \operatorname{diag}(\boldsymbol{\beta}/\boldsymbol{\alpha})\big) \qquad \text{with} \\ &\hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) = [\boldsymbol{A}_1(\boldsymbol{x}_t)\boldsymbol{\mu}_1, \boldsymbol{A}_2(\boldsymbol{x}_t)\boldsymbol{\mu}_2, \ldots, \boldsymbol{A}_d(\boldsymbol{x}_t)\boldsymbol{\mu}_d]^T, \end{aligned} \tag{4.21}$$

Algorithm 1 SVAE algorithm for SDE-based dynamics.

INPUT: A time series of observations $\boldsymbol{y}_{1:N}$; the current variational parameters $\tilde{\boldsymbol{\eta}}$ and the hyperparameters $\boldsymbol{\theta}$ characterizing the distribution $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}^*)$; and the current weights of the encoder and the decoder, E and D^*.

function COMPUTE_GRADIENTS($\boldsymbol{y}_{1:N}$, $\tilde{\boldsymbol{\eta}}$, $\boldsymbol{\theta}^*$, D^*, E)
  ▷ Get the encoding potentials using the encoding network
  $\{\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)\}_{t=1}^{N} \leftarrow \text{NNET}(\boldsymbol{y}_{1:N}, E)$
  $(\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \beta_i\})_{i=1}^{d}$ = UNPACK_GAUSSIAN_GAMMA_PARAMS($\tilde{\boldsymbol{\eta}}$)
  ▷ Update the dynamic potentials
  for t in $1, 2, \ldots, N-1$ **do**
    $\hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \leftarrow [\boldsymbol{A}_1(\boldsymbol{x}_t)\boldsymbol{\mu}_1, \boldsymbol{A}_2(\boldsymbol{x}_t)\boldsymbol{\mu}_2, \ldots, \boldsymbol{A}_d(\boldsymbol{x}_t)\boldsymbol{\mu}_d]^T$
    $\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) \leftarrow \mathcal{N}\Big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t + \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t), \text{diag}(\boldsymbol{\beta}/\boldsymbol{\alpha})\Big)$ ▷ Equation (4.21)
  end for
  ▷ Draw S samples from $q(\boldsymbol{x}_{1:N}, E)$ and compute its entropy $\mathbb{H}\big(q(\boldsymbol{x}_{1:N}, E)\big)$
  ▷ Use Equations (4.36) and (4.37).
  $(\hat{\boldsymbol{x}}_{1:N}^{(1)}, \ldots, \hat{\boldsymbol{x}}_{1:N}^{(S)}), \mathbb{H}\big(q(\boldsymbol{x}_{1:N}, E)\big) \leftarrow \Big($
    FORWARD_BACKWARD($\{\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\}_{t=1}^{N-1}, \{\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)\}_{t=1}^{N}$)
  $\Big)$
  ▷ The likelihood of the real observations $\boldsymbol{y}_{1:N}$ is determined by the decoding network
  for s in $1, 2, \ldots, S$ **do**
    $\log p\big(\boldsymbol{y}_{1:N} \mid \hat{\boldsymbol{x}}_{1:N}^{(s)}, D^*\big) \leftarrow \log \mathcal{N}(\boldsymbol{y}_{1:N} \mid \text{NNET}(\hat{\boldsymbol{x}}_{1:N}^{(s)}, D^*)\,)$
  end for
  ▷ Use Equation (4.19)
  $\mathcal{L} \leftarrow \frac{1}{S}\sum_{s=1}^{S} \log p\big(y_{1:N} \mid \boldsymbol{x}_{1:N}^{(s)}, D^*\big) + \frac{1}{S}\sum_{s=1}^{S} \mathbb{E}_{q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)}\Big[\log \tilde{p}\big(\boldsymbol{x}_{1:N}^{(s)} \mid \tilde{\boldsymbol{f}}, \boldsymbol{\lambda}, \boldsymbol{\theta}^*\big)\Big]$
    $+\mathbb{H}\big(q(\boldsymbol{x}_{1:N}, E)\big) - \mathcal{KL}\big(q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) \mid p(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)\big)$
  ▷ The Fisher matrix is necessary to compute the natural gradients of $\tilde{\boldsymbol{\eta}}$
  $\boldsymbol{F}(\tilde{\boldsymbol{\eta}}) \leftarrow \text{FISHER\_MATRIX}(\tilde{\boldsymbol{\eta}})$ ▷ Equation (4.39)
  return $\nabla_{D^*, E, \boldsymbol{\theta}^*}\mathcal{L}$, $\boldsymbol{F}(\tilde{\boldsymbol{\eta}})^{-1}\nabla_{\tilde{\boldsymbol{\eta}}}\mathcal{L}$,
end function

where the $\boldsymbol{A}_i$ has been defined in Equation (4.10), $\boldsymbol{\mu}_i$ is the i-th mean of the i-th Gaussian-Gamma distribution from $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$, and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the result of stacking the $\{\alpha_i\}_{i=1}^{d}$ and $\{\beta_i\}_{i=1}^{d}$ parameters from this same distribution. According to Equation (4.21), we approximate the dynamics of the system as if it evolved following

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \boldsymbol{w}_t, \\ \boldsymbol{w}_t &\sim \mathcal{N}\big(\boldsymbol{0}, \text{diag}(\boldsymbol{\beta}/\boldsymbol{\alpha})\big), \end{aligned} \tag{4.22}$$

where $\boldsymbol{w}_t$ plays the role of a random noise.

Next section details how we can draw samples from the variational distribution $\boldsymbol{x}_{1:N}$.

4.5.2 Sampling from the phase space

Our variational model is almost analogous to many state space models. Therefore, it is possible to reuse common methodologies from the dynamical systems literature to try to address the problem of drawing samples of $\boldsymbol{x}_{1:N}$. To make the exposition clearer and closer to the literature, during this section we will explicitly note the variational distribution $q(\boldsymbol{x}_t)$ as $q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:N})$, highlighting the fact that $q(\boldsymbol{x}_t) \approx p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:N})$.

Filtering and smoothing are key concepts in state space models. Filtering refers to the computation of the belief state $q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$, i.e., computing the distribution of $\boldsymbol{x}_t$ given all the evidence available until that moment. Smoothing means to compute $q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:N})$, given all the evidence. Using these concepts, it is also possible to draw samples from our phase space. The key insight to efficiently draw samples from the Markov chain is to perform a forward filtering pass collecting information about the observations from the past; and then perform sampling in the backward pass, using at each node the conditional node potentials and combining them with the information from the past of the node [98, Section 17.4.3].

However, closed form solutions of the filtering and smoothing problems are only available for a few special cases. For example, the well-known Kalman filter provides the exact distributions for linear Gaussian systems [70].

Many different approximate methods have been suggested for nonlinear cases without exact solutions. In GP dynamical systems, where the transition dynamics are governed by Gaussian Processes (GPs), several approximations based on moment matching have been proposed [19, 26, 27]. The key idea is to approximate the predictive distributions of the dynamic model with a Gaussian distribution which has the very same mean and covariance matrix as the true predictive distribution.

To perform filtering and sampling in our graphical model we resort to message passing algorithms. In this sort of algorithms, inference is expressed in terms of the propagation of local messages throughout the graph [11]. Specifically, if we convert our graphical model into an undirected graph (by dropping the direction of the arrows in Figure 4.10), we see that the resulting graph is a tree. Therefore, we will use the generalization of message passing algorithms to trees, the so-called sum-product algorithm [11]. Figure 4.13 illustrates the messages that node $\boldsymbol{x}_2$ needs to receive in order to compute its marginal distribution. Intuitively, messages $m_{1\to 2}(\boldsymbol{x}_2)$ and $m_{3\to 2}(\boldsymbol{x}_2)$ summarize the relevant information about the past and the

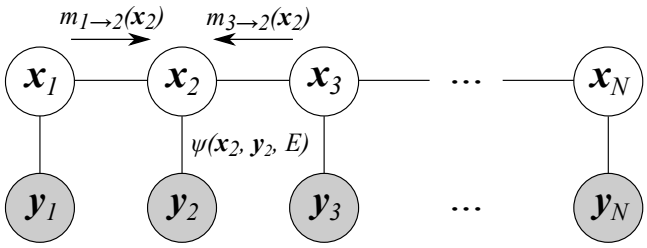

Figure 4.13: A fragment of the message passing in the graphical model illustrating the messages that involve the node $\boldsymbol{x}_2$. The information from the nodes is combined with the node potential $\psi(\boldsymbol{x}_2, \boldsymbol{y}_2, E)$ to yield the marginal posterior distribution of $\boldsymbol{x}_2$.

future, respectively. This information is combined with the node potential $\psi(\boldsymbol{x}_2, \boldsymbol{y}_2, E)$ to compute the proper posterior. Note that the node potential $\psi(\boldsymbol{x}_2, \boldsymbol{y}_2, E)$ provides the code learned by the encoding network.

The messages are defined as follows:

$$m_{i\to j}(\boldsymbol{x}_j) = \int \psi(\boldsymbol{x}_i, \boldsymbol{x}_j)\psi(\boldsymbol{x}_i, \boldsymbol{y}_i, E)m_{k\to i}(\boldsymbol{x}_i)d\boldsymbol{x}_i,$$

where k denotes the other neighbor of node i (apart from j).

Forward pass

In this section, we focus on the forward pass and therefore the messages can be written as

$$m_{t\to t+1}(\boldsymbol{x}_{t+1}) = \int \psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)m_{t-1\to t}(\boldsymbol{x}_t)d\boldsymbol{x}_t.$$

Instead of focusing in the message $m_{t\to t+1}(\boldsymbol{x}_{t+1})$, we study a slightly different type of message,

$$\alpha(\boldsymbol{x}_t) = m_{t-1\to t}(\boldsymbol{x}_t)\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E), \tag{4.23}$$

since it allows to connect our presentation with the forward pass of HMMs and Kalman filters. The new messages can also be computed recursively:

$$\alpha(\boldsymbol{x}_{t+1}) = \psi(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}, E)\int \psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t. \tag{4.24}$$

The formulation of the message passing algorithm in terms of $\alpha(\boldsymbol{x}_t)$ is advantageous due to its interpretation. From its definition in Equation (4.23), it is clear that $\alpha(\boldsymbol{x}_t)$ combines the information from the past with the code from the encoding network, and therefore

$$\alpha(\boldsymbol{x}_t) \propto q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}).$$

Taking this into account, we can interpret Equation (4.24) as follows. First, we predict the following step using the current available information:

$$q(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t}) \propto \int \psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t. \tag{4.25}$$

Then, we measure the observed $\boldsymbol{y}_{t+1}$ and we consequently update the belief state

$$q(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t+1}) \propto \alpha(\boldsymbol{x}_{t+1}) = \underbrace{\psi(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}, E) \underbrace{\int \psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t}_{\text{Predict}}}_{\text{Update}} \tag{4.26}$$

This process is known as the predict-update cycle in the literature of HMMs and Kalman filters, which permits to compute the messages in closed form. However, the integral from Equation (4.26) cannot be computed exactly, due to the nonlinear terms from $\psi(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})$ (see Equation (4.20)). To tackle the issue, we use moment matching to compute a Gaussian approximation to the predicted distribution $q(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t})$ [19, 26, 27]. We proceed by induction, assuming that the message $\alpha(\boldsymbol{x}_t)$ is Gaussian. That is:

$$q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) \propto \alpha(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{\mu}^x_{t|t}, \boldsymbol{\Sigma}^x_{t|t}), \tag{4.27}$$

where we have used the standard notation of Kalman filters to denote the parameters of the Gaussian. In general, $\boldsymbol{\mu}^x_{t|\tau}$ denotes the mean of $\boldsymbol{x}_t$ conditioned on the observation of $\boldsymbol{y}_{1:\tau}$. In Equation (4.27) we have the particular case $\tau = t$. This notation also applies to the covariance parameter.

The mean of Equation (4.25) is

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}_{t+1}}\left[\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t}\right] &= \mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{w}_t}\left[\boldsymbol{x}_t + \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \boldsymbol{w}_t \mid \boldsymbol{y}_{1:t}\right] \\
&= \mathbb{E}_{\boldsymbol{x}_t}\left[\boldsymbol{x}_t + \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\right] \\
&= \boldsymbol{\mu}_{t|t} + \mathbb{E}_{\boldsymbol{x}_t}\left[\hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\right],
\end{aligned} \tag{4.28}$$

where we have plugged in the variational dynamics described by Equation (4.22), we have used that $\boldsymbol{w}_t$ has zero mean, and we have substituted $\mathbb{E}_{\boldsymbol{x}_t}[\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}]$ by its definition in Equation (4.27). The a-th component of the remaining expectation can be expanded as follows:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}_t}\left[\hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\right]_a &= \int \boldsymbol{K}_a(\boldsymbol{x}_t, \tilde{\boldsymbol{X}})\boldsymbol{K}_a^{-1}(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}})\boldsymbol{\mu}_a\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t \\ &= \int \sum_i \boldsymbol{K}_a(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_i)\boldsymbol{\beta}_i^a(\tilde{\boldsymbol{X}})\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t \\ &= \sum_i \boldsymbol{\beta}_i^a(\tilde{\boldsymbol{X}}) \int \boldsymbol{K}_a(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_i)\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t,\end{aligned} \tag{4.29}$$

where $\boldsymbol{\beta}^a(\tilde{\boldsymbol{X}}) = \boldsymbol{K}_a^{-1}(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}})\boldsymbol{\mu}_a$. For the squared exponential kernel, the integral has a closed form solution [26]. Therefore, Equation (4.29) can be written as

$$\mathbb{E}_{\boldsymbol{x}_t}\left[\hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\right]_a = \boldsymbol{\beta}^a(\tilde{\boldsymbol{X}})^T\boldsymbol{\gamma}^a(\tilde{\boldsymbol{X}}),$$

with $\boldsymbol{\gamma}^a(\tilde{\boldsymbol{X}}) = [\gamma_1^a(\tilde{\boldsymbol{X}}), \gamma_2^a(\tilde{\boldsymbol{X}}), \dots \gamma_m^a(\tilde{\boldsymbol{X}})]^T$, where

$$\begin{aligned}\gamma_i^a(\tilde{\boldsymbol{X}}) = \int \boldsymbol{K}_a(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_i)\alpha(\boldsymbol{x}_t)d\boldsymbol{x}_t = &A_a \mid \boldsymbol{\Sigma}_{t|t}^x\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{I} \mid^{-1/2} \times \\ &\exp\Big(-\frac{1}{2}(\boldsymbol{\mu}_{t|t} - \tilde{\boldsymbol{x}}_i)^T(\boldsymbol{\Sigma}_{t|t}^x + \boldsymbol{\Lambda}_a)^{-1}(\boldsymbol{\mu}_{t|t} - \tilde{\boldsymbol{x}}_i)\Big),\end{aligned}$$

and where $\boldsymbol{\Lambda}_a$ and A_a are parameters of the squared exponential kernel, as defined in Equation (4.8).

We also have to match the other moment defining a Gaussian distribution: the covariance. Using the law of total covariance, the required covariance can be expressed as

$$\begin{aligned}\mathrm{Cov}\Big[(\boldsymbol{x}_{t+1})_a, (\boldsymbol{x}_{t+1})_b \mid \boldsymbol{y}_{1:t}\Big] = &\mathbb{E}_{\boldsymbol{x}_{t-1}}\Big[\mathrm{Cov}[(\boldsymbol{x}_{t+1})_a, (\boldsymbol{x}_{t+1})_b \mid \boldsymbol{x}_t] \,\Big|\, \boldsymbol{y}_{1:t}\Big] \\ &+ \mathrm{Cov}\Big[\mathbb{E}[(\boldsymbol{x}_{t+1})_a \mid \boldsymbol{x}_t], \mathbb{E}[(\boldsymbol{x}_{t+1})_b \mid \boldsymbol{x}_t] \,\Big|\, \boldsymbol{y}_{1:t}\Big].\end{aligned} \tag{4.30}$$

Substituting the variational dynamics (Equation (4.22)) into Equation (4.30) and rearranging

results in

$$\begin{aligned}\mathrm{Cov}\Big[(\boldsymbol{x}_{t+1})_a,(\boldsymbol{x}_{t+1})_b \mid \boldsymbol{y}_{1:t}\Big] =&\delta_{ab}\frac{\beta_a}{\alpha_a}+\mathrm{Cov}\Big[(\boldsymbol{x}_t)_a,(\boldsymbol{x}_t)_b \;\Big|\; \boldsymbol{y}_{1:t}\Big]\\ &+\mathrm{Cov}\Big[(\boldsymbol{x}_t)_a,\hat{\mu}_b(\boldsymbol{x}_t) \;\Big|\; \boldsymbol{y}_{1:t}\Big]+\mathrm{Cov}\Big[\hat{\mu}_a(\boldsymbol{x}_t),(\boldsymbol{x}_t)_b \;\Big|\; \boldsymbol{y}_{1:t}\Big]\\ &+\mathrm{Cov}\Big[\hat{\mu}_a(\boldsymbol{x}_t),\hat{\mu}_b(\boldsymbol{x}_t) \;\Big|\; \boldsymbol{y}_{1:t}\Big]\\ =&\delta_{ab}\frac{\beta_a}{\alpha_a}+(\boldsymbol{\Sigma}^x_{t\,|t})_{a,b}\\ &+\mathrm{Cov}\Big[(\boldsymbol{x}_t)_a,\hat{\mu}_b(\boldsymbol{x}_t) \;\Big|\; \boldsymbol{y}_{1:t}\Big]+\mathrm{Cov}\Big[\hat{\mu}_a(\boldsymbol{x}_t),(\boldsymbol{x}_t)_b \;\Big|\; \boldsymbol{y}_{1:t}\Big]\\ &+\mathrm{Cov}\Big[\hat{\mu}_a(\boldsymbol{x}_t),\hat{\mu}_b(\boldsymbol{x}_t) \;\Big|\; \boldsymbol{y}_{1:t}\Big],\end{aligned} \tag{4.31}$$

where we have used that $\mathrm{Cov}[(\boldsymbol{w}_t)_a,(\boldsymbol{w}_t)_b]=\delta_{ab}\frac{\beta_a}{\alpha_a}$. Solving Equation (4.31) requires finding $\mathbb{E}\left[(\boldsymbol{x}_t)_a,\hat{\mu}_b(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\right]$ and $\mathbb{E}\left[\hat{\mu}_a(\boldsymbol{x}_t),\hat{\mu}_b(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\right]$.

Using similar manipulations to those used in Equation (4.28), we find that

$$\mathbb{E}\Big[(\boldsymbol{x}_t)_a,\hat{\mu}_b(\boldsymbol{x}_t) \;\Big|\; \boldsymbol{y}_{1:t}\Big]=\sum_i \boldsymbol{\beta}_i^b(\tilde{\boldsymbol{X}})\boldsymbol{\gamma}_i^b(\tilde{\boldsymbol{X}})\boldsymbol{\Gamma}_a^{i,b,t}, \tag{4.32}$$

being

$$\boldsymbol{\Gamma}^{i,b,t}=\Big(\boldsymbol{\Lambda}_b^{-1}+(\boldsymbol{\Sigma}^x_{t\,|t})^{-1}\Big)^{-1}\Big(\boldsymbol{\Lambda}_b^{-1}\tilde{\boldsymbol{x}}_i+(\boldsymbol{\Sigma}^x_{t\,|t})^{-1}\boldsymbol{\mu}_{t\,|t}\Big).$$

Finally, and following [27], we find

$$\mathbb{E}\Big[\hat{\mu}_a(\boldsymbol{x}_t),\hat{\mu}_b(\boldsymbol{x}_t) \;\Big|\; \boldsymbol{y}_{1:t}\Big]=\boldsymbol{\beta}^a(\tilde{\boldsymbol{X}})\boldsymbol{Q}^{a,b}\boldsymbol{\beta}^b(\tilde{\boldsymbol{X}}), \tag{4.33}$$

where the entries of $\boldsymbol{Q}^{a,b}$ are given by

$$\begin{aligned}Q_{ij}^{a,b} =&A_aA_b \mid (\boldsymbol{\Lambda}_a^{-1}+\boldsymbol{\Lambda}_b^{-1})(\boldsymbol{\Sigma}^x_{t\,|t})^{-1}+\boldsymbol{I}\mid^{-1/2}\\ &\times\exp\Big(-\frac{1}{2}(\tilde{\boldsymbol{x}}_{\boldsymbol{i}}-\tilde{\boldsymbol{x}}_j)^T(\boldsymbol{\Lambda}_a+\boldsymbol{\Lambda}_b)^{-1}(\tilde{\boldsymbol{x}}_{\boldsymbol{i}}-\tilde{\boldsymbol{x}}_j)\Big)\\ &\times\exp\Big(-\frac{1}{2}(\tilde{\boldsymbol{z}}_{\boldsymbol{ij}}-\boldsymbol{\mu}^x_{t\,|t})^T((\boldsymbol{\Lambda}_a^{-1}+\boldsymbol{\Lambda}_b^{-1})^{-1}+\boldsymbol{\Sigma}^x_{t\,|t})^{-1}(\tilde{\boldsymbol{z}}_{\boldsymbol{ij}}-\boldsymbol{\mu}^x_{t\,|t})\Big)\end{aligned}$$

with

$$\tilde{\boldsymbol{z}}_{\boldsymbol{ij}}=\boldsymbol{\Lambda}_b(\boldsymbol{\Lambda}_a+\boldsymbol{\Lambda}_b)^{-1}\tilde{\boldsymbol{x}}_i+\boldsymbol{\Lambda}_a(\boldsymbol{\Lambda}_a+\boldsymbol{\Lambda}_b)^{-1}\tilde{\boldsymbol{x}}_j.$$

Using Equations (4.32) and (4.33), the covariance given by Equation (4.31) is completely determined. Therefore, the moment matching step required to approximate the predicted distribution $q(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t})$ (Equation (4.25)) with a Gaussian is complete.

