INSA Toulouse

Department of Applied Mathematics

Data Assimilation

Inverse Problems, Assimilation, Control, Learning

by Jérôme Monnier

This document is an automatically generated shortened version of the complete course manuscript. The complete manuscript is more detailed, you can obtain it on the course webpage.

Goals: solve inverse problems, assimilate datasets into physically-based models, infer / identify model parameters, calibrate models, learn model terms from rich datasets, compute gradients of large dimensional model outputs

Keywords: inverse problems, best estimates, optimisation, gradient-based algorithms, model-based feedback control, PDE-based models, adjoint method, Earth sciences.

Required knowledge: basic approximation methods, numerical optimisation, classical PDE models (skills in weak forms and basics of functional analysis are a plus), numerical schemes (Finite Differences, Finite Element is a plus), Python programming.

Contents

1

[1]The sections indicated with a * are "to go further sections". These sections can be skipped as a first reading, or if you are not interested in deeper mathematical basis, mathematical proofs.

Part I

Inverse Problems: Basics Principles and Tools, Examples

Chapter 1

Inverse problems

The outline of this chapter is as follows.

Contents

1.1 Direct - inverse?

In the common sense, the term "model" denotes a *direct* model (also called *"forward"* model).

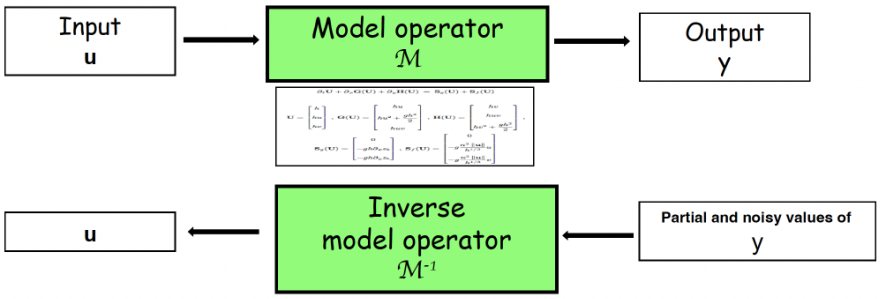


Figure 1.1: Representation of a direct model with its input variable ("parameter") u (a-priori vectorial) and its output variable y; and its inverse counterpart.

1.2 Examples

Example 1) The historical Lagrange interpolation problem.
Find a polynomial $p(x)$ of degree n, of coefficients $(u_0, \cdots, u_n)$ that fit given values $(y_1, , ..., y_n)$ at some points $(x_1, ..., x_n)$.
This Lagrange interpolation problem is actually the inverse problem of the following direct problem: calculate the given polynomial $p(x)$ at the points $(x_1, ..., x_n)$.

This example is somehow a problem of *parameters identification*, given the "model" $p(u; x)$.

Example 2) A PDE-based identification problem.
Let us consider a diffusion phenomena in a material. The non homogeneous diffusivity of the material (e.g. a conductivity of a biological tissue) is denoted by $u(x)$.
Given $u(x)$ in the domain Ω, find the scalar quantity $y(x,t)$ (e.g. an electrical field or wave intensity) satisfying:

$$\begin{cases} \partial_t y(x,t) - div(u(x)\nabla y(x,t)) = 0 & \text{in } \Omega\times]0,T[\\ y(x,0) \;=\; y_0(x) & \text{in } \Omega \\ y(x,t) = y_d(x,t) & \text{in } \partial\Omega\times]0,T[\end{cases} \tag{1.1}$$

The direct problem consists to solve this classical Boundary Value Problem (BVP).

The inverse problem is as follows.
Given some *boundary measurements* of the field $y(x,t)$ and the flux $[u(x)\partial_n y(x,t)]$ on $\partial\Omega$, determine the unknown / uncertain diffusivity coefficient $u(x)$ in the domain Ω.

The Electrical Impedance Tomography (EIT) problem A particular case is the Electrical Impedance Tomography (EIT) problem.
" Electrical Impedance Tomography (EIT) is a noninvasive type of medical imaging in which the electrical conductivity, permittivity, and impedance of a part of the body is inferred from surface electrode measurements and used to form a tomographic image of that part" (from Wikipedia page). See eg. Fig. 1.2.
In the context of Electrical Impedance Tomography (EIT), the inverse problem aims at recovering conductivity (and permittivity) inside a body from surface measurements of electrical currents and potentials.

It is potentially an ill-posed problem, depending on the assumptions, see e.g. [L. Borcea, Inv. Problems (2002)].

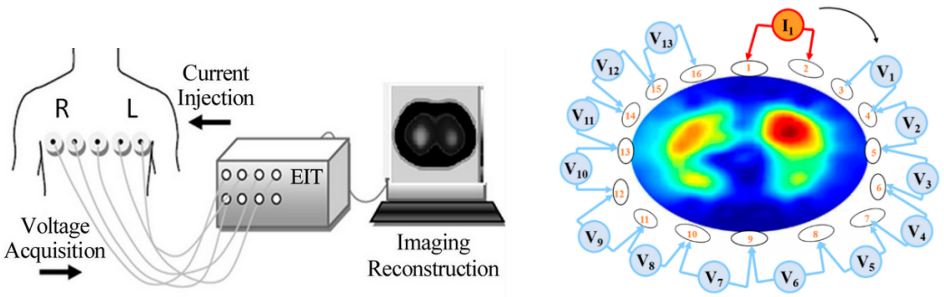


Figure 1.2: Electrical Impedance Tomography (EIT) for cardio-pulmonary monitoring: voltage measurements around the thorax using an EIT system with 16 electrodes. (R) Image extracted from C. Putensen et al., J. Clinical Medecine (2019).

1.3 General concepts

Basic formalism

In real-world problems, the measurements, denoted by z^{obs} ($z^{obs} \equiv y^{obs}$ if the observations are directly the model outputs), are *almost always incomplete, sparse or inaccurate.*

$$y^{obs} = \mathcal{M}(u) + \varepsilon \tag{1.2}$$

with ε a global error term incorporating both the observation errors and the structural model error: $\varepsilon = \varepsilon_{obs} + \varepsilon_{mod}$.
ε is defined as a stochastic field following an a-priori distribution, actually Gaussian when no other information is available.

We have assumed here that the observations are directly the model outputs: $z^{obs} = y^{obs}$.

1.3.1 Even the inversion of linear operators may not be trivial...

In the linear case, the direct model is represented by a matrix M.
Naïvely solving the inverse problem as $u = (M^{-1}y^{obs})$ may not work for few reasons. Two trivial ones are:
- observations y^{obs} can be not numerous enough therefore providing an undetermined problem,
- the error term ε can be unknown.

Moreover a third reason is due to the fact that the inverse operator M^{-1} can be ill-conditioned (small variations of y implies large variations of u).
Note that M well-conditioned direct model implies that M^{-1} ill-conditioned (and conversely).
Indeed, the 2-norm condition number reads: $\kappa_2(M) = \frac{\max_i |\lambda(M)|}{\min_i |\lambda_i(M)|}$, λ_i the eigenvalues.
Moreover, $\lambda_i(M^{-1}) = (\lambda_i(M))^{-1}$. Therefore the statement.

"Mathematically invertible" does not mean "numerically easily invertible"...

In all the sequel, we define the control-to-state operator $\mathcal{M}$ (also called here the "model operator") as follows:

$$\boxed{\mathcal{M} : u \in \mathcal{U} \mapsto y \in \mathcal{Y}} \tag{1.3}$$

1.3.2 Well-posedness, ill-posedness

In the Hadamard sense[1], a model is well-posed if the following conditions hold:
i) it admits an unique solution,
ii) the solution depends continuously on the data or input parameters.

The first condition i) (existence and uniqueness) is related to the functional space the solution is sough in.
The second condition ii) is a stability condition. This property is crucial too. Indeed, if this condition is not satisfied then any measurement or numerical error generates large errors in the model response i.e. a highly perturbed solution.

In all the sequel, it will be assumed that the direct model is well-posed. This is a necessary condition to go further !
Assumption ii) may be re-read as follows: the control-to-state operator $\mathcal{M}$ is continuous.

Even if the direct problem is well-posed, the inverse problem is often severely ill-posed !

Ill-posed inverse problems are somehow the opposite of well-posed direct problems.
Direct problems are usually the way that can be solved easily (compared to the inverse problem). Actually, direct and inverse models are back-and-forth transmission of information between u and y. Roughly, if the direct model operator maps causes to effects, the inverse operator maps the effects to the causes.

Exercise. Propose a PDE-based model which is well-posed in the Hadamard sense.
An answer: linear elliptic BVPs, coercitive in a Hilbert space V, may be well-posed in vertu of the Lax-Milgram theorem. □

1.3.3 Direct - inverse models: reverse frequencies

In real-world problems, the direct models generally represent the lowest frequencies of the modeled phenomena: $\min_i |\lambda(M)|$ is relatively large compared to noise frequencies, see e.g. Fig. 1.3.
The most energetic modes of the Fourier representation of a signal, here $y(u)$ the direct model output, are the lowest frequencies. Noises are high frequencies (therefore not energetic).
The highest frequencies of $y(u)$ are the lowest frequencies of $u(y)$! As a consequence, separating noise from the inverse model image is a difficult task.

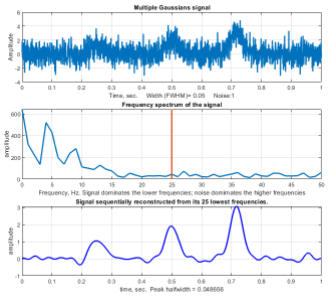


Figure 1.3: Direct models enerally represent the lowest frequencies of the modeled phenomena (here a superimposition of Gaussians).

In real-life modeling, inverse problems are as common as direct problems. The inverse problem classes and techniques to solve them are extremely wide.

Data Assimilation (DA) methods aim at fusing data into mathematical models. DA may be viewed as a class of method aiming at solving some particular inverse problems.

Chapter 2

Basic tools

The outline of this chapter is as follows.

Contents

2.1 Least-square solutions of regression problems, SVD

2.1.1 Linear least-square problems

Let assume that we have m measurements $(z_i)_{1\leq i\leq m}$ we seek to describe by a "model".
To do so, we choose to consider the following linear model with n unknown parameters $(u_i)_{1\leq i\leq n}$:

$$\left\{\begin{array}{rcl} a_{11}u_1 + ... + a_{1n}u_n &=& z_1 \\ ... &=& ... \\ a_{m1}u_1 + ... + a_{mn}u_n &=& z_m \end{array}\right.$$

4 We denote: $A = (a_{ij})_{1\leq i\leq m, 1\leq j\leq n}$ the chosen linear transformation (the linear model is given).
A is a $m \times n$-matrix, $z \in \mathbb{R}^m$ the observation vector and $u \in \mathbb{R}^n$ the unknown input parameter vector.
This is a *identification parameter problem*. This problem reads as:

$$\left\{\begin{array}{l} \text{Given } A \in \mathbb{R}^{m\times n} \text{ and } z \in \mathbb{R}^m, \\ \text{find } u \in \mathbb{R}^n \text{ such that: } Au = z \end{array}\right. \tag{2.1}$$

In practice there is no reason to have $n = m$!...

In the case $n > m$ i.e. in the unlikely case there is more input parameters than observations, the system is under-determined. Generally, it exists solutions but they are non-unique.

In the case $n < m$ i.e. in the usual case there is less input parameters than observations, the system is over-determined. A-priori it does not exist any solution fitting all data.
Indeed, with m input parameters, it can exist a unique solution making fit the observations; but what about the extra $(n - m)$ "constraint equations" ?

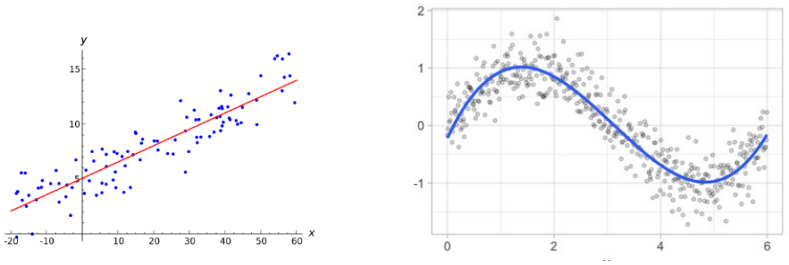


Figure 2.1: Linear least-square problem (with dense observations-measurements !). (L) The most simple case: linear regression; two parameters to identify. (R) An other simple case (polynomial, degree 3).

Least-square solution(s)

$$\begin{cases} \text{Find } u^* \in \mathbb{R}^n \text{ such that:} \\ j(u^*) = \min_{\mathbb{R}^n} j(u) \text{ with } j(u) = \frac{1}{2}\|Au - z\|^2_{2,m} \end{cases} \tag{2.2}$$

It is an unconstrained optimization problem in a convex set.

Then, the solution u has to satisfy the *necessary optimal condition*:

$$\boxed{A^T Au = A^T z} \tag{2.3}$$

This is the *normal equations*.

Examples

The reader can find numerous well documented examples with corresponding Python codes available on the web, e.g. on the https://towardsdatascience.com web site [1].

[1] https://towardsdatascience.com/linear-regression-using-least-squares-a4c3456e8570

2.1.2 Non linear least-square problems

Let us consider the same problem as previously: set up a model describing "at best" m available data $z_i,\ 1 \leq i \leq m$ but based on a *non-linear* model.

$$\boxed{j(u^*) = \min_{\mathbb{R}^n} j(u) \text{ with } j(u) = \frac{1}{2}\|M(u)\|^2_{2,m}} \tag{2.4}$$

with M defined from $\mathbb{R}^n$ into $\mathbb{R}^m$, M *non linear* in u.

Again, it is an unconstrained optimization problem in a convex set.

In the previous linear case, M was equal to: $M(u) = Au - z$.

Note that neither the existence of a solution, nor its uniqueness is guaranteed without assumptions on the non-linear operator $M(\cdot)$ and/or the space solutions.