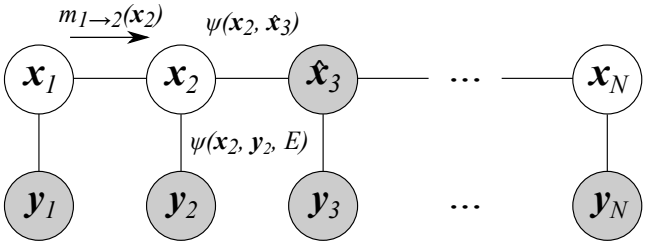

Figure 4.14: Backward sampling in our graphical model. We have noted $\hat{\boldsymbol{x}}_3$ to highlight that it has been observed. Note that the measurement of $\boldsymbol{x}_N$ (or any other node $\boldsymbol{x}_t$, with $t > 3$) does not affect the situation, since the observation of $\hat{\boldsymbol{x}}_3$ makes the graphs on its left and right conditionally independent [11, Chapter 8].

The update step from Equation (4.26) is straightforward, since $\psi(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}, E)$ is Gaussian and the integral has been approximated by another Gaussian factor as a result of the moment matching. Therefore, we can use the multiplication of Gaussians formula (see, for example, [106, Equation (371)]) to compute the final message $\alpha(\boldsymbol{x}_{t+1})$:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = \mathcal{N}(\boldsymbol{\mu}_1 \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \qquad \text{with}$$
$$\boldsymbol{\Sigma}_c = \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}$$
$$\boldsymbol{\mu}_c = \boldsymbol{\Sigma}_c\left(\Sigma_1^{-1}\boldsymbol{\mu}_1 + \Sigma_2^{-1}\boldsymbol{\mu}_2\right).$$

Backward sampling

After computing the filtered distributions $q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ in the forward pass, we can sample from $q(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:N})$ in a backward pass. We can start by sampling $\boldsymbol{x}_N$, since $q(\boldsymbol{x}_N \mid \boldsymbol{y}_{1:N}) \propto \alpha(\boldsymbol{x}_N)$, and then proceed to sample $\boldsymbol{x}_{N-1}$ given that its neighbor from the future is known. Figure 4.14 illustrates the messages and node potentials involved in determining the distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{1:N})$ (for the case $t = 2$, although the context is generic). Note that the product of $m_{t-1\to t}(\boldsymbol{x}_t)$ and $\psi(\boldsymbol{x}_t, \boldsymbol{y}_t, E)$ is the message $\alpha(\boldsymbol{x}_t)$, which can be calculated as described in Section 4.5.2. The node potential $\psi(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_{t+1})$ is also known, but it is not a valid Gaussian term on $\boldsymbol{x}_t$, which prevents us to directly compute $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{1:N})$ as

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{1:N}) \propto \alpha(\boldsymbol{x}_t)\psi(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_{t+1}).$$

A Gaussian approximation to the $\psi(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_{t+1})$ node potential could be obtained by linearizing Equation (4.22) and solving for $\boldsymbol{x}_t$. Another approach which does not require linearization was proposed in [27]. The key insight is to compute a Gaussian approximation to the conditional $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}, \boldsymbol{y}_t)$ using the moment matching techniques from Section 4.5.2. This conditional distribution is computed in two steps. First, a Gaussian approximation to the joint distribution $q(\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \mid \boldsymbol{y}_t)$ is computed. Second, the well-known rules of computing conditionals from a joint Gaussian distribution are applied.

The approximate joint distribution reads

$$q(\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \mid \boldsymbol{y}_t) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}^x_{t|t} \\ \boldsymbol{\mu}^x_{t+1|t} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}^x_{t|t} & \boldsymbol{\Sigma}^x_{t,t+1|t} \\ (\boldsymbol{\Sigma}^x_{t,t+1|t})^T & \boldsymbol{\Sigma}^x_{t+1|t} \end{bmatrix}\right). \tag{4.34}$$

The components $\boldsymbol{\mu}^x_{t|t}$ and $\boldsymbol{\Sigma}^x_{t|t}$ are the mean and covariance of the Gaussian approximation to $\alpha(\boldsymbol{x}_t)$ and therefore are known from the forward pass. On the other hand, the components $\boldsymbol{\mu}^x_{t+1|t}$ and $\boldsymbol{\Sigma}^x_{t+1|t}$ are also known, since they are computed during the prediction step. Hence, the only unknown component from Equation (4.34) is $\boldsymbol{\Sigma}^x_{t,t+1|t}$. The cross-covariance can be determined in term of expectations:

$$\boldsymbol{\Sigma}^x_{t,t+1|t} = \mathbb{E}\Big[\boldsymbol{x}_t \boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t}\Big] - \boldsymbol{\mu}^x_{t|t}(\boldsymbol{\mu}^x_{t+1|t})^T. \tag{4.35}$$

Substituting Equation (4.22) into Equation (4.35), we find that

$$\begin{aligned}\boldsymbol{\Sigma}^x_{t,t+1|t} &= \mathbb{E}\Big[\boldsymbol{x}_t(\boldsymbol{x}_t + \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) + \boldsymbol{w}_t) \mid \boldsymbol{y}_{1:t}\Big] - \boldsymbol{\mu}^x_{t|t}\boldsymbol{\mu}^x_{t+1|t} \\ &= \boldsymbol{\Sigma}^x_{t|t} + \boldsymbol{\mu}^x_{t|t}(\boldsymbol{\mu}^x_{t|t})^T + \mathbb{E}\Big[\boldsymbol{x}_t \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\Big] - \boldsymbol{\mu}^x_{t|t}(\boldsymbol{\mu}^x_{t+1|t})^T,\end{aligned}$$

which concludes the calculation, since the expression for $\mathbb{E}\Big[\boldsymbol{x}_t \hat{\boldsymbol{\mu}}(\boldsymbol{x}_t) \mid \boldsymbol{y}_{1:t}\Big]$ is given by Equation (4.32).

From the joint distribution, it is straightforward to compute the conditional distribution $q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t+1}, \boldsymbol{y}_t) = \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{m}_t, \boldsymbol{C}_t)$, with

$$\begin{aligned}\boldsymbol{m}_t &= \boldsymbol{\mu}^x_{t|t} + \boldsymbol{\Sigma}^x_{t,t+1|t}\left(\boldsymbol{\Sigma}^x_{t+1|t}\right)^{-1}\left(\boldsymbol{x}_{t+1} - \boldsymbol{\mu}^x_{t+1|t}\right) \\ \boldsymbol{C}_t &= \boldsymbol{\Sigma}^x_{t|t} - \boldsymbol{\Sigma}^x_{t,t+1|t}\left(\boldsymbol{\Sigma}^x_{t+1|t}\right)^{-1}\left(\boldsymbol{\Sigma}^x_{t,t+1|t}\right)^T,\end{aligned} \tag{4.36}$$

from which we can sample $\boldsymbol{x}_t$ given $\boldsymbol{x}_{t+1}$. By chaining these samples in the backward pass, a path in the embedding space $\hat{\boldsymbol{x}}^{(s)}_{1:N}$ is obtained.

It must be noted that the backward and forward passes can also be used to compute an approximation to the entropy of $q(\boldsymbol{x}_{1:N}, E)$, that appears in the lower bound expression in Equation (4.17). The entropy of a d-dimensional multivariate Gaussian distribution is

$$\mathbb{H}\Big(\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})\Big) = \frac{d}{2}\log(2\pi e) + \frac{1}{2}\log \mid \boldsymbol{C} \mid = \frac{d}{2} + \log Z(\boldsymbol{C}),$$

where $\log Z(\boldsymbol{C})$ is the partition function of $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$. It can be demonstrated that the partition function of the Markov chain $\log Z$ is just the sum of the partition functions $\{Z_t\}_{t=1}^{N}$ that normalize $q(\boldsymbol{x}_1 \mid \boldsymbol{y}_{1:N})$ and the conditional distributions $\{q(\boldsymbol{x}_t \mid \hat{\boldsymbol{x}}_{t+1}, \boldsymbol{y}_{1:N})\}_{t=2}^{T}$ in each node:

$$\log Z(\boldsymbol{C}) = \sum_{t=1}^{N} \log Z_t(\boldsymbol{C}_t), \tag{4.37}$$

where $\boldsymbol{C}_t$ denotes the covariance matrix of either $q(\boldsymbol{x}_1 \mid \boldsymbol{y}_{1:N})$ or some conditional Gaussian distribution $q(\boldsymbol{x}_t \mid \hat{\boldsymbol{x}}_{t+1}, \boldsymbol{y}_{1:N})$ from Equation (4.36). Therefore, the global $\log Z$ partition function can be easily computed during the backward pass of the message passing algorithm.

4.5.3 Parameter optimization

After sampling the phase space, a Monte Carlo approximation to the lower bound $\mathcal{L}$ (see Equation (4.19)) can be computed. The gradients of the neural networks and the hyperparameters $\boldsymbol{\theta}$ can be calculated using automatic differentiation tools, such as Theano [127], TensorFlow [125] or PyTorch [104]. In our implementation, we have used TensorFlow.

Instead of using standard gradients, the SVAE algorithm exploits the exponential family conjugacy structure to efficiently compute natural gradients with respect to the variational parameters without the need to calculate the Fisher matrix. Regarding the Gaussian-Gamma distributions $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$, the natural parameters of the i-th Gaussian-Gamma are

$$\boldsymbol{\eta}^{(i)} = \left[\eta_0^{(i)}, \eta_1^{(i)}, \boldsymbol{\eta}_2^{(i)}, \boldsymbol{\eta}_3^{(i)}\right]^T = \left[\alpha_i - \frac{1}{2}, -\beta_i - \frac{\boldsymbol{\mu}_i^T \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i}{2}, \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i, \mathrm{Vec}\left(-\frac{\boldsymbol{\Sigma}_i^{-1}}{2}\right)\right]^T,$$

where $\mathrm{Vec}(\cdot)$ vectorizes the matrix $-\frac{\boldsymbol{\Sigma}_i^{-1}}{2}$ so that the natural parameters can be expressed as a vector. We note with $\boldsymbol{\eta}$ the result of concatenating, element-wise, all the $\{\boldsymbol{\eta}^{(i)}\}_{i=1}^{d}$ into a single vector. Similarly, we may use $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to refer to all the parameters modeling the d drift and diffusion terms.

Although theoretically the trajectories following the natural gradient are independent of the parametrization, this property does not always hold in practical applications [92]. In our

case, we empirically found that using the natural parameters for parametrizing the Gaussian-Gamma distributions resulted in a slow convergence. We interpret this observation as follows. The SVAE must learn a phase space which summarizes all the relevant information of the dynamics of $\boldsymbol{y}_{1:N}$, according to a description consistent with a system of SDEs. To do so, the encoder needs to shape the phase space so that the dynamics can be easily described by means of this set of SDEs. At the beginning of the inference, the encoder and the SDEs will be quite inaccurate, resulting in changes in both the neural network and the SDE distribution parameters. We noticed that the changes on the distribution parameters decrease the squared error (between $\Delta \boldsymbol{x}_{1:N-1}$ and the drift estimate) at a very slow pace, which we attribute to the fact that both $\boldsymbol{\eta}_2$ and $\boldsymbol{\eta}_3$ affect the means $\boldsymbol{\mu}$ and that both parameters are changed at once. The issue is specially severe if the length-scales of the kernels are large because the covariance matrix strongly correlates all the inducing-points, which therefore cannot evolve independently.

To avoid this convergence issue we sought a parametrization which permitted each parameter to evolve independently from the others. This also applied to the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters. As detailed below, our parametrization uses the terms $\{\alpha_i/\beta_i\}_{i=1}^d$, which appear at Equation (4.21) determining the variance of the phase space. We parametrize the $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$ distribution using the parameters

$$\begin{aligned} \tilde{\eta}_0^{(i)} &= \log \alpha_i \qquad & \tilde{\eta}_1^{(i)} &= \log \frac{\alpha_i}{\beta_i}, \\ \tilde{\boldsymbol{\eta}}_2^{(i)} &= \boldsymbol{\mu}_i, & \tilde{\boldsymbol{\eta}}_3^{(i)} &= \boldsymbol{\Sigma}_i^{-1} \end{aligned} \tag{4.38}$$

with

$$\tilde{\boldsymbol{\eta}}^{(i)} = \left[\tilde{\eta}_0^{(i)}, \tilde{\eta}_1^{(i)}, \tilde{\boldsymbol{\eta}}_2^{(i)}, \mathrm{Vec}\left(\tilde{\boldsymbol{\eta}}_3^{(i)}\right)\right]^T,$$

where we have vectorized $\tilde{\boldsymbol{\eta}}_3^{(i)}$ (a matrix) so that the final parameter $\tilde{\boldsymbol{\eta}}^{(i)}$ is a vector. Finally, we note with $\tilde{\boldsymbol{\eta}}$ the vector that results from concatenating, element-wise, all the $\{\tilde{\boldsymbol{\eta}}^{(i)}\}_{i=1}^d$. Note that, from the definition of the parameters, it follows that $q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*)$ only depends on $\tilde{\boldsymbol{\eta}}$. Therefore, we shall write

$$q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \boldsymbol{\theta}^*) \triangleq q(\tilde{\boldsymbol{f}}, \boldsymbol{\lambda} \mid \tilde{\boldsymbol{\eta}}).$$

Since the computation of the Fisher matrix that results from this parametrization of $\tilde{\boldsymbol{\eta}}$ is quite tedious, we left the details to Appendix C. However, we show here the final matrix for

completeness:

$$\boldsymbol{F}\left(\tilde{\boldsymbol{\eta}}^{(i)}\right) = \begin{bmatrix} \alpha_i^2\psi^{(1)}(\alpha_i) - \alpha_i & 0 & \mathbf{0} & \mathbf{0} \\ 0 & \alpha_i & \mathbf{0} & \mathbf{0} \\ 0 & 0 & \frac{\alpha_i}{\beta_i}\boldsymbol{\Sigma}_i^{-1} & \mathbf{0} \\ 0 & 0 & \mathbf{0} & \boldsymbol{F}_{\tilde{\boldsymbol{\eta}}_3^{(i)},\tilde{\boldsymbol{\eta}}_3^{(i)}} \end{bmatrix}, \tag{4.39}$$

where $\psi^{(1)}(\cdot)$ is the trigamma function and $\boldsymbol{F}_{\tilde{\boldsymbol{\eta}}_3^{(i)},\tilde{\boldsymbol{\eta}}_3^{(i)}}$ is the Fisher submatrix for the vectorized parameter $\text{Vec}\left(\tilde{\boldsymbol{\eta}}_3^{(i)}\right)$. Instead of dealing with the vectorized version of $\tilde{\boldsymbol{\eta}}_3^{(i)}$, it is easier to express $\boldsymbol{F}_{\tilde{\boldsymbol{\eta}}_3^{(i)},\tilde{\boldsymbol{\eta}}_3^{(i)}}$ as relating the entries from the matrix itself. The element from $\boldsymbol{F}_{\tilde{\boldsymbol{\eta}}_3^{(i)},\tilde{\boldsymbol{\eta}}_3^{(i)}}$ relating $\left[\tilde{\boldsymbol{\eta}}_3^{(i)}\right]_{jk} = \Sigma_{jk}^{-1}$ with $\left[\tilde{\boldsymbol{\eta}}_3^{(i)}\right]_{lm} = \Sigma_{lm}^{-1}$ can be written as

$$F_{[\tilde{\boldsymbol{\eta}}_3^{(i)}]_{jk},[\tilde{\boldsymbol{\eta}}_3^{(i)}]_{lm}} = \frac{1}{4}\left[\Sigma_{jl}\Sigma_{km} + \Sigma_{jm}\Sigma_{kl}\right].$$

The main advantage of our parametrization is that we can directly update α_i/β_i and $\boldsymbol{\mu}_i$, avoiding the convergence problem described in the previous paragraphs. The main disadvantage is that this parametrization requires the calculation of the Fisher matrix (Equation (4.39)) in each update of the SVAE. Therefore, each individual iteration of the algorithm will be slower than the natural parameters version. However, we find that our version usually converges to better representations. A second option that we did not test is briefly mentioned in [61]. This work introduces SVI for GP models, and derives the required natural gradients in terms of the natural parameters. They empirically found that holding the variational covariance parameters fixed in the first iterations of the inference improved the convergence, although they did not try to discuss the causes. We suppose that an interpretation similar to that given for our observations may also apply here, although further research is needed to verify this.

4.5.4 Inducing points initialization

The initialization of the pseudo-inputs can be problematic. The strategy used in Chapter 3 (creating a uniform grid using the minimum and maximum values of $\boldsymbol{x}_{1:N}$, see Section 3.4.1) cannot be directly used since $\boldsymbol{x}_{1:N}$ is a latent space. Furthermore, the boundaries of $\boldsymbol{x}_{1:N}$ change at a rapid pace during the first epochs of training, preventing the pseudo-inputs to keep track of the changes in the phase space structure. This may cause the pseudo-inputs to get stuck in positions far from $\boldsymbol{x}_{1:N}$, precluding the proper learning of the dynamics.

The issue of the fast evolution of the phase space can be alleviated by removing the bias nodes from the latest layers of the encoder. However, it does not guarantee that the problem is completely avoided.

A practical approach to tackle the issue consists on running a few epochs of the training algorithm, letting the encoder and the decoder change their weights, but without optimizing neither the inducing-points nor the $\tilde{\boldsymbol{\eta}}$ parameters of the SDE. We empirically found that 1-5 epochs are usually enough to overcome the "quick evolution" stage of the encoder. After finding some representative sample of the phase space, the inducing-points can be subsequently safely initialized.

Let us assume that some representative trajectory of the phase space is available: $\tilde{\boldsymbol{x}}_{1:N'}$. Note that the length of this trajectory is not necessarily N. We consider two simple approaches for the initialization of the pseudo-inputs: a grid based approach and a K-means-based approach. In the grid based approach a uniformly spaced grid defined by the most extreme values of each dimension of the phase space is created. For example, it is possible to use

$$\big(\min(\tilde{x}_{i,1:N'}), \max(\tilde{x}_{i,1:N'})\big) \qquad \text{for } i = 1, 2, \ldots, d$$

to define the grid limits, although more complex selection criteria can be employed to take into account possible fast expansions of the phase space. Although simple, the main disadvantage of the grid based approach is that some inducing points may be placed in areas of the phase space that are not frequently visited. This issue is particularly important if the phase space is structured as a low-dimensional manifold. An alternative approach that takes greater account of the phase space structure is to select the inducing points as the centroids that result from applying the K-means algorithm on $\tilde{\boldsymbol{x}}_{1:N'}$ [89].