Example: The location problem by Global Navigation System Satellites (GPS, Galileo etc). The location problem by Global Navigation System Satellites (GNSS) consists to estimate x, $x \in \mathbb{R}^3$ (the location value).
The measurements are distances d_i^{obs} from given locations a_i with $i = 1, \cdots, m$ (the m satellites):

$$d_i^{obs} = \|a_i - x^{exact}\|_{2,\mathbb{R}^3} + \varepsilon_i \quad i = 1, .., m$$

where ε denotes a noise (the inacurracy of the measurements).

The least-square problem to be solved is:

$$\min_{x \in \mathbb{R}^3} j(x) \tag{2.5}$$

with: $j : \mathbb{R}^3 \to \mathbb{R}$,

$$j(u) = \|\|a_i - u\|_{2,\mathbb{R}^3} - d_i^{obs}\|_{2,\mathbb{R}^m}^2 = \sum_{i=1}^{m} \left(\|a_i - u\|_{2,\mathbb{R}^3} - d_i^{obs} \right)^2$$

The functional $j(\cdot)$ is non convex, see Fig. 2.2.

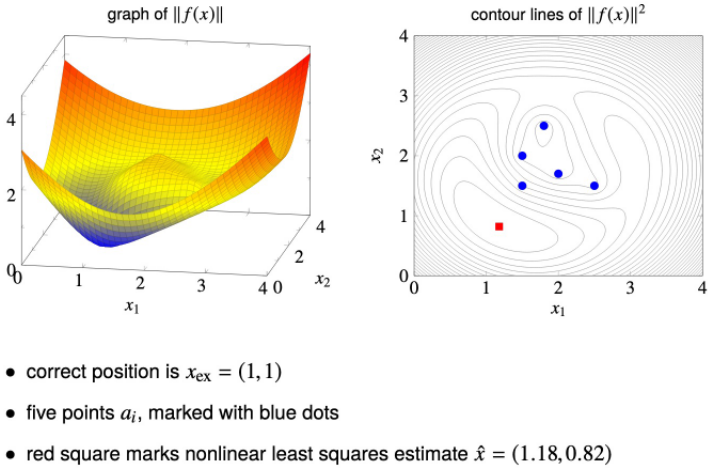


Figure 2.2: Non-linear least square problems: the location problem.
(Image extracted from Vandenberghe's UCLA course).
(L) The graph of $j(u)$, non convex functional in u. (R) Iso-values of $j(u)$, data, exact solution and the least-square one.

Optimality condition: derivatives calculations

The necessary first order optimality condition for the general non-linear square problem (2.4) reads:

$$\boxed{\nabla j(u) = 0} \tag{2.6}$$

We have: $j(u) = \frac{1}{2}\|M(u)\|^2_{2,m}$. Let us denote the Jacobian of M as follows:

$$DM(u) = \left(\frac{\partial M_i}{\partial u_j}(u)\right)_{1\leq i\leq m, 1\leq j\leq n}$$

The Hessian for each model component $M_i(u), 1 \leq i \leq m$ is denoted as follows:
$D^2M_i(u) = \left(\frac{\partial^2 M_i}{\partial u_l \partial u_j}(u)\right)_{1\leq l\leq m, 1\leq j\leq n}$.
Then, the gradient and the Hessian of $j(u)$ read:

$$\nabla j(u) = DM^T(u)M(u)$$

$$H_j(u) \equiv D^2 j(u) = DM^T(u)DM(u) + \sum_{i=1}^{m} M_i(u)D^2M_i(u)$$

Let us remark that in the *linear case*, the gradient read: $\nabla j(u) = A^T M(u) = A^T(Au - z)$ and the Hessian reads: $H_j(u) = A^T A$ since the term D^2M_i in the Hessian expression vanishes.

Exercise 2.1. *Verify the expression of the gradient and the Hessian above.*

Gauss-Newton method, Levenberg-Marquardt method

The *Newton algorithm* applied to the optimality condition $\nabla j(u) = 0$ consists to solve at each iteration:

$$\boxed{H_j(u^n) \cdot \delta u = -\nabla j(u^n) \ ; \quad u^{n+1} = u^n + \delta u}$$

Newton's algorithm requires the computation and the inversion of the Hessian of j.

Exercise 2.2. *Verify that the Newton algorithm applied to the equation* $(\nabla j(u) = 0)$ *reads as above.*

The principle of the *Gauss-Newton method* is to consider in the Newton method by approximating the Hessian by omitting the second order term:

$$H_j(u) \approx DM^T(u)DM(u) \equiv \tilde{H}_j(u) \tag{2.7}$$

This gives at each iteration:

$$\boxed{\tilde{H}_j(u) \cdot \delta u = -\nabla j(u^n) \ ; \quad u^{n+1} = u^n + \delta u} \tag{2.8}$$

with (recall) $\nabla j(u^n) = DM^T(u^n)M(u^n)$.

Note that the linear system to be inverted is symmetric positive, and definite if $DM(u^n)$ is full rank.

Finally, a good alternative method to solve non-linear least square problems is the *Levenberg-Marquardt algorithm*, see e.g. [3].
This algorithm is somehow a damped version of the Gauss-Newton method. It can be seen as the combination of a descent algorithm, next the Gauss-Newton algorithm as above.

2.2 Ill-posed inverse problems: regularization

2.2.1 General cases: Tikhonov's regularization

This approach consists to compute $u^* \in \mathbb{R}^n$ such that $j_\alpha(u^*) = \min_{\mathbb{R}^n} j_\alpha(u)$ with the following enriched cost function:

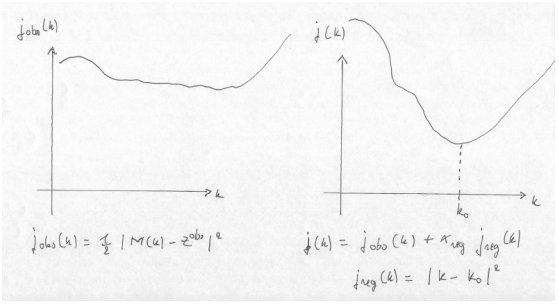
$$\boxed{j_\alpha(u) = \frac{1}{2}\|Au - z\|^2_{2,m} + \alpha_{reg}\frac{1}{2}\|Cu\|^2_{2,n}} \tag{2.9}$$

where α_{reg} is a positive scalar (the weight) making the balance between the two terms.
C is a linear operator (a matrix).
The added regularization term is convex, differentiable.

The simplest choice for C is $C = Id$. In the present linear case (A is linear operator), this provides the least-square solution with the minimal 2-norm.

The solution of the corresponding normal equations reads:

$$\boxed{u^*_{alpha} = (A^T A + \alpha^2_{reg} C^T C)^{-1} \cdot A^T z}$$



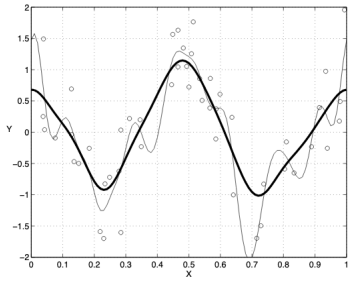


Figure 2.3: Tikhonov regularization. (L) A typical "poorly convex" functional $j_{data}(\cdot)$: ill-conditioned minimisation problem. (M) Regularized functional to be minimized: $j_{reg}(.) = (j_{data}(.) + \alpha_{reg} j_{reg}(.))$ with $j_{reg}(u) = \|u - u_0\|$, u_0 a *prior* value. (R). Data fitting (in the LS sense) without and with regularization (Image extracted from a MIT course).

A few comments

In statistics, the *Tikhonov regularization* is called the *ridge regression* method.
In Machine Learning (ML), it corresponds to the so-called *weight decay* method.

In real-world modeling problems, the regularization operator C is often defined as a differential operator (e.g. gradient operator) or as a Fourier operator e.g. aiming at filtering high frequencies.

Indeed, it turns out that numerous real-world phenomena (therefore the models) have the effect of low-pass filtering. As a consequence, the inverse operator acts as a high-pass filter....

Recall that the eigenvalues (or singular values) are the largest in the reverse mapping when they are the smallest in the forward mapping.

Remark 2.3. *Other regularization terms can be considered in particular in L^q-norm with $q = 1$. In this case, the regularization term is: $\alpha_{reg}\|Cu\|_1$.*
The case $q = 1$ is highly interesting because of the compress sensing property of 1-norm, see e.g.[17].

Chapter 3

Real-world examples of inverse problems

Please consult the supplementary material

Part II

Data Assimilation (DA): Sketch of Methods

Chapter 4

DA in a nutshell

The outline of this chapter is as follows.

Contents

4.1 Data Assimilation (DA): what is it and why is it important?

Data Assimilation (DA) is the science of optimally combining different knowledge sources that we acquire about a phenomenon modeled by various mathematical tools.

- A mathematical model representing the physical phenomena (in the broad sense).
- Observations, also referred to as measurements or data.
- Statistics on the observations and/or prior probability density functions (pdf) on the modeled phenomena.

The goal of DA is to estimate the state of a system as it evolves in time by combining these different knowledge sources in an optimal way. In real-world problems, particularly in environmental sciences (meteorology, oceanography, hydrology, seismology, etc.), data are heterogeneous, multi-scale, and sparse both in space and time. As a consequence, they only partially represent the modeled phenomena.

DA: what for?

Setting up and performing a Data Assimilation (DA) process can be motivated by different goals, including:

- Correcting a model output (given datasets).
- Calibrating the model to improve its prediction accuracy.
- Identifying a physical parameter of the model.

The "traditional" DA methods mentioned above differ from purely ML approaches (e.g., Artificial Neural Networks) since they rely on a model (typically a PDE).
However, Physically-Informed Neural Networks (PINNs) aim at combining Neural Networks (a purely Machine Learning technique) with physical models.

4.2 The different types of DA methods

Up to the years 2020's, DA mainly relied on two different classes of methods (so-called here tradiational methods): sequential and variational ones. The choice may be driven either of the unknown parameters of the inverse problem is of large dimension or not.

1) *Sequential approaches (filters)* dedicated to the estimation, given series of observations.
The fundamental filter is the Kalman Filter (KF) which is optimal in the linear Gaussian case.

2) The *variational approach* (VDA for Variational Data Assimilation) relies on the *optimal control* of the model, with respect to the unknown/uncertain parameter.

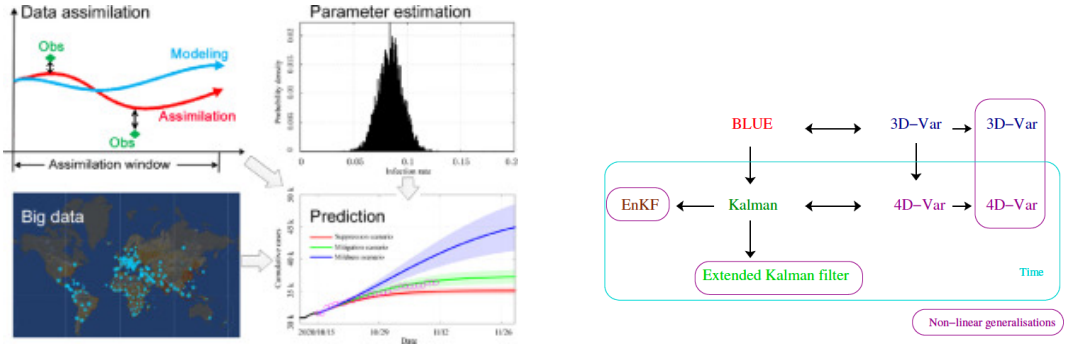


Figure 4.1: (Left) Goals of DA. 1) DA to identify an uncertain input parameter (it can be for a steady-state model). The resulting calibrated model is more accurate. It can be used as a physically-consistent interpolator between data. 2) DA to build up a predictive model (here for a time-dependent model). The model is first calibrated from past observations. Second, it is performed for prediction. (Right) The different DA methods and their connections (see later). Image source: [7]

To model real-world problems, Data Assimilation (DA) must incorporate both deterministic elements (based on known physics) and stochastic elements (to account for uncertainties).

In idealistic Linear Quadratic (LQ) cases (where the model is linear and the cost function to minimize is quadratic), both the VDA approach and the reference sequential method, namely the Kalman Filter (KF), are equivalent (assuming the appropriate norms are considered).

Furthermore, in the Linear Quadratic Gaussian (LQG) case i.e. with Gaussian fields, VDA and the KF can be elegantly interpreted within a Bayesian framework.

Chapter 5

DA by sequential filters

The outline of this chapter is as follows.

Contents

5.1 The Best Linear Unbiased Estimator (BLUE)

5.1.1 A basic 1D example

Let us consider two measurements of a scalar quantity u: $z_1 = 1$ and $z_2 = 2$.
Naturally, one seeks u minimizing the following cost function: $j(u) = (u - z_1)^2 + (u - z_2)^2$. The solution is: $u^* = \frac{3}{2}$.

Now, let us assume that the second data represents the quantity $2u$ and not u (difference of instrument). Then, the two measurements are: $z_1 = 1$ and $z_2 = 4$. We here seek u minimizing the cost function: $g(u) = (u - z_1)^2 + (2u - z_2)^2$.
In this case, the solution is $u^* = \frac{9}{5}$.

This very simple example illustrates that the least-square solution depends on the norm considered in the cost function (here the "natural" 2-norm): *the standard least-square solution depends on the measurement norm* (and on the data accuracy of course too).

Informal definition of the BLUE

Considering the aleatory variable $\hat{u}$ defined from the data z_{obs}, the BLUE u^* is defined from the three following properties:

a) u^* is a linear function of z_{obs}.

b) u^* is unbiased (means are unchanged): $E(u^*) = u$.

c) u^* is optimal in the sense it has the smallest variance among all unbiased linear estimations.