To make both approximations work, some representative trajectory $\tilde{\boldsymbol{x}}_{1:N'}$ must be found. Since the pseudo-inputs initialization is only done once, it is possible to use all the available data $\{\boldsymbol{y}^{(r)}_{1:N}\}_{r=1}^{R}$ for building $\tilde{\boldsymbol{x}}_{1:N'}$ without compromising the efficiency of the algorithm. In principle, there seem to be two options. Either using the outputs of the encoder to build $\tilde{\boldsymbol{x}}_{1:N'}$, or using the forward-backward algorithm to draw samples with the same aim. However, at this point the $\tilde{\boldsymbol{\eta}}$ parameters have not been optimized (see the beginning of this section), and sampling from the phase space does not make sense since the dynamics have not yet been learned. Therefore, the encoder outputs must be used to find some representative sample of the phase space. To do so, the data $\{\boldsymbol{y}^{(r)}_{1:N}\}_{r=1}^{R}$ feeds the encoder, and the resulting outputs serve as building blocks for $\tilde{\boldsymbol{x}}_{1:N'}$. For the K-means-based approach, the encoder's mean can act as $\tilde{\boldsymbol{x}}_{1:N'}$. Alternatively, and to prevent against future expansions of the phase space, some

dilation operations can also be applied to $\tilde{\boldsymbol{x}}_{1:N'}$ before using the K-means algorithm. Note that the encoded mean will be organized in batches (just like, $\{\boldsymbol{y}_{1:N}^{(r)}\}_{r=1}^{R}$), but it is straightforward to arrange them as a single trajectory.

For the grid based approach, the encoder's covariance must be taken into account to compute proper extreme values. However, the covariances can be approximated as being diagonal, since the pseudo-inputs are just being initialized, and there is no point in performing detailed calculations. Then, a lower bound and an upper bound for each dimension are computed using the three-sigma rule for Gaussian distributions:

$$\big(\min(m_{i,1:N'} - 3s_{i,1:N'}), \max(m_{i,1:N'} + 3s_{i,1:N'})\big) \qquad \text{for } i = 1, 2, \ldots, d,$$

being $\boldsymbol{m}_{1:N'}$ and $\boldsymbol{s}_{1:N'}$ the means and standard deviations that can be derived from arranging the encoder's outputs as a single trajectory and treating the covariances as diagonal matrices.

4.6 Validation on synthetic data

In this section, we apply our model to a variety of experiments, to illustrate its potential to capture salient features of experimental time series. The experiments scale in complexity, and each of them tries to illustrate different aspects of the algorithm.

The experiments were run sharing a similar experimental setup. The encoder consisted of two stacked RNNs with GRUs of size 50, followed by a MLP. The MLP used 50 hidden units with Exponential Linear Unit (ELU) activations [22], and a linear output layer without bias (to help with the fast evolution of the phase space issue, see Section 4.5.4). Another MLP with a similar configuration was used as decoder, although bias units where included in its output layer. Despite the decoder being quite small, we also applied dropout with rate 0.25 to prevent it from "memorizing" the phase space [122]. This was key to prevent the phase space from collapsing to a random-walk-like phase space. In this case, the phase space closely matches the prior and there is no incentive to learn more detailed dynamics since the encoder-decoder pair can overfit the data.

The lower bound was optimized using stochastic gradient descent with Nesterov momentum [101]. The learning rate was set to 10^{-4}, whereas the momentum term was set to 0.9.

4.6.1 Excited oscillator

We begin with a simple example in which a deterministic one-dimensional time series $y_{1:N}$ oscillates while slowly increasing its amplitude. Specifically, we generated several realizations of the model using

$$y_t^{(r)} = \sqrt{2}\exp(0.002 \cdot t)\cos\left(\frac{2\pi}{100}t + \phi^{(r)}\right), \qquad \begin{matrix} r = 1, 2, \ldots, 120 \\ t = 1, 2, \ldots, 500 \end{matrix}$$

where $\phi^{(r)}$ is a random sample generated from a uniform distribution $\mathcal{U}(0, 2\pi)$. Note that this is the only source of randomness of the system. Although our first experiment, this time series is quite challenging for our model since the diffusion term should vanish to fit the deterministic equation generating the data ($\beta/\alpha \to 0$). However, the smoothness of the signal makes it easier to interpret the figures from the SVAE algorithm, making it a good illustrative example.

Besides the SVAE architecture described at the beginning of the section, the following parameters were selected. We used $d = 2$ as dimension of the phase space; the size of the graph was set to $N = 150$, effectively limiting the TBPTT to 150 steps; the Monte Carlo approximations were calculated using $S = 10$ samples; and 16 inducing points were employed, initializing them after 5 epochs of training with the grid based approach (see below for more details).

Figures 4.15 and 4.16 show a small segment of the phase space and the reconstructed time series at several stages of the training procedure. Note that the phase space shows the representation generated by the encoding network and the dynamics fitted by the SDE (Figure 4.15). A single illustrative sample of $\boldsymbol{x}_{1:N}$ is also shown. On the other hand, the reconstructed time series shown in Figure 4.16 is the output of the decoding network when fed with the same sample shown in the phase space.

It is illustrative to briefly discuss the training evolution, for a better understanding of the algorithm. We run 5 epochs without optimizing the inducing points and the SDE parameters, as described in Section 4.5.4. After these 5 epochs, the inducing points are initialized using the grid based approach, taking also into consideration the possible expansion of the phase space. This is shown in Figure 4.15a, with the inducing-points represented as reddish dots. At this point, the encoding network has already found a nice phase space that captures the key features of the time series $y_{1:N}$: the oscillatory pattern and the slow change in amplitude. This results in the spiral pattern shown in Figure 4.15a. However, this representation has large

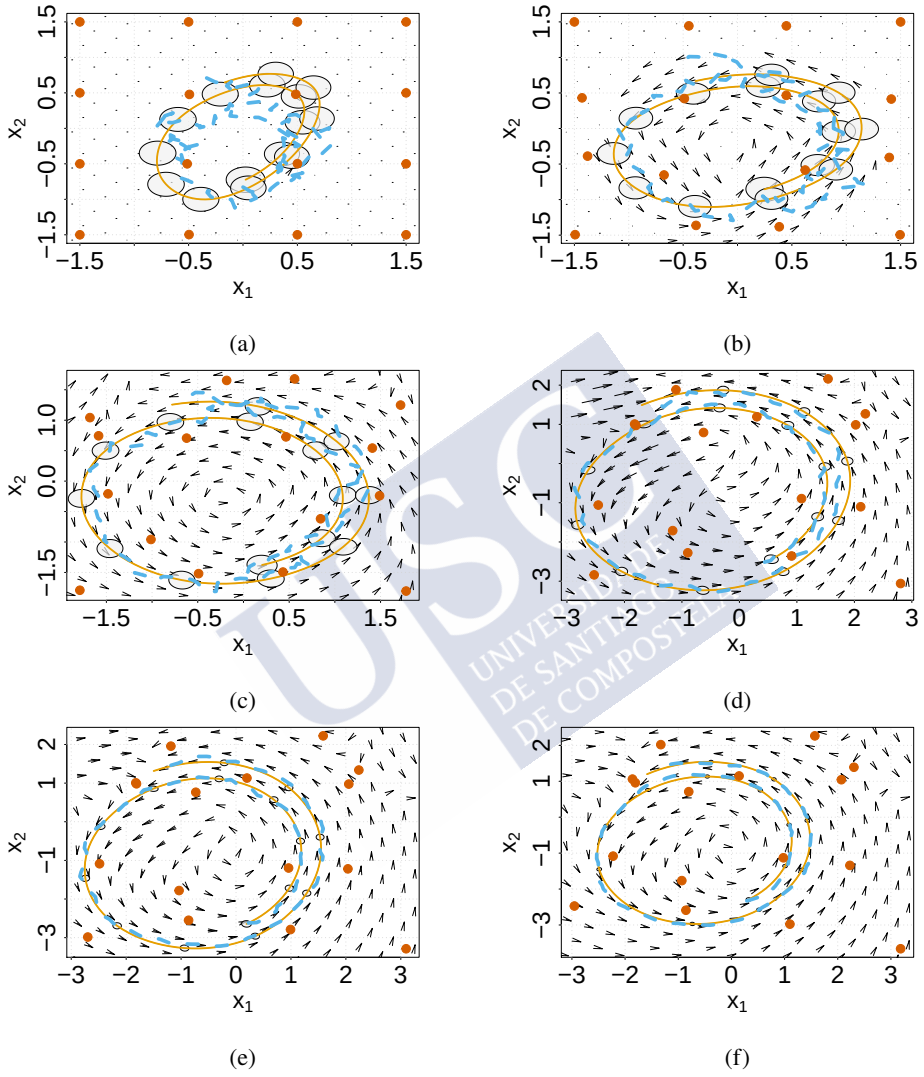

Figure 4.15: Evolution of the phase space after (a) 35 iterations, (b) 131 iterations, (c) 323 iterations, (d) 1163 iterations, (e) 1847 iterations, and (f) 4247 iterations. The mean and covariances of the encoding network are represented by orange lines and gray ellipses, respectively. A single illustrative sample of the phase space is shown with a dashed blue line. Finally, the inducing-inputs are represented with reddish points, whereas the drift is represented by arrows.

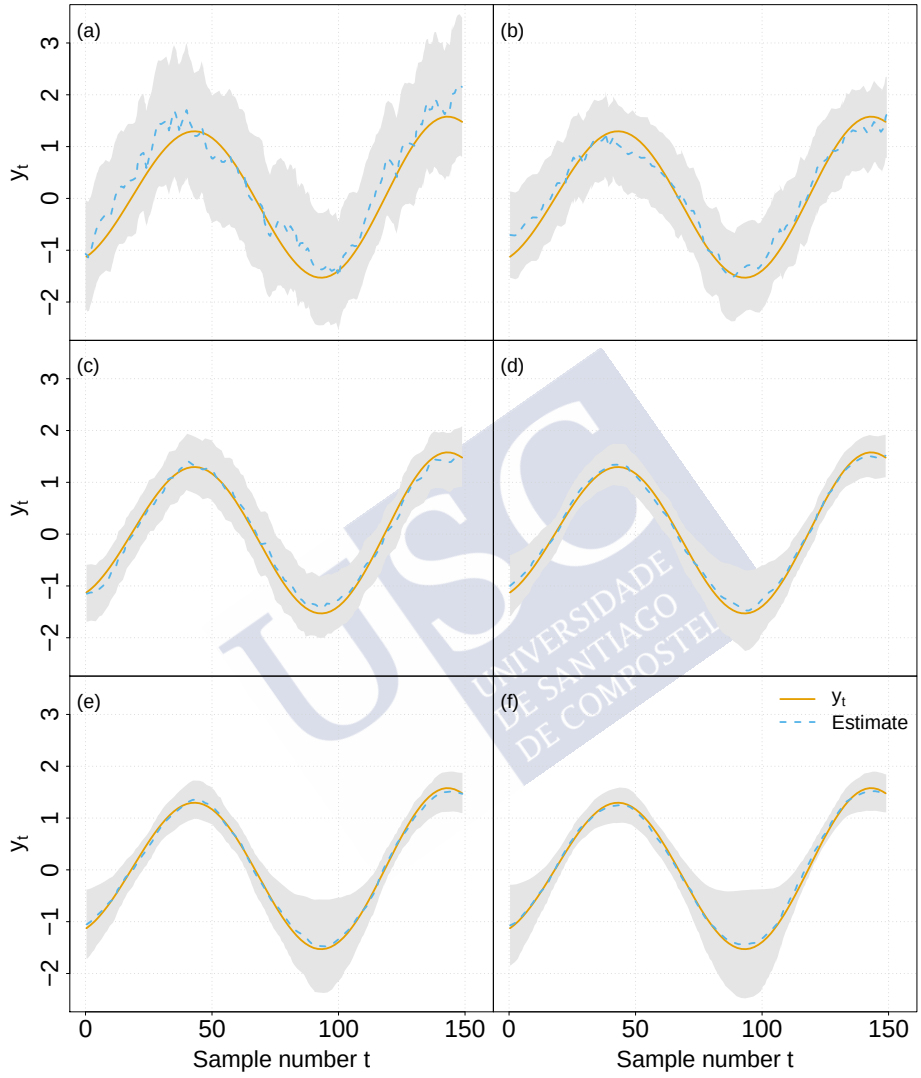

Figure 4.16: Reconstructed time series for the excited oscillator model output by the decoding network after (a) 35 iterations, (b) 131 iterations, (c) 323 iterations, (d) 1163 iterations, (e) 1847 iterations, and (f) 4247 iterations. The shaded areas represent the 95% confidence interval of the reconstructed time series.

uncertainty, as shown by the large gray ellipses, which represent the covariances fitted by the encoder. The dynamics of the phase space, represented with arrows, have not yet been fitted. The large uncertainty in the encoding and the poor fit of the drift results in samples of $\boldsymbol{x}_{1:N}$ that, although following the mean trajectory generated by the encoding network, are quite noisy (dashed blue line). The fact that the node potentials $\psi(\boldsymbol{x}_t, y_t, E)$ "force" these samples to follow the mean of the encoding network permits the dynamics of the phase space to be slowly learned (Figures 4.15a-4.15d).

The samples from the phase space feed the decoding network, which estimates the original time series $y_{1:N}$. Since the inputs are noisy, the outputs are also noisy, resulting in a large error term (see Figure 4.16a). However, they approximately follow the changes on $y_{1:N}$.

Driven by the feedback of the decoding network, the encoding network quickly decreases the covariances, generating smoother samples (Figures 4.15b-4.15d). The decoding network encourages these smooth samples since the smoother its inputs are, the smoother its outputs will be; which permits the network to better fit the observed signal $y_{1:N}$ (Figures 4.16b-4.16d). At the end of the training (Figures 4.15e-4.15f), the samples are very smooth, following closely the mean generated by the encoding network. At the same time, this permits the decoding network to generate an accurate representation of the original signal (Figures 4.16e and 4.16f). Note that the confidence intervals generated by the decoding network decrease around the linear regions of $y_{1:N}$, while increasing around the extreme values of the signal. This is reasonable, since the linear regions can be easily fitted, whereas the extreme values are subject to a larger uncertainty, due to the fact that the signal slowly changes its amplitude.

At the same time the dynamics are learned, the inducing-inputs move around the phase space, clustering around the mean generated by the encoding network.

Note that, even at the beginning of the training, when the dynamics are poorly fitted, the reconstructed series shown in Figure 4.16 are quite accurate. Therefore, Figure 4.16 should not be interpreted as a definitive proof of the ability of the SVAE to capture the dynamics of the input time series, but as demonstration that the encoder and decoder networks work properly. To confirm that the dynamics have been properly learned, we generate predictions from the SDE SVAE given some small segment of data as input. This is illustrated in Figure 4.17. To generate a new prediction, the encoder creates a trajectory in phase space for the given time series (colored in black, up to the vertical line). When the encoder runs out of input data, the last state of the encoded trajectory is used as the initial state of the fitted SDE. The SDE generates a new trajectory without the intervention of the encoder. Finally, the decoder maps

the new phase space trajectories to the original space, generating new synthetic time series. Note that, since the SDE is a stochastic model, different trajectories may result from the same initial state. In Figure 4.17, two different synthetic time series are shown (in dashed-orange and solid-blue). Since the original time series is deterministic, the differences between both samples are small, although noticeable. Despite this, the synthetic time series accurately reproduce the input signal.

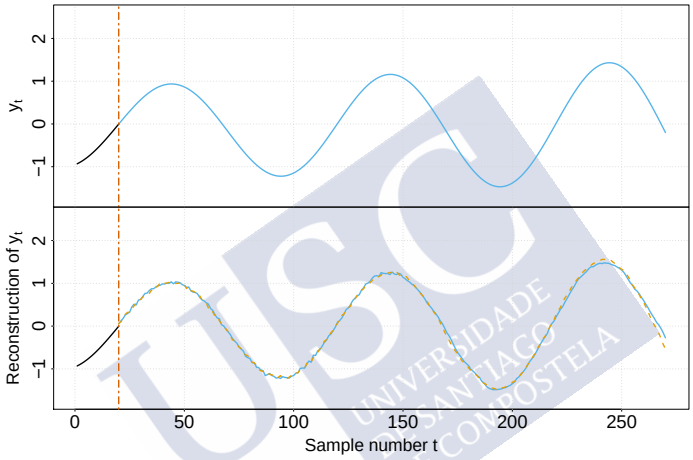

Figure 4.17: Predictions from a SDE SVAE for the excited oscillator. The top panel shows the real data. The bottom panel shows two different predictions (in dashed-orange and solid-blue) generated by the SVAE given the data up to the vertical line, which is colored in black.

Figure 4.17 proves the ability of the SDE SVAE to effectively model the excited oscillator time series, despite the fact of using a SDE to fit a purely deterministic pattern. Nevertheless, it is worth illustrating some limitations of the SDE SVAE. Figure 4.18 illustrates a case where the SDE SVAE fails to reproduce the input series, despite it seems similar to Figure 4.17. The key difference is the amplitude of the series (see the range of the y-axis in Figure 4.17 and in Figure 4.18). The encoder has not been exposed to anything similar before (in terms of amplitude) and therefore, it outputs a code that falls into an unexplored region of the phase space. In this region, the predictions for the drift function are equal to the prior since no data has changed our prior beliefs. Therefore, the drift is zero and there is no evolution in phase space, which results in the decoder outputting a constant value. It is worth noting that there might be some evolution in phase space if the diffusion term was not negligible, in which case

the dynamics in phase space would follow a random walk. However, we have assumed the diffusion term to be constant, as noted in Equation (4.4). Since the diffusion term is close to zero to fit the deterministic input series, it follows that it must be close to zero in all the phase space, preventing the random walk to occur.

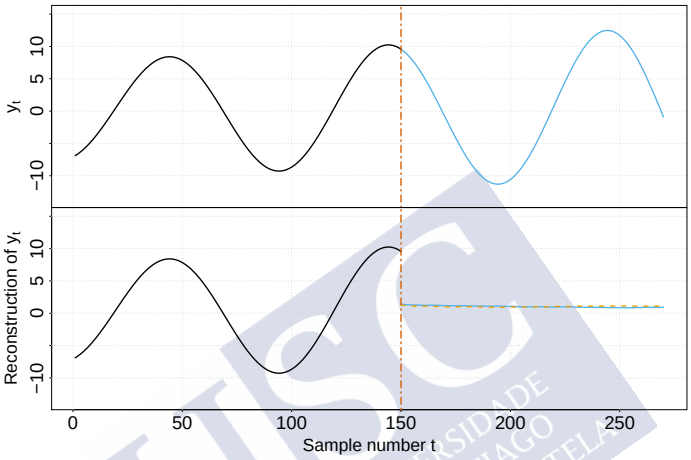

Figure 4.18: Predictions from a SDE SVAE for an unseen series from an excited oscillator.

4.6.2 Lotka-Volterra

In this section, we apply our SDE-based SVAE to a nonlinear stochastic model inspired by the Lotka-Volterra equations. These equations are used in the study of biological ecosystems to describe the interactions between a predator and its prey. The Lotka-Volterra differential equations are usually written as

$$\begin{aligned} \frac{dx_1(t)}{dt} &= \alpha x_1(t) - \beta x_1(t) x_2(t), \\ \frac{dx_2(t)}{dt} &= -\gamma x_2(t) + \delta x_1(t) x_2(t), \end{aligned} \tag{4.40}$$

where x_1 and x_2 are the prey and predator populations, respectively. In the absence of predators, the rate of change of the prey population is proportional to the current population. This is represented by the $\alpha x_1(t)$ term in Equation (4.40) and results in an exponential growth. In the presence of predators, the prey is killed at a rate that is proportional to the rate at which predators and prey encounter each other, represented by the term $\beta x_1(t) x_2(t)$ in Equation (4.40).

Therefore, the equation for the prey population indicates that the rate of change of the prey's population depends on its reproduction rate minus the rate at which it is hunted. Note that, in this model, the prey does not die of natural causes.

On the other hand, the rate of change of the predator population also depends on the interactions between the predator and the prey $\delta x_1(t)x_2(t)$. Note that this term is similar to the predation rate $\beta x_1(t)x_2(t)$. However, the term now contributes to the growth of the predator population (positive sign), and a different constant δ is used since the rate of growth of the predator population does not necessarily match the rate at which it hunts the prey. Finally, $\gamma x_2(t)$ represents the rate of population decrease due to natural death. The assumption is that the greater the population size is, the greater the number of deaths (per unit time). This leads to an exponential decay in absence of prey. Therefore, the growth rate of the predator population is determined by the rate at which it hunts the prey minus its death rate.