Calculation of the BLUE for the 1D basic example

$$z_i = u + \varepsilon_i,\ i = 1, 2$$

The errors of measurements ε_i are supposed to be:
- unbiased: $E(\varepsilon_i) = 0$. *(Sensors are unbiased).*
- with a known variance: $Var(\varepsilon_i) = \sigma_i,\ i = 1, 2$. *(The sensor accuracies are known).*
- uncorrelated : $E(\varepsilon_1\varepsilon_2) = 0$. *(Measurements are independent hence the covariance vanishes; in addition, means vanish therefore this relation).*

By construction (Property a)), the BLUE satisfies: $u^* = a_1 z_1 + a_2 z_2$ (linear model). The coefficients a_i have to be determined. We have: $u^* = (a_1 + a_2)u + a_1\varepsilon_1 + a_2\varepsilon_2$.
By linearity of $E(\cdot)$,

$$E(u^*) = (a_1 + a_2)u + a_1 E(\varepsilon_1) + a_2 E(\varepsilon_2) = (a_1 + a_2)u$$

Property b) of the BLUE (unbiased estimator) implies that $(a_1 + a_2) = 1$ (equivalently $a_2 = (1 - a_1)$).

Recall that by definition, $Var(u^*) = E[(u^* - E(u^*))^2] = E[(u^* - u)^2]$. Therefore:

$$\begin{aligned} Var(u^*) &= E[(a_1\varepsilon_1 + a_2\varepsilon_2)^2] \\ &= a_1^2 E(\varepsilon_1^2) + 2a_1 a_2 E(\varepsilon_1\varepsilon_2) + a_2^2 E(\varepsilon_2^2) \\ &= a_1^2\sigma_1^2 + (1 - a_1)^2\sigma_2^2 \end{aligned}$$

By definition, u^* minimizes the variance (Property c)). The latter is minimal if its derivative with respect to a_1 vanishes. Therefore: $a_1 = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$.

Therefore, the BLUE reads:

$$u^* = \frac{1}{(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2})}(\frac{1}{\sigma_1^2} z_1 + \frac{1}{\sigma_2^2} z_2) \tag{5.1}$$

Note that: $Var(u^*) = \frac{\sigma_1^2\sigma_2^2}{(\sigma_1^2 + \sigma_2^2)}$. Therefore: $\frac{1}{Var(u^*)} = \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right)$.

In statistics, the inverse of a variance is called precision.

Equivalence with an optimization problem

It is easy to verify that the BLUE u^* defined by (5.1) is the unique minimum of the following quadratic cost function:

$$\boxed{j(u) = \frac{1}{2}\frac{1}{\sigma_1^2}(u - z_1)^2 + \frac{1}{2}\frac{1}{\sigma_2^2}(u - z_2)^2} \tag{5.2}$$

Indeed, we have: $j''(u) = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} = \frac{1}{Var(u^*)}$.

The Hessian of the cost function $j(u)$ (the "convexity rate" of j) equals the estimation accuracy, see Fig. 5.1.

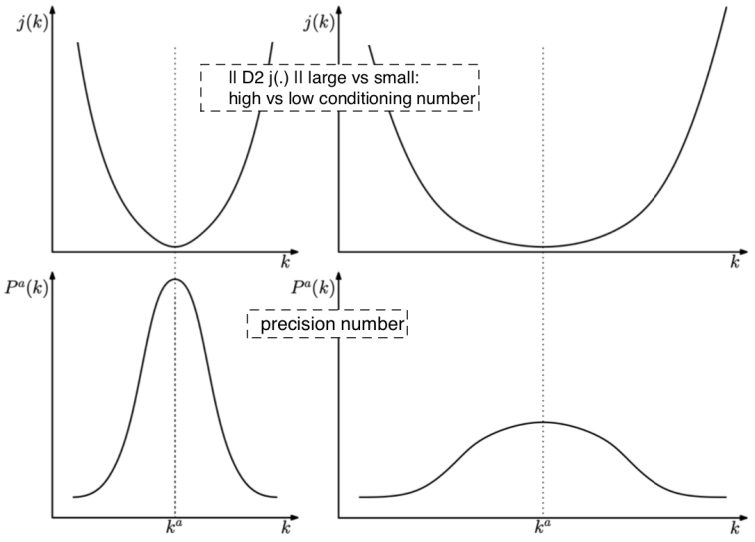


Figure 5.1: 1D case. (Up) the cost fucntiuon. (Down) The second derivative (= here, the precision).
The Hessian = the second derivative $j"(u)$ in the present 1D case.
$j"(x)$ measures the "convexity rate" of $j(u)$, therefore the estimation accuracy of the statistical estimation.

It can be quite easily shown that the BLUE calculated above (assuming unbiased measurements) minimizes the following cost function too:

$$j(u) = \frac{1}{2}(u - z_1, u - z_2) \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}^{-1} \begin{pmatrix} u - z_1 \\ u - z_2 \end{pmatrix} = \frac{1}{2}\|u - z_{obs}\|_N^2 \tag{5.3}$$

In this cost function expression, the norm N takes into account correlations of the measurements errors. As a consequence, the extra diagonal terms are non vanishing.

Recall that: $\|\cdot\|_{\square}^2 =< \square\cdot, \cdot >$.

With m observations

The extension of the calculation above (the BLUE in the 1d case) is straightforward. Under the same assumption on each observation z_i, the BLUE reads:

$$\boxed{u^* = \frac{1}{Var(u^*)}(\sum_{i=1}^{m} \frac{1}{\sigma_i^2} z_i)} \text{ with } \frac{1}{Var(u^*)} = \sum_{i=1}^{m} \frac{1}{\sigma_i^2} \tag{5.4}$$

Therefore the BLUE u^* minimizes the following cost function: $\boxed{j(u) = \frac{1}{2}\sum_{i=1}^{m} \frac{1}{\sigma_i^2}(u - z_i)^2}$.

As previously, the Hessian (here a simple scalar second derivative) defines the "convexity rate"; it measures the analysis accuracy: $j''(u) = \frac{1}{Var(u^*)}$.

With correlated errors, the extended expression $j(u) = \frac{1}{2}\|u\mathbf{1} - \mathbf{z}\|_N^2$ holds with N defined as above.

The BLUE formulated as a sequential filter in the basic 1D case

In this case, the BLUE expression can be re-written as:

$$u^* = \frac{\sigma_2^2 z_1 + \sigma_1^2 z_2}{\sigma_1^2 + \sigma_2^2} = z_1 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)(z_2 - z_1)$$

Let us consider that:
a) z_1 is a first estimation. It is then called the *background* or the *first guess* value.
We denote this first-guess value by u_b : $u_b \equiv z_1$.
b) z_2 is an independent observation. We denote this newly obtained observation by z : $z \equiv z_2$.

Following this point of view, the BLUE u^* reads :

$$\boxed{u^* = u_b + (\frac{\sigma_b^2}{\sigma_b^2 + \sigma_0^2})(z - u_b)} \tag{5.5}$$

The term $(z - u_b)$ is called *the innovation* and denoted by d: $d = (z - u_b)$.

Using the terminology of sequential data assimilation, the equation reads as follows:
"The best estimation u^* equals the first guess + the *gain* times the *innovation*".

5.1.2 The BLUE in the general case

The linear estimation problem

Here, we seek to estimate $u = (u_1 ... u_n)^T \in \mathbb{R}^n$ given the measurements (observations) $z_{obs} = (z_{obs,1}, \ldots, z_{obs,m})^T \in \mathbb{R}^m$. We denote by ε_{obs} the observation errors.

$$\boxed{z_{obs} = Zu + \varepsilon_{obs} \quad u \in \mathbb{R}^n,\ z \in \mathbb{R}^m} \tag{5.6}$$

with Z a linear operator, $Z \in \mathcal{M}_{n \times m}$.

The error term ε_{obs} is assumed to be such that:
1) errors are unbiased: $E(\varepsilon_{obs}) = 0$.
2) covariances are known: $cov(\varepsilon_{obs}) = E(\varepsilon_{obs}\varepsilon_{obs}^T)$ is given.
We set:

$$R = cov(\varepsilon_{obs}) \tag{5.7}$$

The covariance matrix $cov(\varepsilon_{obs})$ is supposed to be definite therefore invertible. R^{-1} is symmetric positive definite therefore defining a norm.

In real-world problems, R is often assumed to be simply diagonal. Thus $R^{-1} = diag(\rho_{obs,1}, \ldots, \rho_{obs,m})$ with $\rho_{obs,i}$ being (a-priori) precisions on the observations.

Definition

A formal definition of the BLUE can be stated as follows.

Definition 5.1. *Considering the linear relation (5.6), the BLUE for u is the vector u^* such that:*

$$u^* = argmin_{v \in \mathbb{R}^n} \left(E[(v-u)^2] \right)$$

subject to $E(v) = u$ (unbiasedness) and the constraint (5.6) of course.

Given u_b a first estimate (also called background), the error of background reads $\varepsilon_b = (u_b - u_t)$ where u_t denotes the "true" solution i.e. the exact solution satisfying $z_t = Zu_t$.
The covariance matrix $cov(\varepsilon_b)$ is assumed to be definite therefore invertible. We set:

$$B = cov(\varepsilon_b) \tag{5.8}$$

B^{-1} is symmetric positive definite therefore defining a norm.
In practice, of course that u_t and $cov(\varepsilon_b)$ are unknown. However, the following results provide useful insights.

The central result

We have:

Proposition 5.2. *Under the assumptions on the error terms ε_{obs} and ε_b above, the two statements below hold.*

1) The expression of the BLUE u^* can be explicitly derived and reads:

$$u^* = u_b + K\,(z_{obs} - Zu_b) \tag{5.9}$$

with u_b the first estimate (background value) and the gain matrix K defined by:

$$K = BZ^T(R + ZBZ^T)^{-1} \tag{5.10}$$

with R and B defined by (5.7) and (5.8) respectively.

2) This expression of u^* above is also the unique minimum of a quadratic cost function $j(u)$:

$$u^* = argmin_{u\in\mathbb{R}^n} j(u) \text{ with } j(u) = \left(\|z_{obs} - Zu\|^2_{R^{-1}} + \|u - u_b\|^2_{B^{-1}}\right) \tag{5.11}$$

The general expression (5.10) of K of course simplifies in the scalar case as in (5.5): $K = (\frac{\sigma_b^2}{\sigma_b^2+\sigma_0^2})$.

Remark 5.3. *The expression of the functional $j(u)$ above is the starting point of the variational approach including for non linear estimation problems. The variational approach is presented in a next chapter.*

5.1.3 Examples

The reader may consult one of the numerous well documented Python codes available on the web, e.g. the https://towardsdatascience.com webpage [1], or e.g. on https://github.com/jolange/BLUE-py.

[1] https://towardsdatascience.com/linear-regression-with-ols-unbiased-consistent-blue-best-efficient-estimator-359a859f757e

5.2 The Kalman Filter

Filters are stochastic algorithms which operate recursively on streams of uncertain input data to produce a statistically optimal estimation of the underlying state. Note that *filters aim at estimating the state of the system (the model output) and not input parameters of the model* as it is done while employing a variational approach (see next Chapter).

5.2.1 The linear dynamic model and observations

Filters naturally apply to dynamical systems since they provide a sequence of states (time series).

$$\boxed{u^k = Mu^{k-1} + \varepsilon_{mod}^{k-1}} \tag{5.12}$$

where M is the transition operator which is here *linear*: M is a $n \times n$ matrix. ε_{mod} denotes the model error.
We assume that at the same time instants, observations z^k, $z^k \in \mathbb{R}^m$, are available, with:

$$\boxed{z_{obs}^k = Zu^k + \varepsilon_{obs}^k} \tag{5.13}$$

The observation operator Z is supposed to be linear too: Z denotes a $m \times n$ matrix.
Both the model errors ε_{mod} and the observation errors ε_{obs} are supposed to be Gaussian, given in $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively. They are supposed to satisfy:

$$\varepsilon_{mod}^k \sim \mathcal{N}(0, Q_k) \text{ and } \varepsilon_{obs}^k \sim \mathcal{N}(0, R_k)$$

where Q and R are the covariance matrices of the model and observation errors respectively.

5.2.2 The KF algorithm

Basic principles

At each iteration k, KF works in two steps:

`Step 1):` the forecast (prediction) step.
A first estimation u_f^k is computed as the solution of the dynamic model (5.12).

`Step 2):` the analysis (correction) step.
Given the newly acquired data z_{obs}^k, a corrected estimation of u^k, denoted by u_a^k, is computed. u_f^k plays here the role of a background value.
Because of the linearity of M and Z plus the assumptions on the errors previously mentioned, u_a^k *is defined as the BLUE.*

Then, the central KF scheme equation reads as follows: for $k \geq 1$,

$$\boxed{u_a^k = u_f^k + K^k(z_{obs}^k - Zu_f^k)} \tag{5.14}$$

with the gain matrix K^k acts as the weight of the innovation term $(z_{obs}^k - Zu_f^k)$, as in the BLUE. However, here in the expression of K^k, B is replaced (see (5.10) in the BLUE case) by the *forecast covariance errors matrix* P_f^k:

$$P_f^k = cov(\varepsilon_f^k) \text{ with } \varepsilon_f^k = (u_f^k - u_t^k)$$

Thus,

$$K^k = P_f^k Z^T \left(R + ZP_f^k Z^T\right)^{-1} \tag{5.15}$$

The analysis error is defined as $\varepsilon_a^k = (u_a^k - u_t^k)$. The related covariance errors matrix reads: $P_a^k = cov(\varepsilon_a^k)$. One can show that P_a^k satisfies, see e.g. [7] Chapter 2:

$$P_a^k = (I - K^k Z)P_f^k \tag{5.16}$$

Extreme cases: perfectly observed system / perfect model

- *Perfect model.* If the forecast errors tends to 0, that is $\|P_a^k\| \to 0$ then $K^k \to 0$ for all k.

- *Perfect data.* If the observations errors tends to 0, that is $\|R\| \to 0$ then $K^k \to Z^{-1}$ for all k.

The weighting of the innovation by the gain K may be read as follows.
As the measurement error covariances tend to 0, the observation z^{obs} is trusted more and more while the model response u_f is trusted less and less.
On the opposite, as the forecast error covariances tend to 0, the model response u_f is trusted more and more while the observation z^{obs} is trusted less and less.