Equation (4.40) is a deterministic model, but it is possible to modify it to take into account stochastic events. The diffusion approximation of the Lotka-Volterra model, described in [4, 15], is given by

$$\begin{aligned}\begin{bmatrix} dx_1(t) \\ dx_2(t) \end{bmatrix} &= \begin{bmatrix} \alpha x_1(t) - \beta x_1(t)x_2(t) \\ \beta x_1(t)x_2(t) - \gamma x_2(t) \end{bmatrix} \mathrm{d}t \\ &+ \begin{bmatrix} \alpha x_1(t) + \beta x_1(t)x_2(t) & -\beta x_1(t)x_2(t) \\ -\beta x_1(t)x_2(t) & \beta x_1(t)x_2(t) + \gamma x_2(t) \end{bmatrix}^{1/2} \begin{bmatrix} dW_1(t) \\ dW_2(t) \end{bmatrix}.\end{aligned} \tag{4.41}$$

Note that, in this limit, δ becomes β, in contrast with Equation (4.40). Furthermore, the diffusion matrix is diagonal and non-constant, which appears to contradict our SDE model. In Section 4.1.2 we argued for the generality of the proposal on the basis of the existence of the Lamperti transformation, which is able to diagonalize diffusion matrices under relatively wide conditions. Since a representation with a state independent diffusion is guaranteed to exist, what we are implicitly assuming is that the encoder network will be able to discover such representation. In this experiment we test this assumption.

We analyse the case $\alpha = 30$, $\beta = 0.2$ and $\gamma = 18$. An illustrative portion of the real phase space of the system is shown in Figure 4.19. The SVAE was fed with 10 different realizations of the $x_1(t)$ signal, sampled at 1000 Hz. Each of the resulting input signals, $y_t^{(r)} = x_1^{(r)}(t/1000)$ had a length of 600 samples.

We used $d = 2$ for the phase space dimension, $N = 150$ for the size of the graphical model, $S = 10$ for the number of Monte Carlo samples, and $m = 10$ for the number of inducing points.

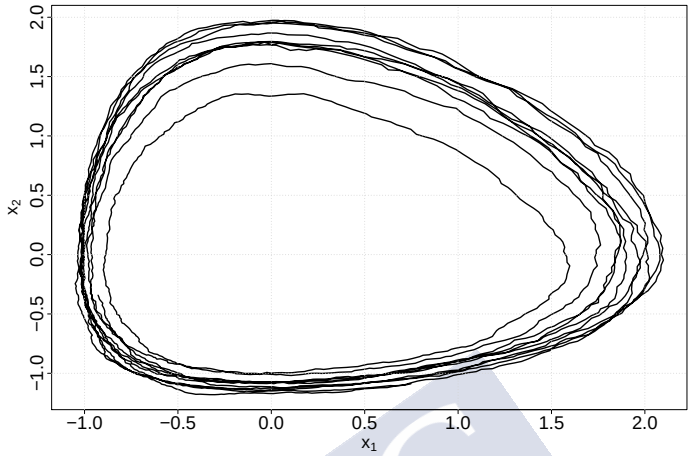

Figure 4.19: Lotka-Volterra phase space.

As in Section 4.6.1, Figures 4.20 and 4.21 show a small segment of the phase space and the corresponding reconstruction at different stages of the training. Figure 4.20a shows the phase space at the very beginning of the procedure, illustrating the need for the pseudo-inputs initialization methods discussed in Section 4.5.4. At this point, the positions of the inducing points were not initialized and therefore they are randomly distributed around $(0, 0)$. Although it might appear that their positions are reasonably good, the encoder's mean quickly expands (compare the magnitudes of the axes in Figures 4.20a and 4.20b), leaving the inducing points behind. To address this, the pseudo-inputs are initialized after 5 epochs, as discussed in Section 4.5.4. In this example, we used the K-means approach, illustrated in Figure 4.20b. To prevent against possible expansions of the phase space, the K-means algorithm was applied to dilated versions of the illustrative trajectory. The later evolution of the inducing points in phase space is purely due to gradient descent optimization.

Note that, in Figure 4.20a, the mean output by the encoder captures the oscillatory nature of the Lotka-Volterra model. However, the samples cannot reproduce this pattern since the dynamics have not been learned yet. At the same time, the SDE is fitted using the phase space samples. This seems like a vicious circle preventing the model to learn any dynamics or drawing plausible phase space samples. The mechanism whereby it is indeed possible to learn the SDE was already sketched in Section 4.6.1, but it is worth emphasizing it due

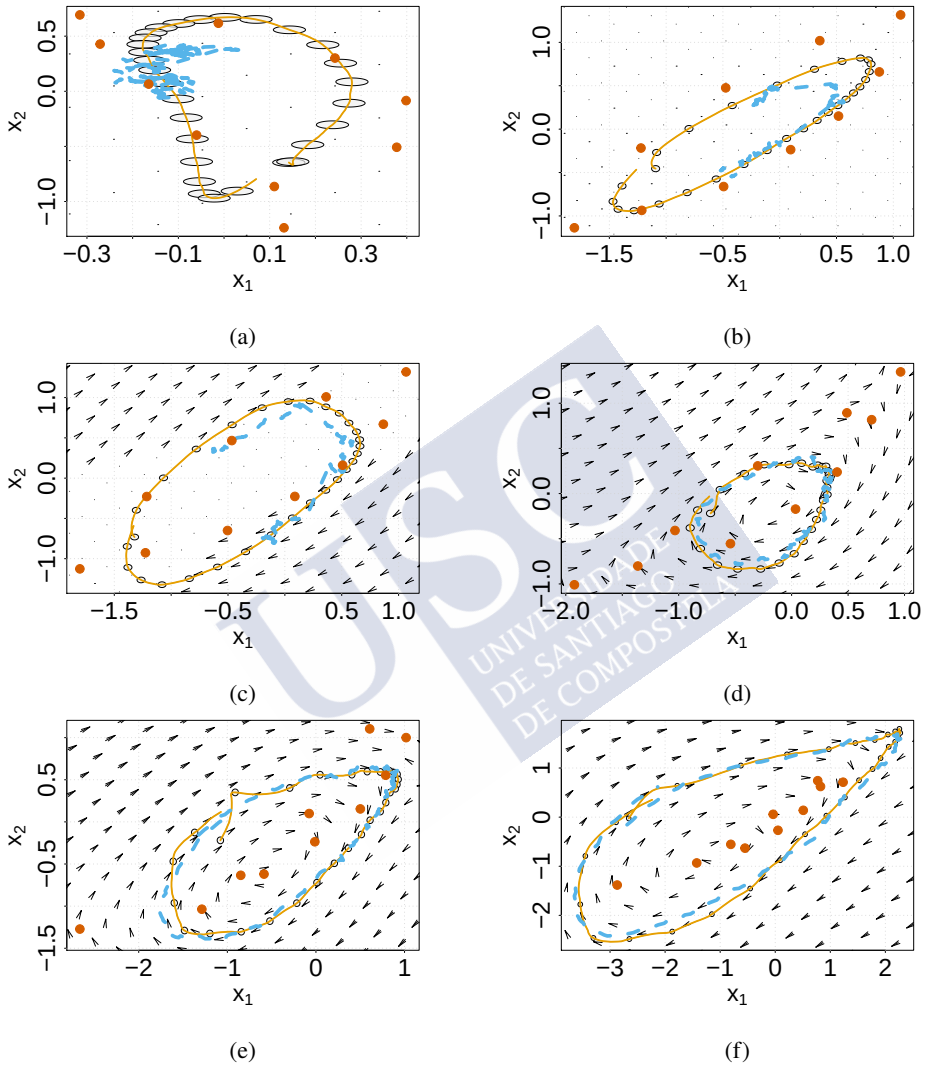

Figure 4.20: Evolution of the Lotka-Volterra phase space after (a) 1 iteration, (b) 5 iterations, (c) 10 iterations, (d) 25 iterations, (e) 55 iterations, and (f) 450 iterations. The mean and covariances of the encoding network are represented by orange lines and gray ellipses, respectively. A single illustrative sample of the phase space is shown with a dashed blue line. Finally, the inducing-inputs are represented with reddish points, whereas the drift is represented by arrows.

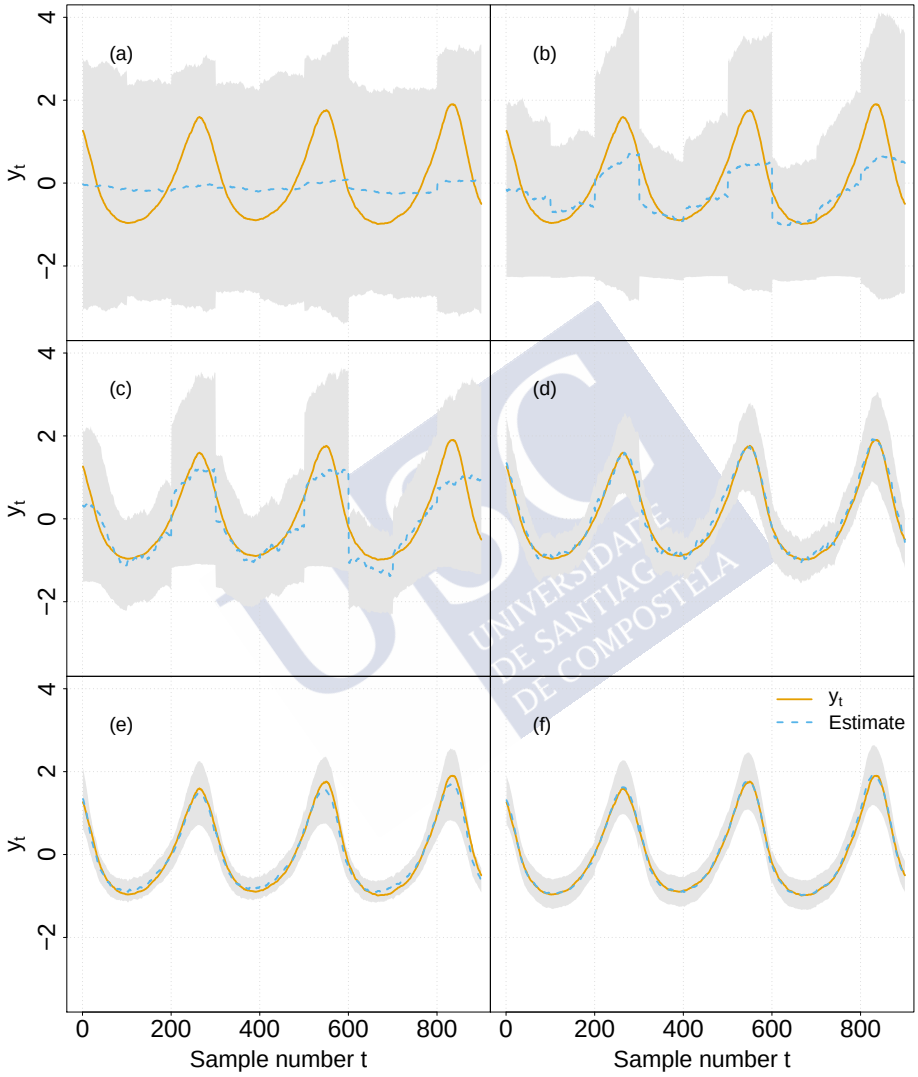

Figure 4.21: Reconstructed Lotka-Volterra time series output by the decoding network after (a) 1 iteration, (b) 5 iterations, (c) 10 iterations, (d) 25 iterations, (e) 55 iterations, and (f) 450 iterations. The shaded areas represent the 95% confidence interval of the reconstructed time series.

to its importance. The encoder-decoder pair quickly discovers that the lower bound can be increased by reducing the mean squared error of the reconstructed time series. To do so, it can increase the accuracy of the reconstructions, or it can also reduce the variability of the samples being drawn, which reduces the variance of the mean squared error. To reduce the variability of the samples, the encoder just decreases the encoder's covariances (compare the ellipses from Figures 4.20a and 4.20b). When the covariances of the encoder have a small magnitude, the forwards-backward algorithm forces the samples to tightly follow the encoder's mean resulting in more accurate reconstructions. Besides increasing the lower bound, there is another important side effect: the dynamics of the phase space can be learned. Note the improvement in the learned dynamics from Figure 4.20b to Figure 4.20c. When the dynamics are accurate enough, the samples are able to complete a full cycle in the phase space, which results in accurate reconstructions of the time series, as shown in Figures 4.20d and 4.21d. The optimization continues polishing the phase space and reducing the variance of the predictions, as illustrated in Figures 4.20e-4.20f and Figures 4.21e-4.21f, respectively.

Figure 4.22 shows predictions generated by a trained SDE SVAE, illustrating its capability to generate credible synthetic signals with similar dynamical properties to those of the training data.

4.6.3 Stochastic Lorenz system

In this section, we study a simple stochastic version of the Lorenz system, a simplified model for the study of atmospheric convection:

$$
\begin{aligned}
dx_1 &= \big(\sigma(x_2 - x)\big)dt + dW_1(t), \\
dx_2 &= \big(x_1(\rho - x_3) - x_2\big)dt + dW_2(t), \\
dx_3 &= (x_1 x_2 - \beta x_3)dt + dW_3(t).
\end{aligned}
$$

We focused on the values $\sigma = 10, \beta = 8/3, \rho = 28$ which, in the case of the deterministic model, are known to produce chaotic behavior. A representative portion of the real phase space of the system is shown in Figure 4.23. The SVAE was fed with 7 different realizations of the $x_1(t)$ signal, sampled at 1000 Hz. Each of the resulting input signals, $y_t^{(r)} = x_1^{(r)}(t/1000)$ had a length of 5000 samples.

We used $d = 3$ for the phase space dimension, $N = 150$ for the size of the graphical model, $S = 10$ for the number of Monte Carlo samples, and $m = 10$ for the number of inducing points. The inducing points were initialized after 5 epochs of training using the K-means approach.

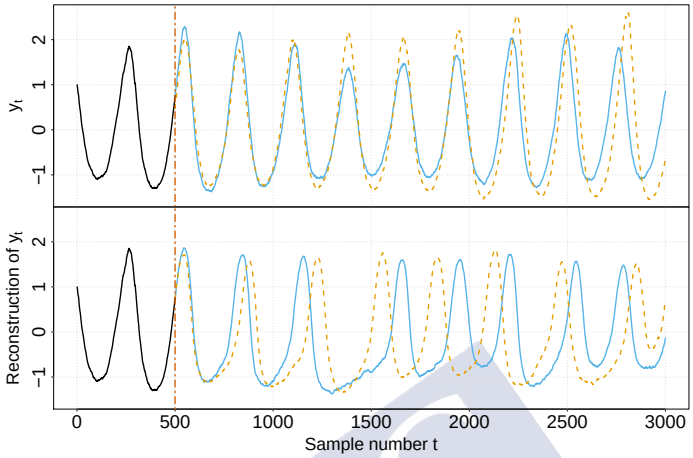

Figure 4.22: Predictions from a SDE SVAE for the Lotka-Volterra model. The top panel shows real data generated from Equation (4.41). Since the model is stochastic, two different realizations are shown. The bottom panel shows two synthetic time series generated by the SVAE given the data up to the vertical line, colored in black.

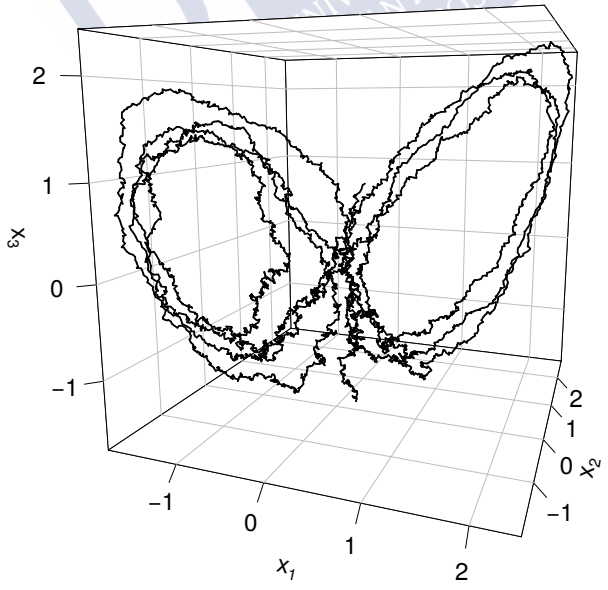

Figure 4.23: Stochastic Lorenz phase space.

Figures 4.24 and 4.25 show a small segment of the phase space and the corresponding reconstruction at various points of training. Note that, since the phase space is tridimensional, we do not show the vector field representing the drift. Figure 4.24 shows that the SDE SVAE is able to construct a phase space that resembles the original one (see Figure 4.23). Moreover, Figure 4.24 illustrates how the decoder is able to accurately reconstruct the input time series from the codes output by the encoder.

Finally, Figure 4.26 shows predictions generated by a trained SDE SVAE. In this case, the synthetic signals can be easily differentiated from the original ones. Although the main dynamical features have been captured, the generated time series seem like smoothed versions of true realizations of the stochastic Lorenz process.

4.7 Discussion

In this chapter, we have introduced a SVAE that is able to simultaneously learn a Markovian representation of the data and its dynamics, that are assumed to be described by a SDE. To this end, the SDE SVAE combines graphical models, which impose the restriction of having SDE-like dynamics, with deep learning methods, which enable the rich modeling of complex dynamics. There are no strict assumptions on the data and therefore almost any time series is susceptible to be analyzed by means of the methods presented in Chapter 3. Furthermore, SVAE takes advantage of SVI and natural gradients to implement an efficient inference algorithm, which permits the application of the method to large datasets that do not fit in main memory.

Since this chapter exploits previous work, it is important to briefly highlight the main differences between our proposal and related works, with special reference to [69], which inspired many of our ideas.

As mentioned previously, our proposal fits nicely in the general SVAE framework proposed in [69]. Indeed, after the introduction of the inducing points as global variables, the resulting graphical model (see Figure 4.10) is a particular case within SVAEs since the model is not really structured. At first, this appears to suggest that the modeling capabilities of our proposal are limited in comparison with those of a generic SVAE. This is only partially true. Our SDE-based model permits the rich modeling of complex dynamics because the drift term, through a GP, may be a nonlinear function. By contrast, [69] illustrates the use of SVAEs using Switching Linear Dynamical System (SLDS), i.e., a model with conditional linear dynamics; the model

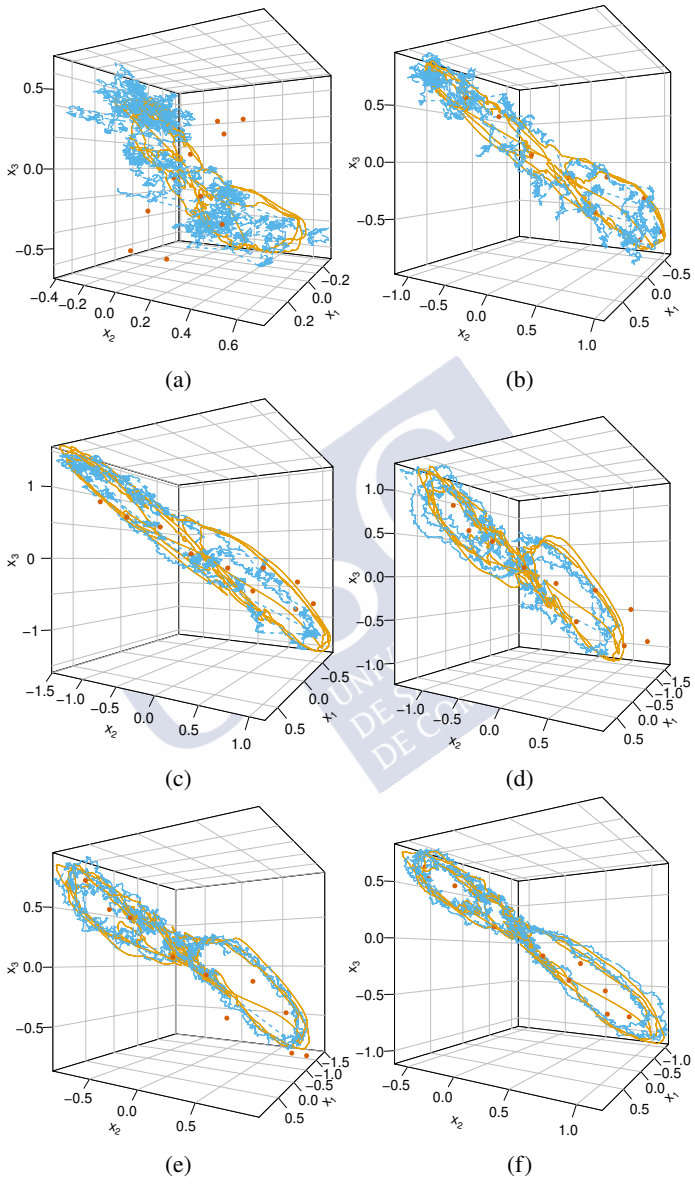

Figure 4.24: Evolution of the Lorenz phase space after (a) 1 iteration, (b) 2 iterations, (c) 4 iterations, (d) 8 iterations, (e) 17 iterations, and (f) 52 iterations. The mean of the encoding network are represented by orange lines. A single illustrative sample of the phase space is shown with a dashed blue line. Finally, the inducing-inputs are represented with reddish points.