Basic extreme cases In the simple case where:

- $Z = Id$,
- the covariance observation errors matrix is diagonal such that $R \equiv \Delta_R^{obs} = diag((\sigma_1^{obs})^2, \ldots, (\sigma_m^{obs})^2)$, $(\sigma_i^{obs})^{-2}$ the precision of the i-th data,

then the gain matrix simplifies as: $K^k = P_f^k \left(\Delta_R^{obs} + P_f^k\right)^{-1}$.

Moreover, if the forecast covariance errors matrix P_f^k is diagonal (and constant along the iterations) as $P_f^k = (\sigma_f)^2 Id$, $(\sigma_f)^{-2}$ the forecast precision, the gain matrix simplifies as in the 1D simple case:

$$K = diag\left(\frac{\sigma_f^2}{((\sigma_1^{obs})^2 + \sigma_f^2)}, \ldots, \frac{\sigma_f^2}{((\sigma_m^{obs})^2 + \sigma_f^2)}\right)$$

The KF algorithm

Initialization. The I.C. of the system state u^0 is given. We set: $\varepsilon^0 = (u^0 - u_t)$ and $P_f^0 = cov(\varepsilon^0) = E(\varepsilon^0(\varepsilon^0)^T)$.
The error covariance matrix P_f^0 is supposed to be given too (...).

From iteration $(k-1)$ to iteration k,

1) *Analysis step.*
- Compute the *Kalman gain matrix* K^k as in (5.15).
- Deduce the analysis value u_a^k as: $u_a^k = u_f^k + K^k(z_{obs}^k - Zu_f^k)$.
- Compute the covariance matrix P_a^k as in (5.16).

2) *Forecast step.*
- Solve the model to obtain the forecast value u_f^{k+1}: $u_f^{k+1} = Mu_a^k$.
- Compute the covariance matrix of forecast errors P_f^{k+1} as:
$P_f^{k+1} = MP_a^kM^T + Q^{k+1}$.

5.2.3 Examples

See one of the numerous well documented Python codes available on the web, e.g.:

- https://machinelearningspace.com/object-tracking-python/
- https://thekalmanfilter.com/kalman-filter-python-example
- https://github.com/Garima13a/Kalman-Filters
- https://arxiv.org/ftp/arxiv/papers/1204/1204.0375.pdf

5.2.4 Pros and cons of KF

⊕ Considering a linear model M and a linear observations operator Z, assuming that the covariances $cov(\varepsilon_{mod})$ and $cov(\varepsilon_{obs})$ are Gaussian, KF is the optimal sequential method, see e.g. [12].

⊕ At each iteration k, the new estimation u_a^k depends on the previous time step state u^{k-1} and the current measurement z^k only: no additional past information is required, see Fig. 5.2.

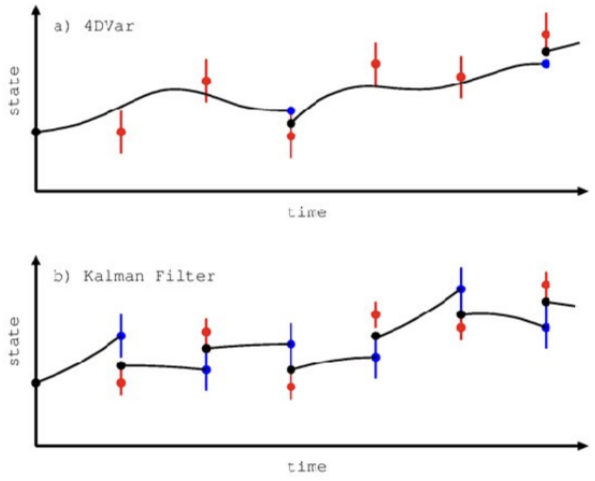


Figure 5.2: (Up) VDA algorithm (Down) KF algorithm. Figure extracted from [**?**].

⊖ The KF scheme (5.14) is bearly affordable for large dimension models ($\dim(u) >>$).
Indeed, to compute the gain matrix K, see (5.10), one needs to invert the matrix $(R + ZBZ^T)$.
Without specific properties of the matrix, in the case of n and/or m large, this requires high computational ressources.

⊖ The basic version of the KF does not apply to non-linear models.
In non-linear cases, KF can be extended by linearizing the "transition" operator M at each time step: this is the idea of the Extended Kalman Filter (ExKF).
However, the ExKF is not optimal anymore. Moreover, its use is limited too by its high computational requirements as the KF.

5.2.5 Extension to non-linear models and/or large dimensional problems: Ensemble KF (EnKF) and hybrid approaches

In summary,

`KF`	The KF is an algorithm that estimates the state of a linear dynamic system based on noisy measurements. It uses a recursive process to update the state estimate as new measurements become available. At each iterate, the estimator relies on the BLUE. The KF is widely used in various fields, including engineering, economics, and computer science. However, it is restricted to linear estimation problems of relatively small dimensions.
`EnKF`	The EnKF is an extension of the KF that addresses the limitations of the KF for large dimensional and/or non linear systems. It uses an ensemble of state vectors to represent the probability distribution of the system state. The EnKF updates the ensemble members based on observations and propagates them forward in time using a numerical model. It is widely employed in geosciences but not only.
`VDA`	As it will studied now, the variational method is based on another approach: it aims to find the optimal estimate of the system state by minimizing the cost function $j(u)$. It then relies on an optimization algorithm which iteratively adjusts the state (model prediction) $y(u)$ based on the observations z_{obs}. The variational approach is a good option to address large dimensional problems (m=dim(u) >>) and non linear problems (M and/or Z non linear). It is widely employed in geosciences but not only.

Note that the current state-the-art for operational large dimensional and complex multi-physics non-linear models consists to combine VDA with EnKF. Such hybrid approaches have been developed in particular in operational weather forecast centers like ECMWF or NOAA for example.

In the linear Gaussian case (this has to be clarified), each of these methods can be nicely related to a Bayesian analysis.

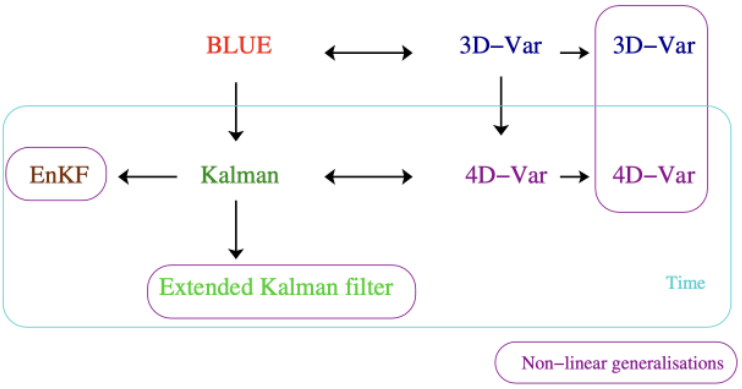


Figure 5.3: The classical DA methods. Image extracted from [7]

Chapter 6

DA by variational approach in simple cases

The outline of this chapter is as follows.