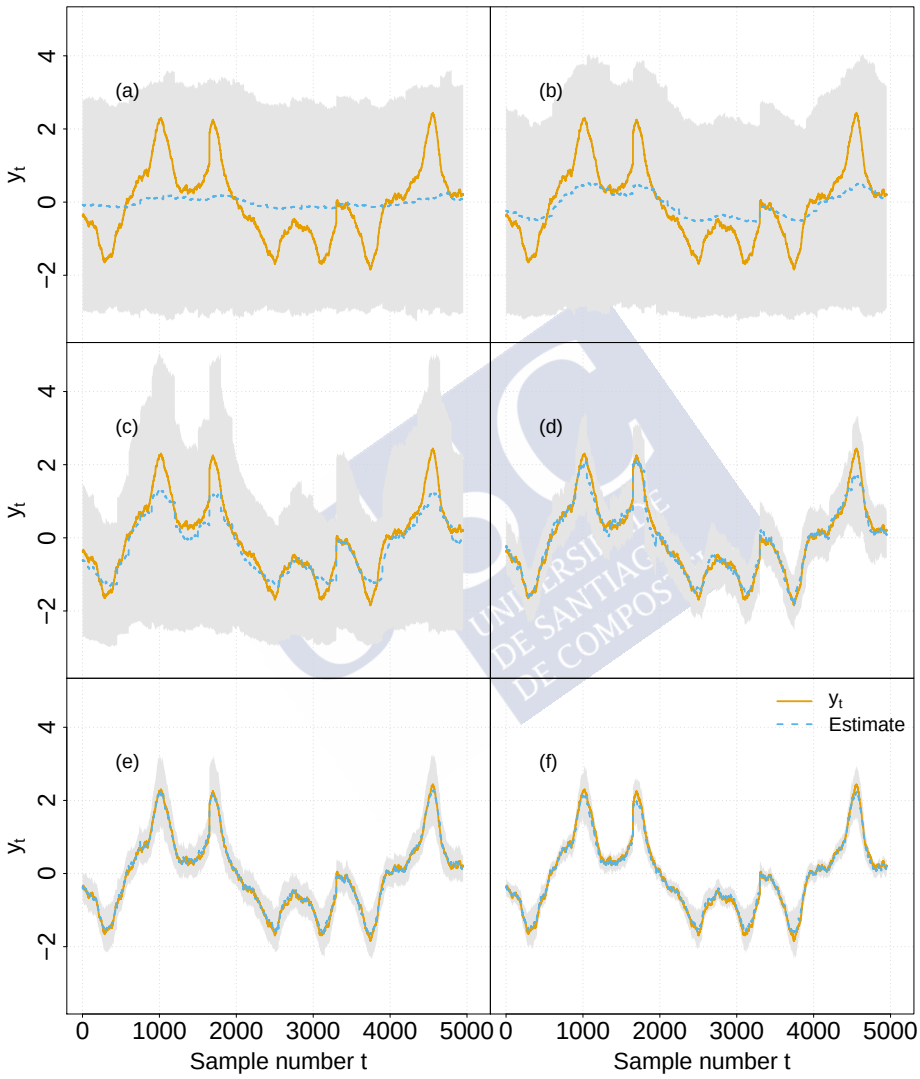

Figure 4.25: Reconstructed time series output by the decoding network after (a) 1 iteration, (b) 2 iterations, (c) 4 iterations, (d) 8 iterations, (e) 17 iterations, and (f) 52 iterations. The shaded areas represent the 95% confidence interval of the reconstructed time series.

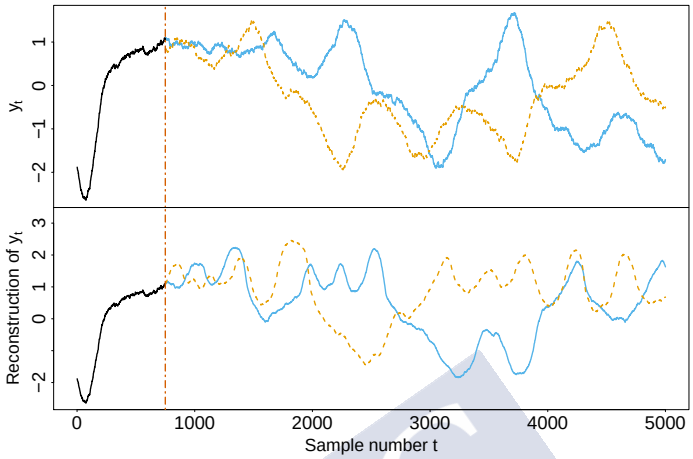

Figure 4.26: Predictions from a SDE SVAE for the Lorenz model. The top panel shows the real data. The bottom panel shows two different predictions (in dashed-orange and solid-blue) generated by the SVAE given the data up to the vertical line, which is colored in black.

is still very expressive because the latent dynamics of $\boldsymbol{x}_{1:N}$ are controlled by another set of local latent variables $\boldsymbol{z}_{1:N}$. The sequence $\boldsymbol{z}_{1:N}$ is usually referred to as the mode sequence, since it permits the system to switch the dynamical mode. The switching modes of SLDS would permit to effectively approximate nonlinear or even non-stationary dynamics.

If there is a single nonlinear dynamic mode in the data, both SDE-based and SLDS-based SVAEs can be regarded as equally expressive, although different in meaning. In this case, the SDE SVAE provides a physically meaningful phase space, whereas the phase space of the SLDS SVAE would be difficult to interpret due to the switching dynamics. Therefore, the SDE SVAE may be better suited for building physically plausible models, which can be subsequently refined or used for taking modeling decisions, expediting model building.

On the other hand, a single SDE cannot accurately describe data driven by nonlinear non-stationary dynamics, whereas that the SLDS can still try to model it by adding more modes. It must be noted, however, that our proposal could be extended by adding a mode sequence to the SDE SVAE.

There is also another key difference between the two works which can be understood by comparing the application domain in which they are considered. [69] illustrates SVAEs using high dimensional video recordings. In this context, the encoder is a MLP that aims to reduce

the dimensionality of the input data. In our proposal, we focused on finding a proper phase space for experimental input data, which usually requires increasing the dimensionality of the input space. To that end, we consider the use of a RNN as encoder, since its ability to contextualize each input $\boldsymbol{y}_t$ is key to build a proper phase space.

In this sense, our work can be compared with [78], which proposed a generalization of Kalman filters by extensively using deep neural networks. In [78] the encoder, the decoder and the dynamical model are parametrized through neural networks, and they explicitly consider RNNs as a possible recognition network. The resulting model subsumes a large family of linear and nonlinear space models as particular cases. However, this approach can not be extended to general graphical models, and it does not exploit SVI nor natural gradients.

In summary, we find that the main contributions of this chapter are: (1) leveraging SVAEs with the methods from Chapter 3, which enables the modeling of nonlinear dynamics without switching modes; (2) providing a physically meaningful representation of the data, since it can be interpreted as a phase space described by a SDE; and (3) exploiting recent advances in GP dynamical systems to efficiently sample from the nonlinear SDE phase space, as required by the SVAE training algorithm.

The ability of the SDE SVAE to capture the main dynamical features of the data and summarize them in the phase space was demonstrated using synthetic data. Although the results proved that the SDE SVAE is a powerful modeling tool, some drawbacks were also revealed.

Figure 4.18 illustrates an example in which the SDE SVAE fails to generate convincing samples from an input signal, despite it resembles those time series used for the training. However, the signal from Figure 4.18 is, in fact, quite different from the training data in terms of amplitude. It may be argued that Figure 4.18 does not illustrate a drawback of our framework since we are asking the SVAE to reproduce something that has not seen before or, at least, that it shares this limitation with any VAE. However, any human could easily extrapolate the results from Figure 4.17 to Figure 4.18 and it would be desirable for the SDE SVAE to be able to do so. The inability of the SDE SVAE to properly extrapolate may be due to the use of the squared exponential kernel function. This kernel was conveniently chosen to model the drift since it permits computing the integrals from Section 4.5.2 in closed form. However, the squared exponential kernels are only able to extrapolate around "small" regions around the inducing-points, where "small" is defined in terms of the length-scale hyperparameter. Hence the inability of our technique to predict the evolution of the system when there has been a

significant change in the amplitude.

Figure 4.26 illustrates the other drawback of the SDE SVAE. In this example, the synthetic signals seem like smoothed versions of true realizations of the stochastic Lorenz process. It is known that image-generating VAEs that use Gaussian likelihoods as loss functions yield blurry images [32]. This is usually attributed to the fact that the Gaussian likelihood does not permit to capture the true distribution of the images, which it is usually multimodal. In this sense, it can be argued that our lower bound is also failing to capture the diffusion-like properties of the input series. Although suggestive, this is an speculative hypothesis that requires further validation.

Finally, it must be noted that the model proposed in this chapter has still room for improvement within the SVAE framework. As previously introduced, the inclusion of latent modes would permit describing the data in terms of switching SDEs, enabling the application of the method to virtually any time series, including those with highly nonlinear non-stationary dynamics.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 5

CONCLUSIONS

Nowadays, there is a great interest in the scientific community on the dynamical behavior of complex systems in biology, physics, chemistry, economy, psychology or sociology, among many others. As a consequence of the complexity of these dynamical phenomena, there is a trend towards devising more sophisticated time series models. These models should be able to account for nonlinear behavior, the interaction of a large number of subsystems, or even the presence of stochastic features in the laws governing the evolution of the system. In this context, developing a model from first principles is a challenge. An alternative strategy is to build a model based on observed data from the system. It seems reasonable to assume that, with the big data revolution, this strategy will become increasingly common. This also poses an enormous challenge to time series analysis, since new techniques should scale effectively with the size of the data.

The present work provides a Bayesian approach to learning plausible models capable of characterizing complex time series in which deterministic and stochastic phenomena concur. It is worth noting that, although we explore flexible models from the machine learning setting able to capture rich dynamical features, these models are firmly grounded on well-known physical concepts. We kept our models deeply rooted in physics because, in our opinion, it is through explicit model building that we gain a truly deeper understanding about how a system behaves. Therefore, a model built from data should not only give accurate predictions, but also help the researcher in taking modeling decisions, and this is easier if both the researcher and the model share the same vocabulary, the vocabulary of physics.

Two main approaches were actually developed. In the first approach, we focused on a

simple superposition model, grounded on the hypothesis that the interactions between the stochastic and deterministic phenomena are negligible. The second main approach relied on SDEs to model systems where the stochastic and deterministic part interact.

The first approach is discussed in Chapter 2. To enable the superposition model to capture complex dynamics without relying on the deterministic-stochastic interactions, the stochastic part is assumed to be a fBm. fBm is a non-stationary model frequently used to describe long-term correlations. A large number of physical and informational systems exhibit such a behavior, which supports the applicability of the model. The other component of the system is assumed to be a band-limited signal. It is likely for a band-limited signal to be accurately predictable and, therefore, we refer to it as the deterministic component. Under the assumptions of this model, we devised a method that is able to characterize the fractal stochastic component and to estimate the deterministic part. The main contributions of this Chapter are: (1) the key observation that the presence of the deterministic component causes a measurable deviation from the energy's power-law that we expect in pure fBm processes; and (2) the exploitation of this observation by means of Bayesian estimation techniques, which enable the approximate reconstruction of the deterministic part of an observed signal while characterizing its stochastic part. The method was tested on both synthetic and real signals, demonstrating its ability to capture interesting features from the data. With the aim of facilitating the reproducibility of the results, the proposed method was implemented and released under an Open Source License in the `R` package `fracdet`. The package is briefly overviewed in Appendix A. It must be noted that releasing the code as an `R` package guarantees that it can be easily installed in any operating system, and that its functions are well-documented.

Chapter 3 starts the study of systems where deterministic and stochastic components interact with each other, by exploiting the modeling capabilities of SDEs. To simplify the mathematical analysis of SDEs, the Wiener process is used as the source of the randomness of the system. Stated another way, the long-term modeling capabilities of fBms are sacrificed to introduce a simple interaction model. Indeed, our assumptions about the SDE require the time series to be Markovian. Under these hypothesis, Chapter 3 develops a non-parametric estimation method for SDEs based on GPs. The main contributions of the Chapter are: (1) providing estimates for any type of diffusion function (which required the development of a Laplace approximation) and (2) proposing a sparse approximation to the true GP posterior that permits to efficiently handle middle-sized experimental time series (size of $N \approx 10^3 - 10^5$). It must be noted that although these sizes include most typical experimental datasets, these

are rapidly increasing in size, which may limit the applicability of the method in the future. The issue is related to the use of variational inference, which requires a full pass through the entire dataset at each iteration. The proposed algorithm was validated using both simulated data and real data from economy and paleoclimatology. The application of the method to real data demonstrates its ability to capture the behavior of complex systems. We also released an open-source implementation of the method in the `R` package `voila`. A brief overview of the package is provided in Appendix B.

Chapter 4 tries to overcome the main constraint that the use of SDEs driven by a Wiener process imposes: the Markovianity assumption. To that end, we propose a new SVAE with latent SDE dynamics. The SVAE is capable of discovering a map, parametrized as a neural network, from an arbitrary time series to a Markovian representation. Through this transformation, any time series is susceptible to be analyzed by means of the methods of Chapter 3. To prevent the transformation from enforcing the fulfillment of the Markovian property by creating an overly simple representation, an inverse transformation is also sought. This prevents the forward transformation to dismiss valuable information. The inverse transformation is also parametrized with a neural network and, therefore, the pair of transformations acts as a encoder-decoder neural network. Following the SVAE architecture, the latent space is constrained to follow a SDE through the use of graphical models. To permit a better optimization, the learning of both the transformations and the dynamics is performed simultaneously. Finally, the other major drawback of the method of Chapter 3 is addressed: the inference procedure is scaled up to handle large datasets that do not fit in main memory by applying stochastic optimization techniques to variational inference, which is usually known as SVI. The main contributions of this chapter are: (1) leveraging SVAEs with the methods from Chapter 3, which permits modeling nonlinear dynamics without switching modes and also (2) finding a physically meaningful representation of the data that can be interpreted as a phase space; and (3) exploiting recent advances in GP dynamical systems to efficiently sample from the nonlinear SDE phase space, as required by the SVAE training algorithm. The SDE SVAE was tested using data from well-known physical models, showing that it is able to learn plausible phase spaces and to synthesise new time series with dynamical features similar to those of the original ones. We have also released code implementing the main methods of this chapter at [46].

Although the use of SDE for data-driven modeling had already been considered (see, for example, [42]), we have made some contributions to leverage the predictive performance of

Bayesian machine learning methods while keeping the physical interpretability of the models. This is particularly true in Chapter 4, in which we managed to extend the applicability of SDEs to non-Markovian time series while maintaining the model close to the physical domain by means of the phase space. The interpretability of the SDE SVAE does not only help the researcher in the elucidation of plausible models, but it may also inspire new analysis methods borrowed from other fields. For example, the phase space itself provides valuable information from a modeling perspective, and it may be subject to further research by means of phase space analysis methods or the study of attractors [71].

Still, the present work barely scratches the surface of the possible approaches for modeling complex time series with deterministic and stochastic components. During the development of this thesis, a number of important issues were left open. The main research directions that may be worth exploring are outlined below:

- An important drawback of the method introduced in Chapter 2 is that, to get an estimate of the energy of the deterministic component, the researcher must select the wavelet resolution levels to use in the regression procedure. This is a subjective step that depends on the experience of the researcher, hindering reproducibility. When CGSA assumptions hold, the selection of these levels could be avoided by using the CGSA algorithm. However, if those assumptions are not met, there is a lack of tools for objectively estimating those features of a fractal time series that does not uniquely contain fractal components. Therefore, the design of some mechanism to automatically select the wavelet resolution levels free from deterministic contributions should be addressed in future improvements. To that end, simple methods such as segmented regression combined with model selection criteria may serve as a good starting point.

- A number of works have tried to analyze SDEs driven by richer types of noise, including fBm. See, for example, [80]. Therefore, an interesting line of work would be to generalize the SGP estimation method from Chapter 3 to permit arbitrary noises to act as the stochastic force driving the SDE. This is very suggestive, since it would permit relating the model from Chapter 2 with the SDE model from Chapter 3, and it would also address the main limitation from the model from Chapter 2: it ignores the interactions between the deterministic and stochastic components.

- Regarding the methods from Chapter 3, we would like to design some criteria to automatically estimate an appropriate number of pseudo-inputs m taking into account

both the modified lower bound (Equation (3.23)) and the additional computational time required when m increases. The substitution of the exponential transformation used to ensure the positiveness of g in favor of other less-explosive transformation should also be considered in future improvements. The numerical stability of the method would certainly improve with the use of smoother transformations, but the formal expressions required for the variational inference problem would be more complicated. An alternative to avoid the complicated mathematical expressions would be to use black-box variational inference frameworks [129]. Since black-box methods are based in stochastic variational inference, its application would also permit scaling the method to datasets much bigger than those studied in Chapter 3.

- With the aim of supporting reproducible research, the source code of the SDE SVAE from Chapter 4 has been published in [46]. However, this implementation is not as mature as in the case of `fracdet` or `voila`. More effort should be made in order to provide an easy-to-use package that helps in the adoption of the SDE SVAE model.

- In Chapter 4 we have argued that the incapacity of the SDE SVAE to extrapolate the learned dynamics were due to the constraints that the squared exponential kernels impose to the GP modeling of the drift function. It follows that, to improve the generalization properties of the drift estimates, we should endow the SDE SVAE with the possibility of using different kernels, or even a combination of them. Indeed, sophisticated kernels are usually hand-crafted to enable pattern discovery and extrapolation in specialized applications [135]. The extension of the methods presented in Chapter 4 to handle richer covariance functions should be simple for those kernels consisting on combinations of squared exponentials, trigonometric functions, and polynomials; since the integrals from Section 4.5.2 can still be calculated analytically [27]. However, other kernels may require the development of new approximation methods for preserving the efficiency of the filtering and sampling steps from Section 4.5.2.

- We have also speculated that the reason for the smoothness of the synthetic samples of the Lorenz system could be due to the lower bound failing to capture the diffusion-like properties of the input series. This hypothesis should be confirmed through and exhaustive study. If confirmed, a modification of the lower bound to alleviate the problem should be sought.

- The inclusion of latent mode sequences in the SDE SVAE should be considered (see Section 4.7). The inclusion of latent modes would permit the description of the data in terms of switching SDEs, enabling the application of the method to virtually any time series, including those with highly nonlinear non-stationary dynamics. Furthermore, the successful consecution of this work would entail a substantial enhancement of the SLDS SVAE. Although the modeling capabilities would probably be quite similar, in a switching SDE model the different modes would have a very precise meaning, i.e., a change of the dynamics due to non-stationarity. On the other hand, a change in the mode of a SLDS may be due to the non-stationary conditions or due to the need to approximate nonlinear dynamics through switching linear dynamics.

- Finally, we have argued that the phase space of the SDE SVAE could be used for characterizing the dynamics of the system by means of phase space analysis methods. These techniques have been successfully applied to many different nonlinear deterministic systems from a wide variety of fields. However, a key to their widespread use is the existence of the Takens' theorem. The Takens' theorem guarantees that some well-defined quantities (invariants) yield the same result when measured on the true phase space and in the reconstructed one. This result forms a bridge between nonlinear theory and the analysis of experimental data. Therefore, it is necessary to explore under which circumstances we may expect the SDE SVAE to yield a phase space topologically equivalent to the original one (at least, in the deterministic case).

APPENDIX A

THE FRACDET PACKAGE

fracdet is an R package implementing the deterministic-stochastic model proposed in Chapter 2 and available at [44]. The package also implements the estimation method that, under the assumptions of the model, is able to characterize the fractal stochastic component and to provide an estimation of the deterministic component present in a given time series. In this Appendix, we provide a brief overview of how to use fracdet, motivated by a few simple examples.

A.1 Installation

Running fracdet requires R version 3.0.0 or higher. The source code for fracdet is hosted on GitHub at http://github.com/citiususc/fracdet, distributed under the free software license GPL-3. fracdet can be installed directly from this repository using the tools provided by the devtools package, as detailed in Listing A.1.

Listing A.1: Installation of fracdet from github.

```
library("devtools")
install_version("FGN", "2.0-12")
install_github("citiususc/fracdet")
```

fracdet depends on several R packages which are automatically installed, with the exception of the FGN package, which is no longer maintained at the Comprehensive R Archive Network (CRAN). That's why we need to explicitly install it in Listing A.1. The remaining dependencies are: wavethresh, MASS, Rcpp, RcppArmadillo and RcppGSL. The

RcppGSL links against the GNU Scientific Library (GSL) and, therefore, requires the GSL library to be installed in the system. In a Debian-based system, the GSL library can be easily installed using the command listed in Listing A.2.

Listing A.2: Installation of the GSL library in a Debian system.

```
sudo apt-get install gsl-bin libgsl0-dev
```

A.2 The waveletVar class

To introduce the basics of fracdet we first consider a simple stochastic series with long-term dependencies:

$$Y[n] = B[n].$$

According to the convenient modeling approach proposed in [36], long-memory signals can be studied as realizations of one of two classes of processes: dfGn or dfBm. Since the increments of a dfBm signal yield a stationary dfGn signal, we may focus on the dfBm model. Thus, we assume that $B[n]$ can be approximated by a dfBm.

As discussed in Chapter 2, wavelet analysis provides a proper framework for studying dfBm signals. Furthermore, it is possible to estimate the dfBm parameters by studying how the wavelet coefficients' variance depend on the resolution level. fracdet provides the waveletVar class to represent and manipulate the wavelet coefficients' variances. Listing A.3 illustrates its use by computing the wavelet coefficients' variance of a simulated dfBm with parameters $H = 0.3$ and $\sigma^2 = 1$. Note that the resulting plot of Listing A.3 is not shown.

Listing A.3: Computation of the wavelet coefficient's variance using fracdet.

```
library("fracdet")
# Simulate a fBm with Hurst exponent H = 0.3 and 2 ^ 14 samples
b = fbmSim(n = 2 ^ 14, H = 0.3)
# Compute the wavelet decomposition (wd) of the B[n] time series.
# We use the symmetric boundary conditions (bc)
wb = wd(b, bc = "symmetric")
# Compute the wavelet coefficients' variances and plot them
vpr = waveletVar(wb)
plot(vpr)
```

A.3 dfBm parameter estimation

The shape and slope of the wavelet coefficients' variance depends on the Hurst exponent of the dfBm. To estimate it, `fracdet` provides the `estimateFbmPars` function, which is based on the `R` implementation of the (weighted) Nonlinear Least Squares (NLS) method. Listing A.4 applies this function to the signal generated in Listing A.3. Note that the resulting estimates of the dfBm parameters are close to the real ones.

Listing A.4: H and σ^2 estimation of a dfBm process using the `fracdet` package.

```
# We can easily estimate the parameters of a dfBm using
# a waveletVar object
model = estimateFbmPars(vpr, use_resolution_levels = c(5:13))
# model is a nls object, which represents a nonlinear least
# squares regression in R. Therefore, we can obtain the
# parameter estimations with the usual R commands
print(coef(model))
## ==>         H    sigma2
## ==> 0.2954949 1.0121385
```

A.4 The `fracdet` class

Let us now assume that the observed signal $Y[n]$ is the result of the superposition of a stochastic series $B[n]$ and some band-limited deterministic signal $x[n]$, as assumed in Chapter 2:

$$Y[n] = x[n] + B[n].$$

The first step when analyzing any observed time series with the method proposed in Chapter 2 should be testing if the wavelet variances are compatible with the assumptions of the model. Listing A.5 simulates a time series consisting of a dfBm (with $H = 0.3$ and $\sigma^2 = 1$) and a sinusoidal signal, and then computes the wavelet coefficients' variance to ensure that they approximately follow the expected power law behavior. The resulting wavelet coefficient's variance should be similar to the black points shown in Figure A.1.

Listing A.5: H and σ^2 estimation of a dfBm process using the `fracdet` package.

```
# Simulate a signal consisting of a dfBm and a sinusoidal term
n = 2 ^ 14 # length of the signal
x = 2 * sin(2 * pi * 1:n / 10)
```

```
y = fbmSim(n = n, H = 0.3) + x
# Compute the wavelet coefficients variances and plot them
wy = wd(y, bc = "symmetric")
vpr = waveletVar(wy)
plot(vpr)
```

Figure A.1: Wavelet coefficients' variance of a signal consisting of a dfBm and a sine obtained with `fracdet`. Note the deviation from the expected behavior on levels 11 and 12.

Figure A.1 clearly shows that there is a deviation from the power law behavior that we could expect if $Y[n]$ was a pure dfBm. Therefore, we may conclude that this is likely due to the presence of some deterministic component. Figure A.1 also manifests that this deterministic component has its energy well concentrated around the wavelet resolution levels 11 and 12. Hence, it should be possible to obtain accurate estimates of the dfBm parameters. Listing A.6 uses the `estimateFbmPars` function to perform this estimation, taking into account that the resolution levels 11 and 12 should be ignored.

The `fracdet` class can now be used to ease the procedure of working with a signal that fulfils the assumptions of our deterministic-stochastic model. The parameters required for the `fracdet` constructor are the wavelet transform of the observed signal, and the fitted-model object used for estimating the dfBm parameters. Listing A.6 uses this class and its associated methods to check that the dfBm parameter estimates are still accurate, and to generate a plot with the experimental and fitted wavelet variances. This plot should be similar to Figure A.1.

Listing A.6: H and σ^2 estimation of a dfBm process using the fracdet package.

```
# Ignore resolution levels 11 and 12 to obtain the
# fBm parameters estimates.
model = estimateFbmPars(vpr, use_resolution_levels = c(5:10,13))
# Create a fracdet object...
fd = fracdet(wy, model)
# ... and check the fbm parameters estimates.
print(coef(fd))
## ==>         H    sigma2
## ==> 0.3051759 1.0353774
# A plot of the fitted variances may also be useful.
plot(getWaveletVar(fd))
points(getFittedWaveletVar(fd), col = 2, pch = 2)
legend("topright", c("Experimental variance", "Fitted variance"),
       col = 1:2, pch = 1:2, bty="n")
```

By fitting the wavelet variances we have not only characterized the dfBm process, but we have also gained information about the deterministic contributions in each wavelet resolution level. We shall use this information to obtain an estimation of the deterministic signal.

A.5 Estimation of the deterministic signal

Using all the available information about the observed signal, a Bayesian model of the distribution of the deterministic and stochastic wavelet coefficients is built. The available information consists of:

- The estimates of the dfBm parameters (which completely characterize the statistical properties of the dfBm signal).
- The well-known statistical properties of the dfBm signals in the wavelet domain.
- The deviations from the theoretical wavelet coefficients' variance permit the estimation of the energy distribution across the resolution levels of the deterministic signal.

Using this information, the wavelet transform of the deterministic signal is estimated by calculating the decision rule that minimizes the posterior expected value of a squared loss function (see Section 2.1.2). The estimateDetSignal implements all these steps and returns the estimate of the wavelet transform of the deterministic signal, as illustrated in

Listing A.7. Note that `estimateDetSignal` expects as input an object of the `fracdet` class.

Listing A.7: H and σ^2 estimation of a dfBm process using the `fracdet` package.

```
# Estimate the deterministic signal (this may take a while) taking
# into account the deviations in level 11, 12.
wx = estimateDetSignal(fd, estimate_from = 11:12)
x_est = wr(wx)
```

Figure A.2 compares the real $x[n]$ with the estimate obtained in Listing A.7.

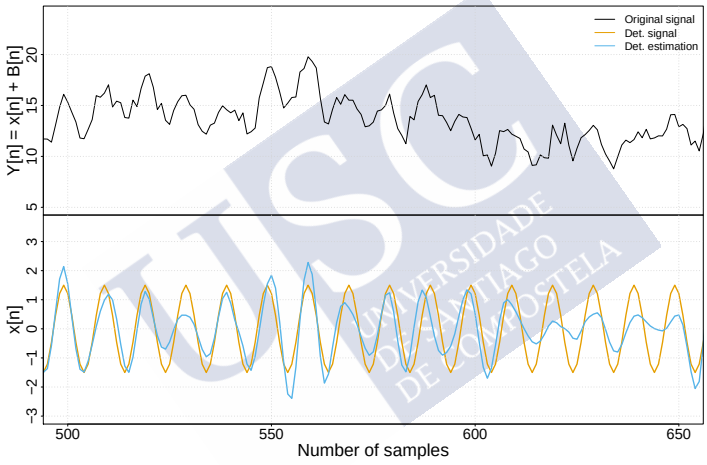

Figure A.2: Deterministic estimation obtained with `fracdet`. Top panel: simulated signal consisting of a dfBm and a sine. Bottom panel: real deterministic signal $x[n]$ and the estimate obtained using the `fracdet` package.

APPENDIX B

THE voila PACKAGE

voila is an R package for the non-parametric estimation of Langevin equations from a densely observed time series [45]. This package not only implements the SGP method proposed in Chapter 3, but also the KBR and POLY methods. In this Appendix, we focus on the functions included in voila to perform the SGP-based estimation of SDE terms. We use a simple example to guide the discussion: the estimation of the drift and diffusion functions of an Ornstein-Uhlenbeck process.

B.1 Installation

Running voila requires R version 3.0.0 or higher. The source code for voila is hosted on GitHub at https://github.com/citiususc/voila, distributed under the free software license GPL-3. voila can be installed directly from this repository using the tools provided by the devtools package, as detailed in Listing B.1.

Listing B.1: Installation of voila from github.

```
library("devtools")
install_github("citiususc/voila")
```

voila depends on several R packages which are automatically installed when using Listing B.1. The required dependencies are: yuima, methods, KernSmooth, doParallel, foreach, zoo, Rcpp and RcppArmadillo.

B.2 An usage example

In this Section, we use `voila` to estimate the drift and diffusion terms of an unidimensional Ornstein-Uhlenbeck model from a single realization of the process. Listing B.2 shows how to simulate a SDE using the `voila simulate_sde` function. This function provides a simplified interface to the excellent SDE simulation tools included in the `Yuima` package [16]. The simulated Ornstein-Uhlenbeck process is shown in Figure B.1.

Listing B.2: Simulation of an Ornstein-Uhlenbeck process using `voila`.

```
library("voila")
# Simulate a Ornstein-Uhlenbeck time series using voila
drift = "-x"
diffusion = "sqrt(1.5)"
x = simulate_sde(drift, diffusion, samplingPeriod = 0.001, tsLength = 20000)
plot(seq(0, len = length(x), by = samplingPeriod), x, type = "l",
     ylab = "x(t)", xlab = "Time t", main = "Ornstein-Uhlenbeck process")
```

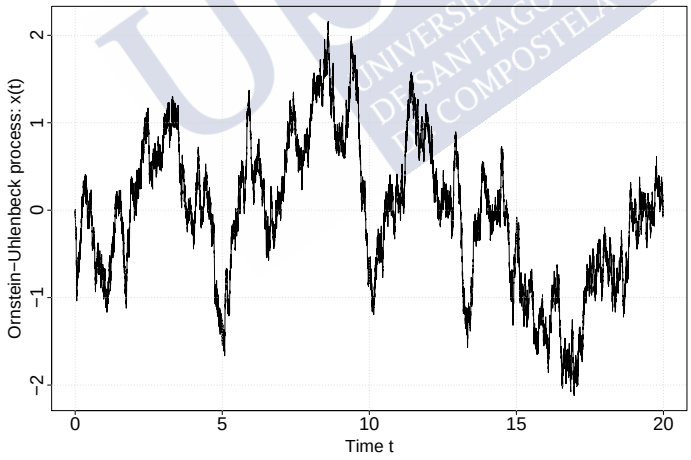

Figure B.1: Ornstein-Uhlenbeck process simulated with the `voila` package.

B.2.1 Selection of the GPs' kernels

The first step before running the variational inference routine is to define the covariance kernels we want to use for the drift and diffusion models. For our example, we choose a

rational quadratic kernel for the drift, and a linear combination of a constant kernel and a squared exponential kernel for the diffusion. The amplitudes of the diffusion kernels are constrained to sum to a constant, as described by Equation (3.26). Although the kernels determine the most important properties of the resulting GPs, our selection is not guided by any objective reason, but it is intended to illustrate some of the possible kernels available in the package.

All kernels are created using the `sde_kernel` function, as illustrated in Listing B.3. Besides the type of the kernel, the input dimension, its hyperparameters and a regularization constant are mandatory parameters. The regularization constant is a small value that can be added to the diagonal of the kernel covariance matrix to improve the stability of the computations, as described in Equation (3.27).

As discussed in Section 3.4.1, the selection of some of the hyperparameters of the diffusion kernel is not straightforward. This is due to the fact that the GP related to the diffusion is not directly modeling it, but its logarithm, i.e., $g(x) = \exp\left(s(x)\right)$. To facilitate the selection of these hyperparameters, some heuristics were derived in Section 3.4.1 (see Equation (3.24)). The `select_diffusion_parameters` function implements such heuristics. This function receives as arguments the time series, its sampling frequency and a prior belief about the variance of $g(x)$; and returns proper values for the kernel hyperparameters. See Listing B.3 for an example.

Listing B.3: Definition of the GPs' kernels using `voila`.

```
# The dimensionality of the data
inputDim = ncol(x)
# Our prior belief about the amplitude of the drift and diffusion functions
functionsUncertainty = 5
# A small value to be added to the diagonal of the covariance matrix for
# stability purposes
epsilon = 1e-5
# Create the kernels defining the behavior of the gaussian processes.
# Create a Rational Quadratic Kernel for the drift with some initial values
# for the hyperparameters. These hyperparameters will be optimized during
# the inference process.
driftKer = sde_kernel("rq_kernel",
                      list("amplitude" = functionsUncertainty,
                           "alpha" = 1,
                           "lengthScales" = 1.5),
                      inputDim, epsilon)
```

```
# Create an Exponential kernel + constant term for the diffusion:
# Voila uses a lognormal prior to ensure the positiveness of the diffusion.
# The "select_diffusion_parameters" function permits selecting a proper
# amplitude for the  kernel from our prior belief about the amplitude of
# the diffusion function. It also selects a mean value for the lognormal
# distribution (denoted with v in the theoretical exposition)
diffParams = select_diffusion_parameters(x, samplingPeriod,
                                         priorOnSd = functionsUncertainty)
diffKer =  sde_kernel("exp_const_kernel",
                      list("maxAmplitude" = diffParams$kernelAmplitude,
                           "expAmplitude" = diffParams$kernelAmplitude * 1e-3,
                           "lengthScales" = 1.5),
                      inputDim, epsilon)
```

B.2.2 Running variational inference and checking the results

After defining the kernels for the drift and diffusion models, we should also specify the number of inducing points to be used for the SGP approximation. To do that, we only have to initialize a variable representing the initial positions of the pseudo-inputs. In Listing B.4, we initialize them on a uniform grid between the minimum and the maximum values of the time series.

After completing the definition of our model, we can run the variational inference routine using the `sde_vi` function, as shown in Listing B.4.

Listing B.4: Running variational inference with `voila` to estimate the SDE parameters.

```
## Some parameters for the algorithm:
# The number of inducing points
noInducingPoints = 10
# Since the data is 1D, we shall infer the equations for the 1st
# (and unique) dimension:
targetIndex = 1
# Select some initial pseudo-inputs. The positions of the pseudo-inputs
# are optimized during the inference
pseudoInputs = matrix(seq(min(x), max(x), len = noInducingPoints), ncol = 1)

# Perform the variational inference (VI)
inference = sde_vi(targetIndex, x, samplingPeriod, pseudoInputs,
                   driftKer, diffKer, diffParams$v)
## ==>  Starting Variational Inference
## ==>  Initial Lower Bound L = -59222.9
## ==>  Iteration 1| Distributions update | L = 36691.347
```

```
## ==>  Iteration 1| Hyperparameter optimization | L = 36691.904
## ==>
## ==>  Iteration 2| Distributions update | L = 36698.412
## ==>  Iteration 2| Hyperparameter optimization | L = 36698.44
## ==>
## ==>  Iteration 3| Distributions update | L = 36698.455
## ==>  Iteration 3| Hyperparameter optimization | L = 36698.475
## ==>
## ==>  CONVERGENCE
```

The results of the inference routine are stored in the `inference` variable, which contains all the information required to assess the convergence of the variational inference, or generating predictions for the drift and diffusion terms. This is illustrated in Listing B.5. The resulting plot of the lower bound is shown in Figure B.2. It is worth noting that the predictions of both the drift and diffusion terms are obtained from their posterior distributions. These posterior distributions are represented with the `voila` class `sgp_sde`, which permits generating the predictions with the generic `R` function `predict`. Both `inference$drift` and `inference$diffusion` are instances of this class.

Listing B.5: Prediction of the drift and diffusion functions based on the posterior distributions.

```
# Check convergence
plot(inference$likelihoodLowerBound[-1], type = "o",
     main = "Likelihood Lower Bound",
     ylab = "Lower Bound", xlab = "Iteration")

# Get the estimations for the drift
predictionSupport = matrix(
        seq(quantile(x,0.05), quantile(x,0.95), len = 100),
        ncol = 1)
driftPred = predict(inference$drift, predictionSupport)

# Get the estimations for the diffusion
# The diffusion uses a lognormal gaussian process,
# so we must specify log = TRUE
diffPred = predict(inference$diffusion, predictionSupport, log = TRUE)
```

The `driftPred` and `diffPred` variables obtained from `predict` in Listing B.5 are also instances of another `voila` class: `sde_prediction`. This class facilitates working with the predictions of the SDE terms. For example, Listing B.6 illustrates how to obtain nice representations of the drift and diffusion estimates by simply using the generic `plot` function.

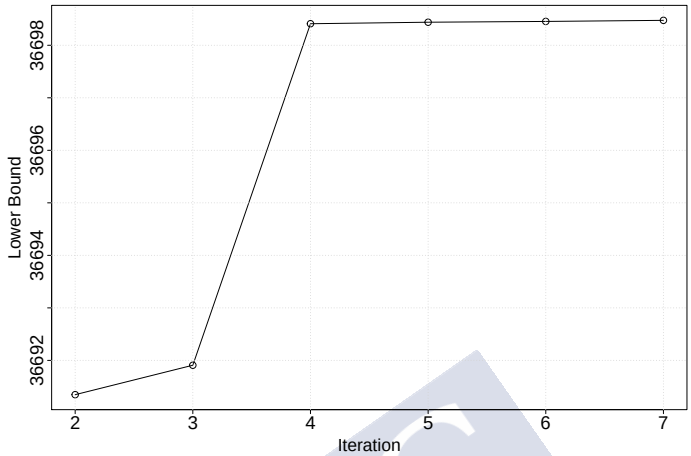

Figure B.2: The lower bound optimized by the variational inference routine.

The resulting plot of the estimates is shown in Figure B.3, which highlights the accuracy of the SGP estimation procedure.

Listing B.6: Plotting the probabilistic predictions of the drift and diffusion terms with `voila`.

```
# Plot drift
plot(driftPred, xlab = "x", ylab = "Drift f(x)")
lines(predictionSupport,
      eval(parse(text = drift), list(x = predictionSupport)),
      col = 2)
legend("topright", c("Estimate", "Real"),
       lty = 1, col = 1:2, bty = "n")

# Plot diffusion
plot(diffPred, ylim = c(1, 2), xlab = "x", ylab = "Diffusion g(x)")
abline(h = eval(parse(text = diffusion), list(x = predictionSupport)) ^ 2,
       col = 2)
legend("topright", c("Estimate", "Real"),
       lty = 1, col = 1:2, bty = "n")
```

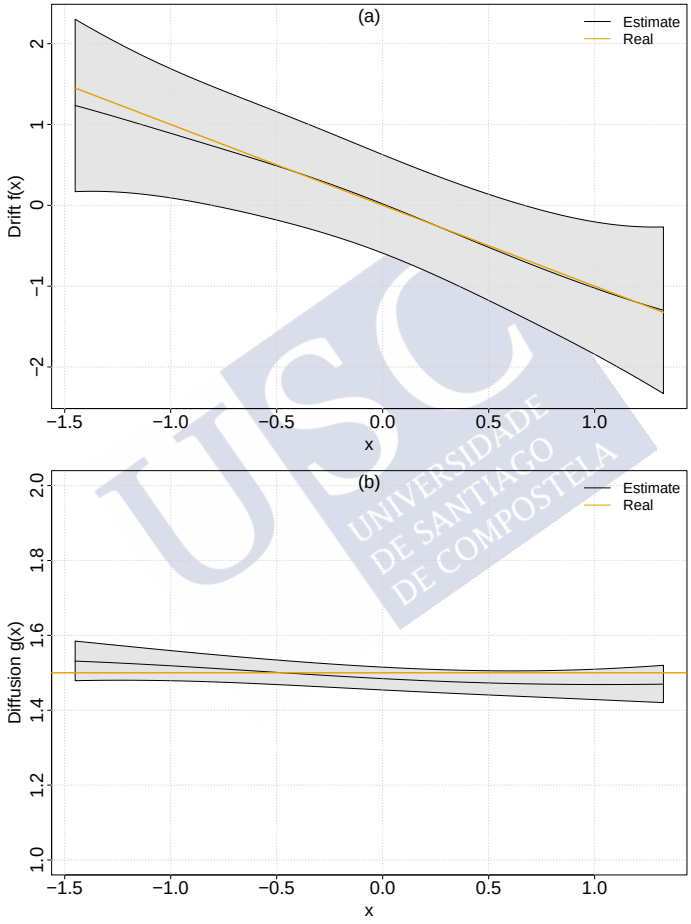

Figure B.3: Estimates of the (a) drift and (b) diffusion terms of the Ornstein-Uhlenbeck system obtained with the voila package.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Appendix C

Calculation of the Fisher matrix for the Gaussian-Gamma distribution

In this appendix, we refer to the variational distribution modeling the drift and diffusion of a single dimension from the phase space, i.e., $\tilde{\boldsymbol{f}}_i, \lambda_i$. However, to keep the notation uncluttered, we drop the index i from the equations. With the parametrization given by Equation (4.38), the logarithm of the Gaussian-Gamma distribution $q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})$ can be written as

$$\log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}}) = \begin{bmatrix} \exp(\tilde{\eta}_0) - \frac{1}{2} \\ -\exp(\tilde{\eta}_0 - \tilde{\eta}_1) - \frac{\tilde{\boldsymbol{\eta}}_2^T \tilde{\boldsymbol{\eta}}_3 \tilde{\boldsymbol{\eta}}_2}{2} \\ \tilde{\boldsymbol{\eta}}_3 \tilde{\boldsymbol{\eta}}_2 \\ -\mathrm{Vec}\left(\frac{\boldsymbol{\eta}_3}{2}\right) \end{bmatrix}^T \begin{bmatrix} \log \lambda \\ \lambda \\ \lambda \tilde{\boldsymbol{f}} \\ \lambda \mathrm{Vec}(\tilde{\boldsymbol{f}} \tilde{\boldsymbol{f}}^T) \end{bmatrix} + \\ - \left(\log \Gamma\big(\exp(\tilde{\eta}_0)\big) - (\tilde{\eta}_0 - \tilde{\eta}_1)\exp(\tilde{\eta}_0) - \frac{1}{2}\log \mid \tilde{\boldsymbol{\eta}}_3 \mid \right),$$

and the Fisher information matrix is given by

$$\boldsymbol{F}(\tilde{\boldsymbol{\eta}}) = \mathbb{E}_{\tilde{\boldsymbol{f}}, \lambda}\left[\Big(\nabla_{\tilde{\boldsymbol{\eta}}} \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})\Big)\Big(\nabla_{\tilde{\boldsymbol{\eta}}} \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})\Big)^T \right]. \tag{C.1}$$

The calculation of the gradients is straightforward although tedious. For example, the derivative with respect to $\tilde{\eta}_0$ can be computed as follows:

$$\frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial \tilde{\eta}_0} = \exp(\tilde{\eta}_0)\left[\log \lambda - \big(\psi^0\big(\exp(\tilde{\eta}_0)\big) - (\tilde{\eta}_0 - \tilde{\eta}_1)\big)\right] - [\lambda \exp(\tilde{\eta}_0 - \tilde{\eta}_1) - \exp(\tilde{\eta}_0)], \tag{C.2}$$

where $\psi^0(\cdot)$ denotes the digamma function. To ease the mathematical manipulation of Equation (C.2) we introduce $\alpha \triangleq \alpha(\tilde{\boldsymbol{\eta}}) = \exp(\tilde{\eta}_0)$ and $\beta \triangleq \beta(\tilde{\boldsymbol{\eta}}) = \exp(\tilde{\eta}_0 - \tilde{\eta}_1)$. This does not affect the expression of the Fisher matrix, since α and β should be read as functions of $\tilde{\boldsymbol{\eta}}$. Furthermore, to simplify the computation of the expectations required by Equation (C.1), we introduce

$$\mathbb{E}_{\tilde{\boldsymbol{f}},\lambda}[\log\lambda] = \psi^0(\alpha) - \log\beta$$

and

$$\mathbb{E}_{\tilde{\boldsymbol{f}},\lambda}[\lambda] = \frac{\alpha}{\beta}$$

into Equation (C.2). This yields

$$\frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial \tilde{\eta}_0} = \alpha\,[\log\lambda - \mathbb{E}\log\lambda] - \beta\,[\lambda - \mathbb{E}\lambda], \tag{C.3}$$

where we have written $\mathbb{E}_{\tilde{\boldsymbol{f}},\lambda}[\cdot]$ as $\mathbb{E}[\cdot]$ to keep the notation uncluttered.

We proceed similarly to compute the other derivatives required to compute the gradients from Equation (C.1), obtaining

$$\begin{aligned}
\frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial \tilde{\eta}_1} &= \beta\,[\lambda - \mathbb{E}\lambda], \\
\frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial \tilde{\boldsymbol{\eta}}_2} &= \lambda \cdot \boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{f}} - \boldsymbol{\mu}), \\
\frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial (\tilde{\boldsymbol{\eta}}_3)_{ij}} &= -\frac{\lambda}{2}\left[(\tilde{f}_i - \mu_i)(\tilde{f}_j - \mu_j)\right] + \frac{1}{2}\Sigma_{ij}.
\end{aligned} \tag{C.4}$$

To compute the Fisher matrix, we take expectations on the derivatives from Equations (C.3) and (C.4). We use subindices to denote the different entries of the matrix, depending on which derivatives are being used. For example, $F_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_3}$ denotes the $1 \times d$ matrix (being d the phase space dimension) that results from calculating

$$\boldsymbol{F}_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_3} = \mathbb{E}\left[\left(\nabla_{\tilde{\eta}_0}\log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})\right)\left(\nabla_{\tilde{\boldsymbol{\eta}}_3}\log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})\right)^T\right].$$

The calculations involving $\tilde{\eta}_0$ are straightforward:

$$\begin{aligned}
F_{\tilde{\eta}_0,\tilde{\eta}_0} &= \mathbb{E}\left[\frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial \tilde{\eta}_0} \frac{\partial \log q(\tilde{\boldsymbol{f}}, \lambda, \mid \tilde{\boldsymbol{\eta}})}{\partial \tilde{\eta}_0}\right] \\
&= \mathbb{E}\left[\Big(\alpha\big(\log\lambda - \mathbb{E}\log\lambda\big) - \beta\big(\lambda - \mathbb{E}\lambda\big)\Big)^2\right] \\
&= \alpha^2\mathbb{E}\left[\big(\log\lambda - \mathbb{E}\log\lambda\big)^2\right] - 2\alpha\beta\cdot\mathbb{E}\left[(\log\lambda - \mathbb{E}\log\lambda)(\lambda - \mathbb{E}\lambda)\right] + \beta^2\mathbb{E}\left[\big(\lambda - \mathbb{E}\lambda\big)^2\right] \\
&= \alpha^2\mathrm{Var}\left[\log\lambda\right] - 2\alpha\beta\big(\mathbb{E}\left[\lambda\log\lambda\right] - \mathbb{E}\log\lambda\mathbb{E}\lambda\big) + \beta^2\mathrm{Var}\left[\lambda\right].
\end{aligned} \tag{C.5}$$

Using the relations

$$\begin{aligned}
&\mathbb{E}\left[\log\lambda\right] = \psi^0(\alpha) - \log\beta, \\
&\mathrm{Var}\left[\log\lambda\right] = \psi^1(\alpha), \\
&\mathbb{E}\left[\lambda\right] = \frac{\alpha}{\beta}, \\
&\mathrm{Var}\left[\lambda\right] = \frac{\alpha}{\beta^2}, \\
&\mathbb{E}\left[\lambda\log\lambda\right] = \frac{\alpha}{\beta}(\psi^0(\alpha+1) - \log\beta)
\end{aligned}$$

where $\psi^1(\cdot)$ denotes the trigamma function, it is possible to write Equation (C.5) as

$$\begin{aligned}
F_{\tilde{\eta}_0,\tilde{\eta}_0} &= \alpha^2\psi^1(\alpha) - 2\alpha\beta\Big(\frac{\alpha}{\beta}(\psi^0(\alpha+1) - \log\beta) - \frac{\alpha}{\beta}(\psi^0(\alpha) - \log\beta)\Big) + \beta^2\frac{\alpha}{\beta^2} \\
&= \alpha^2\psi^1(\alpha) - 2\alpha^2\Big(\psi^0(\alpha+1) - \psi^0(\alpha)\Big) + \alpha.
\end{aligned}$$

Using the recurrence relation $\psi^0(\alpha+1) = \psi^0(\alpha) + \frac{1}{\alpha}$ finally yields

$$F_{\tilde{\eta}_0,\tilde{\eta}_0} = \alpha^2\psi^1(\alpha) - \alpha.$$

A similar computation for $F_{\tilde{\eta}_0,\tilde{\eta}_1}$ yields

$$\begin{aligned}
F_{\tilde{\eta}_0,\tilde{\eta}_1} &= \mathbb{E}\left[\big(\alpha(\log\lambda - \mathbb{E}\log\lambda) - \beta(\lambda - \mathbb{E}\lambda)\big)\big(\beta(\lambda - \mathbb{E}\lambda)\big)\right] = \\
&= \alpha\beta\,\mathbb{E}\left[(\log\lambda - \mathbb{E}\log\lambda)(\lambda - \mathbb{E}\lambda)\right] - \beta^2\mathbb{E}\left[(\lambda - \mathbb{E}\lambda)^2\right] \\
&= \alpha\beta\frac{1}{\beta} - \beta^2\frac{\alpha}{\beta^2} \\
&= 0.
\end{aligned}$$

The term $F_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_2}$ can be computed as

$$\begin{aligned} F_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_2} &= \mathbb{E}\left[\big(\alpha(\log\lambda - \mathbb{E}\log\lambda) - \beta(\lambda - \mathbb{E}\lambda)\big)\big(\lambda\tilde{\boldsymbol{\eta}}_3(\tilde{\boldsymbol{f}} - \mathbb{E}\tilde{\boldsymbol{f}})\big)\right] \\ &= \mathbb{E}_\lambda\left[\big(\alpha(\log\lambda - \mathbb{E}\log\lambda) - \beta(\lambda - \mathbb{E}\lambda)\big)\lambda\tilde{\boldsymbol{\eta}}_3\mathbb{E}_{\tilde{\boldsymbol{f}}|\lambda}\left[\tilde{\boldsymbol{f}} - \mathbb{E}\tilde{\boldsymbol{f}}\right]\right] \\ &= 0, \end{aligned}$$

where we have exploited the factorization of $q(\tilde{\boldsymbol{f}}, \lambda \mid \tilde{\boldsymbol{\eta}})$ to write the expectation required by the Fisher term as iterated expectations, and the fact that the expectation involving $\tilde{\boldsymbol{f}}$ cancels.

Proceeding in an analogous way, $F_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_3}$ can be written as

$$\begin{aligned} F_{\tilde{\eta}_0,[\tilde{\boldsymbol{\eta}}_3]_{ij}} &= -\frac{1}{2}\mathbb{E}_\lambda\left[\big(\alpha(\log\lambda - \mathbb{E}\log\lambda) - \beta(\lambda - \mathbb{E}\lambda)\big)\mathbb{E}_{\tilde{\boldsymbol{f}}|\lambda}\left[\lambda(\tilde{f}_i - \mathbb{E}\tilde{f}_i)(\tilde{f}_j - \mathbb{E}\tilde{f}_j)\right]\right] \\ &= -\frac{1}{2}\mathbb{E}_\lambda\left[\big(\alpha(\log\lambda - \mathbb{E}\log\lambda) - \beta(\lambda - \mathbb{E}\lambda)\big)\lambda\frac{1}{\lambda}\Sigma_{ij}\right] \\ &= -\frac{\Sigma_{ij}}{2}\mathbb{E}_\lambda\left[\alpha(\log\lambda - \mathbb{E}\log\lambda) - \beta(\lambda - \mathbb{E}\lambda)\right] \\ &= 0. \end{aligned}$$

We shall start now the calculations involving $\tilde{\eta}_1$. The term $F_{\tilde{\eta}_1,\tilde{\eta}_1}$ can be written as

$$\begin{aligned} F_{\tilde{\eta}_1,\tilde{\eta}_1} &= \beta^2\mathbb{E}\left[(\lambda - \mathbb{E}\lambda)^2\right] = \\ &= \beta^2\mathrm{Var}\left[\lambda\right] \\ &= \beta^2\frac{\alpha}{\beta^2} \\ &= \alpha. \end{aligned}$$

The term $F_{\tilde{\eta}_1,\tilde{\eta}_0}$ is just $F_{\tilde{\eta}_0,\tilde{\eta}_1}$ due to the symmetry of the Fisher matrix. On the other hand, it can be easily demonstrated that the terms $F_{\tilde{\eta}_1,\tilde{\boldsymbol{\eta}}_2}$ $F_{\tilde{\eta}_1,\tilde{\boldsymbol{\eta}}_3}$ are zero (matrices with all their terms equal to 0) if they are calculated in a manner similar to how it was done for $F_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_2}$ and $F_{\tilde{\eta}_0,\tilde{\boldsymbol{\eta}}_3}$.

The term $F_{\tilde{\boldsymbol{\eta}}_2,\tilde{\boldsymbol{\eta}}_2}$ can be computed as

$$\begin{aligned} F_{\tilde{\boldsymbol{\eta}}_2,\tilde{\boldsymbol{\eta}}_2} &= \mathbb{E}\left[\lambda^2\boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{f}} - \mathbb{E}\tilde{\boldsymbol{f}})(\tilde{\boldsymbol{f}} - \mathbb{E}\tilde{\boldsymbol{f}})^T\boldsymbol{\Sigma}^{-1}\right] \\ &= \boldsymbol{\Sigma}^{-1}\mathbb{E}_\lambda\left[\lambda^2\mathbb{E}_{\tilde{\boldsymbol{f}}|\lambda}\left[(\tilde{\boldsymbol{f}} - \mathbb{E}\tilde{\boldsymbol{f}})(\tilde{\boldsymbol{f}} - \mathbb{E}\tilde{\boldsymbol{f}})^T\right]\right]\boldsymbol{\Sigma}^{-1} \\ &= \boldsymbol{\Sigma}^{-1}\mathbb{E}_\lambda\left[\lambda^2\frac{1}{\lambda}\boldsymbol{\Sigma}\right]\boldsymbol{\Sigma}^{-1} \\ &= \frac{\alpha}{\beta}\boldsymbol{\Sigma}^{-1}. \end{aligned}$$

The only unknown term involving $\tilde{\boldsymbol{\eta}}_2$ that remains to be computed is $F_{\tilde{\boldsymbol{\eta}}_2,\tilde{\boldsymbol{\eta}}_3}$, which can be expressed as

$$\begin{aligned} F_{\tilde{\boldsymbol{\eta}}_2,[\tilde{\boldsymbol{\eta}}_3]_{ij}} &= \mathbb{E}\left[\left(\lambda^2\boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{f}}-\mathbb{E}\tilde{\boldsymbol{f}})\right)\left(-\frac{\lambda}{2}(\tilde{f}_i-\mathbb{E}\tilde{f}_i)(\tilde{f}_j-\mathbb{E}\tilde{f}_j)+\frac{1}{2}\Sigma_{ij}\right)\right] \\ &= -\frac{1}{2}\boldsymbol{\Sigma}^{-1}\mathbb{E}_\lambda\left[\lambda^3\mathbb{E}_{\tilde{\boldsymbol{f}}|\lambda}\left[(\tilde{\boldsymbol{f}}-\mathbb{E}\tilde{\boldsymbol{f}})(\tilde{f}_i-\mathbb{E}\tilde{f}_i)(\tilde{f}_j-\mathbb{E}\tilde{f}_j)\right]\right]. \end{aligned} \tag{C.6}$$

The Isserlis' theorem [67] states that, if (X_1, X_2, X_3, X_4) is a zero-mean Gaussian vector,then

$$\mathbb{E}[X_1X_2X_3X_4] = \mathbb{E}[X_1X_2]\,\mathbb{E}[X_3X_4] + \mathbb{E}[X_1X_3]\,\mathbb{E}[X_2X_4] + \mathbb{E}[X_1X_4]\,\mathbb{E}[X_2X_3], \tag{C.7a}$$

$$\mathbb{E}[X_1X_2X_3] = 0. \tag{C.7b}$$

Therefore, invoking Equation (C.7b) in Equation (C.6) yields

$$F_{\tilde{\boldsymbol{\eta}}_2,[\tilde{\boldsymbol{\eta}}_3]_{ij}} = \mathbf{0}.$$

Finally, the term $F_{\tilde{\boldsymbol{\eta}}_2,\tilde{\boldsymbol{\eta}}_3}$ can be written as

$$\begin{aligned} F_{[\tilde{\boldsymbol{\eta}}_3]_{ij},[\tilde{\boldsymbol{\eta}}_3]_{kl}} =& \mathbb{E}\left[\left(-\frac{\lambda}{2}(\tilde{f}_i-\mathbb{E}\tilde{f}_i)(\tilde{f}_j-\mathbb{E}\tilde{f}_j)+\frac{1}{2}\Sigma_{ij}\right)\left(-\frac{\lambda}{2}(\tilde{f}_k-\mathbb{E}\tilde{f}_k)(\tilde{f}_l-\mathbb{E}\tilde{f}_l)+\frac{1}{2}\Sigma_{kl}\right)\right] \\ =& +\frac{1}{4}\mathbb{E}\left[\left(\lambda^2(\tilde{f}_i-\mathbb{E}\tilde{f}_i)(\tilde{f}_j-\mathbb{E}\tilde{f}_j)(\tilde{f}_k-\mathbb{E}\tilde{f}_k)(\tilde{f}_l-\mathbb{E}\tilde{f}_l)\right)\right] \\ & -\frac{1}{4}\mathbb{E}\left[\left(\lambda(\tilde{f}_i-\mathbb{E}\tilde{f}_i)(\tilde{f}_j-\mathbb{E}\tilde{f}_j)\Sigma_{kl}\right)\right] \\ & -\frac{1}{4}\mathbb{E}\left[\left(\lambda\Sigma_{ij}(\tilde{f}_k-\mathbb{E}\tilde{f}_k)(\tilde{f}_l-\mathbb{E}\tilde{f}_l)\right)\right] \\ & +\frac{1}{4}\Sigma_{ij}\Sigma_{kl} \\ =& +\frac{1}{4}\mathbb{E}\left[\left(\lambda^2(\tilde{f}_i-\mathbb{E}\tilde{f}_i)(\tilde{f}_j-\mathbb{E}\tilde{f}_j)(\tilde{f}_k-\mathbb{E}\tilde{f}_k)(\tilde{f}_l-\mathbb{E}\tilde{f}_l)\right)\right]-\frac{1}{4}\Sigma_{ij}\Sigma_{kl}. \end{aligned}$$

Using Equation (C.7a) from the Isserlis' theorem to expand the first term of the right hand side finally yields

$$F_{[\tilde{\boldsymbol{\eta}}_3]_{ij},[\tilde{\boldsymbol{\eta}}_3]_{kl}} = \frac{1}{4}\left[\Sigma_{ik}\Sigma_{jl}+\Sigma_{il}\Sigma_{jk}\right].$$

Therefore, the final Fisher matrix can be written as

$$\boldsymbol{F}(\tilde{\boldsymbol{\eta}}) = \begin{bmatrix} \alpha^2\psi^{(1)}(\alpha)-\alpha & 0 & \mathbf{0} & \mathbf{0} \\ 0 & \alpha & \mathbf{0} & \mathbf{0} \\ 0 & 0 & \frac{\alpha}{\beta}\boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ 0 & 0 & \mathbf{0} & F_{\tilde{\boldsymbol{\eta}}_3,\tilde{\boldsymbol{\eta}}_3} \end{bmatrix},$$

where $F_{\tilde{\boldsymbol{\eta}}_3,\tilde{\boldsymbol{\eta}}_3}$ should be read as the submatrix that results from arranging the entries from $F_{[\tilde{\boldsymbol{\eta}}_3]_{ij},[\tilde{\boldsymbol{\eta}}_3]_{kl}}$ to match the ordering of the vectorized parameter $\mathrm{Vec}(\tilde{\boldsymbol{\eta}}_3)$.

Bibliography

[1] Felix Abramovich, Theofanis Sapatinas, and Bernard W. Silverman. Wavelet thresholding via a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(4):725–749, 1998.

[2] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[3] Katrine K. Andersen, N. Azuma, J.M. Barnola, Matthias Bigler, P. Biscaye, N. Caillon, J. Chappellaz, Henrik Brink Clausen, Dorthe Dahl-Jensen, Hubertus Fischer, et al. High-resolution record of Northern Hemisphere climate extending into the last interglacial period. *Nature*, 431(7005):147–151, 2004.

[4] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[5] Alireza Bahraminasab, Fatemeh Ghasemi, Aneta Stefanovska, Peter V.E. McClintock, and R. Friedrich. Physics of brain dynamics: Fokker–Planck analysis reveals changes in EEG δ–θ interactions in anæsthesia. *New Journal of Physics*, 11(10):103051, 2009.

[6] Jan Beran. *Statistics for long-memory processes*, volume 61. CRC Press, 1994.

[7] James O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, second edition, 2013.

[8] Theo Berger. A wavelet based approach to measure and manage contagion at different time scales. *Physica A: Statistical Mechanics and its Applications*, 436:338–350, 2015.

[9] José M. Bernardo and Adrian F.M. Smith. *Bayesian theory*. IOP Publishing, 2001.

[10] Francesca Biagini, Yaozhong Hu, Bernt Øksendal, and Tusheng Zhang. *Stochastic calculus for fractional Brownian motion and applications*. Springer, 2008.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[12] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[13] Ludwig Boltzmann. *Wissenschaftliche abhandlungen*, volume 1. Cambridge University Press, 2012.

[14] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988.

[15] Richard J. Boys, Darren J. Wilkinson, and Thomas B.L. Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18(2):125–135, 2008.

[16] Alexandre Brouste, Masaaki Fukasawa, Hideitsu Hino, Stefano M. Iacus, Kengo Kamatani, Yuta Koike, Hiroki Masuda, Ryosuke Nomura, Teppei Ogihara, Yasutaka Shimuzu, Masayuki Uchida, and Nakahiro Yoshida. The YUIMA project: A computational framework for simulation and inference of stochastic differential equations. *Journal of Statistical Software*, 57(4):1–51, 2014.

[17] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[18] Camillo Cammarota and Mario Curione. Modeling trend and time-varying variance of heart beat RR intervals during stress test. *Fluctuation and Noise Letters*, 10(02):169–180, 2011.

[19] Joaquín Quiñonero Candela, Agathe Girard, Jan Larsen, and Carl Edward Rasmussen. Propagation of uncertainty in Bayesian kernel models-application to multiple-step ahead

forecasting. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003 (ICASSP'03)*, volume 2, pages II–701. IEEE, 2003.

[20] Hugh A. Chipman, Eric D. Kolaczyk, and Robert E. McCulloch. Adaptive Bayesian wavelet shrinkage. *Journal of the American Statistical Association*, 92(440):1413–1421, 1997.

[21] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[22] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

[23] John C. Cox, Jonathan E. Ingersoll Jr., and Stephen A. Ross. A theory of the term structure of interest rates. *Econometrica: Journal of the Econometric Society*, pages 385–407, 1985.

[24] Alessandro De Gregorio and Stefano M. Iacus. Adaptive Lasso-type estimation for multivariate diffusion processes. *Econometric Theory*, 28(04):838–860, 2012.

[25] Comisión Nacional de la Energía. Informe sobre el efecto del día de la semana en la determinación de los precios de los carburantes (periodo 2007-2012)., March 2013. Available at `https://www.cnmc.es/Portals/0/Ficheros/Promocion/Informes_y_Estudios_Sectoriales/2012/2012_CNMC_InformeEfectoSemanaPreciosCarburantes.pdf`.

[26] Marc Peter Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic moment-based Gaussian process filtering. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 225–232. ACM, 2009.

[27] Marc Peter Deisenroth, Ryan Darby Turner, Marco F. Huber, Uwe D. Hanebeck, and Carl Edward Rasmussen. Robust filtering and smoothing with Gaussian processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871, 2012.

[28] Peter D. Ditlevsen, Katrine K. Andersen, and Anders Svensson. The DO-climate events are probably noise induced: statistical investigation of the claimed 1470 years cycle. *Climate of the Past*, 3(1):129–134, 2007.

[29] Peter D. Ditlevsen and Ove D. Ditlevsen. On the stochastic nature of the rapid climate shifts during the last ice age. *Journal of Climate*, 22(2):446–457, 2009.

[30] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[31] David L. Donoho and Iain M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

[32] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[33] Eugene Dubossarsky, Jerome H. Friedman, John T. Ormerod, and Matthew P. Wand. Wavelet-based gradient boosting. *Statistics and Computing*, 26(1-2):93–105, 2016.

[34] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[35] Albert Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der Physik*, 322(8):549–560, 1905.

[36] Andras Eke, Peter Herman, László Kocsis, and Lajos R. Kozak. Fractal characterization of complexity in temporal physiological signals. *Physiological Measurement*, 23(1):R1, 2002.

[37] Oliver Faust, U. Rajendra Acharya, Hojjat Adeli, and Amir Adeli. Wavelet-based EEG processing for computer-aided seizure detection and epilepsy diagnosis. *Seizure*, 26:56–64, 2015.

[38] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman lectures on physics, Vol. I: The new millennium edition: mainly mechanics, radiation, and heat*, volume 1. Basic books, 2011.

[39] Patrick Flandrin. Wavelet analysis and synthesis of fractional Brownian motion. *IEEE Transactions on Information Theory*, 38(2):910–917, 1992.

[40] Rudolf Friedrich and Joachim Peinke. Description of a turbulent cascade by a Fokker-Planck equation. *Physical Review Letters*, 78(5):863, 1997.

[41] Rudolf Friedrich and Joachim Peinke. Statistical properties of a turbulent cascade. *Physica D: Nonlinear Phenomena*, 102(1):147–155, 1997.

[42] Rudolf Friedrich, Joachim Peinke, Muhammad Sahimi, and M. Reza Rahimi Tabar. Approaching complexity by stochastic methods: From biological systems to turbulence. *Physics Reports*, 506(5):87–162, 2011.

[43] Luca Gammaitoni, Peter Hänggi, Peter Jung, and Fabio Marchesoni. Stochastic resonance. *Reviews of Modern Physics*, 70(1):223, 1998.

[44] Constantino A. García. `fracdet`: estimation of deterministic and fractal components in time series. `https://github.com/citiususc/fracdet`, 2016.

[45] Constantino A. García. `voila`: variational inference for Langevin equations. `https://github.com/citiususc/fracdet`, 2017.

[46] Constantino A. García. vaele: an SDE-based Structured Variational Autoencoder. `https://github.com/constantino-garcia/vaele`, 2018.

[47] Constantino A. García, Abraham Otero, Paulo Félix, Jesús Presedo, and David G. Márquez. Nonparametric estimation of stochastic differential equations with sparse Gaussian processes. *Physical Review E*, 96:022104, Aug 2017.

[48] Constantino A. García, Abraham Otero, Paulo Félix, Jesús Presedo, and David G Márquez. Simultaneous estimation of deterministic and fractal stochastic components in non-stationary time series. *Physica D: Nonlinear Phenomena*, 2018.

[49] Constantino A. García, Abraham Otero, Xosé Vila, and David G. Márquez. A new algorithm for wavelet-based heart rate variability analysis. *Biomedical Signal Processing and Control*, 8(6):542–550, 2013.

[50] Constantino A. García, Abraham Otero, Xosé A Vila, María José Lado, Leandro Rodríguez-Liñares, Jesús María Presedo, and Arturo José Penín. *Heart Rate Variability Analysis with the R Package RHRV*. Springer, 2017.

[51] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian data analysis*. Taylor & Francis, 2014.

[52] Fatemeh Ghasemi, Muhammad Sahimi, J. Peinke, R. Friedrich, G. Reza Jafari, and M. Reza Rahimi Tabar. Markov analysis and Kramers-Moyal expansion of nonstationary stochastic processes with application to the fluctuations in the oil price. *Physical Review E*, 75(6):060102, 2007.

[53] Daniel T. Gillespie. *Markov processes: an introduction for physical scientists*. Elsevier, 1991.

[54] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58:35–55, 2007.

[55] Ary L. Goldberger, David R. Rigney, and Bruce J. West. Chaos and fractals in human physiology. *Scientific American*, 262(2):42–49, 1990.

[56] Herbert Goldstein. *Classical mechanics*. Pearson Education, 2011.

[57] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press, 2016.

[58] David J. Griffiths. *Introduction to quantum mechanics*. Cambridge University Press, 2016.

[59] Shlomo Havlin, Sergey V. Buldyrev, Ary L. Goldberger, Rosario N. Mantegna, S.M. Ossadnik, Chunk-Kang Peng, Michael Simons, and H. Eugene Stanley. Fractals in biology and medicine. *Chaos, Solitons & Fractals*, 6:171–201, 1995.

[60] Rainer Hegger and Gerhard Stock. Multidimensional Langevin modeling of biomolecular dynamics. *The Journal of Chemical Physics*, 130(3):034106, 2009.

[61] James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, page 282, 2013.

[62] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in Neural Information Processing Systems*, pages 3–10, 1994.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[64] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[65] Werner Horsthemke. Noise induced transitions. In *Non-Equilibrium Dynamics in Chemical Systems*, pages 150–160. Springer, 1984.

[66] Stefano M. Iacus. *Simulation and inference for stochastic differential equations: with R examples*. Springer, 2009.

[67] Leon Isserlis. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, 12(1/2):134–139, 1918.

[68] E.T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003.

[69] Matthew Johnson, David K. Duvenaud, Alex Wiltschko, Ryan P. Adams, and Sandeep R. Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pages 2946–2954, 2016.

[70] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[71] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge University Press, 2004.

[72] Brandon C. Kelly, Jill Bechtold, and Aneta Siemiginowska. Are the variations in quasar optical flux driven by thermal fluctuations? *The Astrophysical Journal*, 698(1):895, 2009.

[73] William J. Kennedy and James E. Gentle. *Statistical computing*. Marcel Dekker Inc, New York, 1980.

[74] Marvin S. Keshner. 1/f noise. In *Proceedings of the IEEE*, volume 70, pages 212–218. IEEE, 1982.

[75] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[76] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[77] Peter E. Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. Applications of Mathematics. Springer, 1999.

[78] Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[79] S. Krumscheid, M. Pradas, G.A. Pavliotis, and S. Kalliadasis. Data-driven coarse graining in action: Modeling and prediction of complex systems. *Physical Review E*, 92(4):042139, 2015.

[80] Kestutis Kubilius. The existence and uniqueness of the solution of an integral equation driven by a p-semimartingale of special type. *Stochastic Processes and their Applications*, 98(2):289–315, 2002.

[81] Tom Kuusela. Stochastic heart-rate model can reveal pathologic cardiac dynamics. *Physical Review E*, 69(3):031916, 2004.

[82] David Lamouroux and Klaus Lehnertz. Kernel-based regression of drift and diffusion coefficients of stochastic processes. *Physics Letters A*, 373(39):3507–3512, 2009.

[83] P. Lançon, G. Batrouni, L. Lobry, and N. Ostrowsky. Drift without flux: Brownian walker with a space-dependent diffusion coefficient. *EPL (Europhysics Letters)*, 54(1):28, 2001.

[84] Paul Langevin. Sur la théorie du mouvement Brownien. *Comptes-Rendus de l'Académie des Sciences, Paris*, 146(530-533):530, 1908.

[85] Kristian Stegenborg Larsen and Michael Sørensen. Diffusion models for exchange rates in a target zone. *Mathematical Finance*, 17(2):285–306, 2007.

[86] Pedro G. Lind, Iván Herráez, Matthias Wächter, and Joachim Peinke. Fatigue load estimation through a simple stochastic model. *Energies*, 7(12):8279–8293, 2014.

[87] Pedro G. Lind, Alejandro Mora, Jason A.C. Gallas, and Maria Haase. Reducing stochasticity in the North Atlantic Oscillation index with coupled Langevin equations. *Physical Review E*, 72(5):056706, 2005.

[88] Hui Liu, Hong-qi Tian, Di-fu Pan, and Yan-fei Li. Forecasting models for wind speed using wavelet, wavelet packet, time series and artificial neural networks. *Applied Energy*, 107:191–208, 2013.

[89] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[90] Torbjörn Lundahl, William J. Ohley, Steven M. Kay, and Robert Siffert. Fractional Brownian motion: A maximum likelihood estimator and its application to image texture. *IEEE Transactions on Medical Imaging*, 5(3):152–161, 1986.

[91] Reinhard Mahnke, Jevgenijs Kaupužs, and I. Lubashevsky. Probabilistic description of traffic flow. *Physics Reports*, 408(1):1–130, 2005.

[92] James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

[93] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[94] Abdessamad Mekkaoui. Derivation of stochastic differential equations for scrape-off layer plasma fluctuations from experimentally measured statistics. *Physics of Plasmas*, 20(1):010701, 2013.

[95] Patrick Milan, Matthias Wächter, and Joachim Peinke. Turbulent character of wind energy. *Physical Review Letters*, 110(13):138701, 2013.

[96] Jan Kloppenborg Møller and Henrik Madsen. *From state dependent diffusion to constant diffusion in stochastic differential equations by the Lamperti transform*. DTU Informatics, 2010.

[97] Kevin P. Murphy. Conjugate Bayesian analysis of the Gaussian distribution. Technical report, 2007.

[98] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[99] Guy Nason. *wavethresh: Wavelets statistics and transforms.*, 2013. R package version 4.6.6.

[100] Peter Nemenyi. Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. International Biometric Society, 1962.

[101] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 269, pages 543–547, 1983.

[102] Ertugrul M. Ozbudak, Mukund Thattai, Iren Kurtser, Alan D. Grossman, and Alexander Van Oudenaarden. Regulation of noise in the expression of a single gene. *Nature Genetics*, 31(1):69–73, 2002.

[103] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[104] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch, 2017.

[105] Donald B. Percival and Andrew T. Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge University Press, 2006.

[106] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. *Technical University of Denmark*, 7:15, 2008.

[107] Arkady S. Pikovsky and Jürgen Kurths. Coherence resonance in a noise-driven excitable system. *Physical Review Letters*, 78(5):775, 1997.

[108] Jens Prusseit and Klaus Lehnertz. Measuring interdependences in dissipative dynamical systems with estimated Fokker-Planck coefficients. *Physical Review E*, 77(4):041914, 2008.

[109] Joaquín Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[110] Yalda Rajabzadeh, Amir Hossein Rezaie, and Hamidreza Amindavar. A robust nonparametric framework for reconstruction of stochastic differential equation models. *Physica A: Statistical Mechanics and its Applications*, 450:294–304, 2016.

[111] Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

[112] Frederick Reif. *Fundamentals of statistical and thermal physics*. Waveland Press, 2009.

[113] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[114] Hannes Risken. Fokker-planck equation. In *The Fokker-Planck Equation*, pages 63–95. Springer, 1996.

[115] Paulo Rocha, Frank Raischel, João P. Boto, and Pedro G. Lind. Uncovering the evolution of nonstationary stochastic variables: The example of asset volume-price fluctuations. *Physical Review E*, 93(5):052122, 2016.

[116] Sten Rüdiger, Jianwei Shuai, Wilhelm Huisinga, Chamakuri Nagaiah, Gerald Warnecke, Ian Parker, and Martin Falcke. Hybrid stochastic and deterministic simulations of calcium blips. *Biophysical Journal*, 93(6):1847–1857, 2007.

[117] Andreas Ruttor, Philipp Batz, and Manfred Opper. Approximate Gaussian process inference for the drift function in stochastic differential equations. In *Advances in Neural Information Processing Systems*, pages 2040–2048, 2013.

[118] Tim Sauer, James A. Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65(3-4):579–616, 1991.

[119] Teresa Scholz, Frank Raischel, Vitor V Lopes, Bernd Lehle, Matthias Wächter, Joachim Peinke, and Pedro G Lind. Parameter-free resolution of the superposition of stochastic signals. *Physics Letters A*, 381(4):194–206, 2017.

[120] Michael Schulz. On the 1470-year pacing of Dansgaard-Oeschger warm events. *Paleoceanography*, 17(2), 2002.

[121] Bernard W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[122] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[123] Yannis Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, 9(1):21–29, 2001.

[124] Floris Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics*, 898(1):366–381, 1981.

[125] TensorFlow Development Team. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from `tensorflow.org`.

[126] Ahmed H. Tewfik and M. Kim. Correlation structure of the discrete wavelet coefficients of fractional Brownian motion. *IEEE Transactions on Information Theory*, 38(2):904–909, 1992.

[127] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, May 2016.

[128] Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, volume 12, pages 567–574, 2009.

[129] Michalis K. Titsias and Miguel Lázaro-Gredilla. Local expectation gradients for black box variational inference. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2638–2646. Curran Associates, Inc., 2015.

[130] U.S. Energy Information Administration. Crude oil prices from `https://www.eia.gov/dnav/pet/pet_pri_spt_s1_d.htm`, 2017. Last accessed: 2019-01-16.

[131] Brani Vidakovic. Nonlinear wavelet shrinkage with Bayes rules and Bayes factors. *Journal of the American Statistical Association*, 93(441):173–179, 1998.

[132] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[133] Marian Von Smoluchowski. Zur kinetischen theorie der brownschen molekularbewegung und der suspensionen. *Annalen der Physik*, 326(14):756–780, 1906.

[134] Chong Wang and David M. Blei. Variational inference in nonconjugate models. *Journal of Machine Learning Research*, 14(Apr):1005–1031, 2013.

[135] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, pages 1067–1075, 2013.

[136] Yoshiharu Yamamoto and Richard L. Hughson. Extracting fractal components from time series. *Physica D: Nonlinear Phenomena*, 68(2):250–264, 1993.

[137] Han Lun Yap, Armin Eftekhari, Michael B. Wakin, and Christopher J. Rozell. A first analysis of the stability of Takens' embedding. In *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 404–408. IEEE, 2014.

[138] Matthew D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Acronyms

AIC Akaike Information Criterion.

CGSA Coarse Graining Spectral Analysis.

CRAN Comprehensive R Archive Network.

dfBm discrete fractional Brownian motion.

dfGn discrete fractional Gaussian noise.

DO Dansgaard-Oeschger.

ELU Exponential Linear Unit.

EM Expectation Maximization.

fBm fractional Brownian motion.

fGn fractional Gaussian noise.

GEM Generalized Expectation Maximization.

GP Gaussian Process.

GPs Gaussian Processes.

GRU Gated Recurrent Unit.

GSL GNU Scientific Library.

HMM Hidden Markov Model.

iid independent and identically distributed.

KBR Kernel Based Regression.

LSTM Long Short-Term Memory.

MAD Median Absolute Deviation.

MLP Multilayer Perceptron.

NLS Nonlinear Least Squares.

PSD Power Spectral Density.

RHS Right-hand side.

RNN Recurrent Neural Network.

RON Research Octane Number.

SDE Stochastic Differential Equation.

SGP Sparse Gaussian Process.

SLDS Switching Linear Dynamical System.

SNR Signal-to-Noise Ratio.

SVAE Structured Variational Autoencoder.

SVI Stochastic Variational Inference.

TBPTT Truncated Backpropagation Through Time.

VAE Variational Autoencoder.

VI variational inference.

List of Figures

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Tables