Contents

6.1 Introduction

In what follows, the inverse problem is as follows.

$$\begin{cases} \text{Estimate } u(x) \text{ such that:} \\ A(u(x); y(x)) = B(u(x)) \text{ in } \Omega \text{ accompanied with } B.C. \text{ on } \partial\Omega \\ \text{ with some observations of the system state available : } z_{obs}(x) = Z(y(x)). \end{cases} \tag{6.1}$$

$A(u; y)$ denotes a PDE-based operator.

Example. The physical-based model is defined by: $A(u; y) = -div(\lambda(u)\nabla y) + y^3$, with $\lambda(\cdot)$ a given function, and the RHS $B(u) = g(\|\nabla u\|)$ with $g(\cdot)$ a given function. The observation operator is defined by $z_{obs} = \partial_n y$ on a piece of the boundary Γ, $\Gamma \subset \partial\Omega$.

Contrary to the classical "explicit" least square problems, see (2.1), the unknown u is related to the measurements z_{obs} through the mathematical model $\mathcal{M}$, $\mathcal{M} : u \mapsto y(u)$ ($y(u)$ solution of (6.1) with u given).
As a consequence, even in the linear case, that is $\mathcal{M}(u)$ linear, the optimal solution $u^* = arg\min_u j(u)$ is not obtained by simply solving a linear system as the normal equations, see Section 2.1.1.

6.2 The VDA formulation

Let us present a typical VDA formulation for a general non linear however stationary PDE-based model.

6.2.1 The (direct) model and the parameter-to-state operator

Given the unknown/uncertain parameter u, find y which satisfies the Boundary Value Problem (BVP):

$$A(u(x)); y(x)) = B(u(x)) \text{ with } x \in \Omega \subset \mathbb{R}^d \tag{6.2}$$

accompanied by Boundary Conditions (BC).

The PDE-based operator $A(\cdot;\cdot)$ is a-priori non-linear both in u and y that is the maps $u \mapsto A(u;\cdot)$ and $y \mapsto A(\cdot; y)$ are non-linear.

The parameter-to-state map (also previously called the model operator) $\mathcal{M}$ is defined as:

$$\mathcal{M} : u(x) \mapsto y(u)(x)$$

6.2.2 The observation operator and the cost function

Data (also called observations or measurements) are denoted by z^{obs}. Data are not necessarily of same nature than the state of the system y.
Then, one needs to introduce an *observation operator* Z which maps the state of the system y onto the observations space as:

$$z(x) = Z(y(x)) \tag{6.3}$$

The observation operator $Z(\cdot)$ can be a linear or not.
Let us introduce the so-called observation function $J(u; y)$ defined as:

$$\boxed{J(u; y) \;=\; J_{obs}(y) + \alpha_{reg} J_{reg}(u)} \tag{6.4}$$

with

$$J_{obs}(y) = \|Z(y) - z^{obs}\|^2_{R^{-1}} \text{ and potentially } J_{reg}(u) = \|u - u_b\|^2_{B^{-1}} \tag{6.5}$$

Then the cost function $j(u)$ is defined from the observation function $J(u; y^u)$ as:

$$\boxed{j(u) = J(u\,; y^u) \text{ where } y^u \text{ is solution of (6.8).}} \tag{6.6}$$

The cost function $j(u)$ depends on the control u through the state of the system $y^u = \mathcal{M}(u)$.

6.2.3 The optimization problem

Solving the VDA problem consists to find:

$$\boxed{u^*(x) = arg \min_{u(x) \in \mathcal{U}} j(u(x))} \tag{6.7}$$

This is an optimal control problem.
The solution u^* (if existing) is an optimal control of the system. This is the estimate we are looking for. This is the VDA solution.

6.3 Linear model, finite dimensional case

In this section, the VDA solution is derived in the case *the model is linear and finite dimensional*.

6.3.1 Problem statement

Let u be the control variable, $u \in \mathbb{R}^m$. Let the model be linear and represented by $A(u)$, $A(u) \in \mathcal{M}_{n\times n}$ a non singular real matrix. The (direct) model reads:

$$\boxed{\begin{cases} \text{Given } u \in \mathbb{R}^m, \text{ find } y \in \mathbb{R}^n \text{ such that:} \\ A(u)\ y \ = \ Fu \end{cases}} \tag{6.8}$$

with F a rectangular matrix, $F \in \mathcal{M}_{n\times m},\ m < n$, of maximal rank m.

Let $J(u;y)$ be the objective function measuring the misfit between the state y and the observations defined as in (6.4) (6.5).
However, here the observation operator Z is linear: $Z \in \mathcal{M}_{n\times n}$. Therefore:

$$\boxed{J(u;y) = \|Zy - z^{obs}\|^2_{R^{-1}} + \alpha_{reg}\|u - u_b\|^2_{B^{-1}}} \tag{6.9}$$

The estimation problem consists to find $u^* \in \mathbb{R}^m$ such that: $\boxed{j(u^*) = \min_{U_h} j(u)}$.

6.3.2 On the numerical resolution of the VDA problem

Few approaches are a-priori possible: roughly, global optimization methods or local minimization methods. The choice mainly depends on the CPU time required to evaluate the cost function $j(u)$ and on the control variable dimension m, therefore the dimension of the gradient $\nabla j(u)$.

If the CPU time required to evaluate the cost function $j(u)$ is negligible (let say in fractions of seconds using your laptop or a super-computer, whatever), then one can adopt a *global optimization approach* based on stochastic algorithms e.g. Monte-Carlo type algorithms, heuristic methods (e.g. genetic algorithms) or surface response approximation.

On contrary, if the dimension of the parameter u is large ($m = dim(U_h) = O(10)$ and more), moreover if the computation of the cost function $j(u)$ is CPU-time consuming (this the case e.g. if considering a 3d PDE model), then global optimization is not worth considering. Then, one has to adopt *local minimization* approaches based on *algorithms of descent*.

6.3.3 Computing the cost function gradient $\nabla j(u)$

A brute force approach: approximation by Finite Differences

$$\nabla j(u) \cdot \delta u \approx \frac{j(u + \varepsilon \delta u) - j(u)}{\varepsilon} \tag{6.10}$$

with δu a given direction, $\delta u \in \mathbb{R}^m$.

This numerical approach requires $(m+1)$ evaluations of $j(u)$ therefore $(m+1)$ resolutions of the direct model. This is generally not possible for m large!...
Moreover, the descent algorithms can be sensitive to the gradient accuracy. Here, the obtained accuracy of $\nabla j(u)$ is not controlled since depending on an "optimal" choice of ε which cannot be a-priori known.

The few differentials

Recall that $j : \mathbb{R}^m \to \mathbb{R}$, $j(u) = J(u; y(u))$, with $J(u; y)$ defined by (6.5).

We have the gradients $\nabla j(u) \in \mathbb{R}^m$, $\nabla_u J(u; y) \in \mathbb{R}^m$ and $\nabla_y J(u; y) \in \mathbb{R}^n$.

We have the map (model operator) $\mathcal{M} : u \in \mathbb{R}^m \mapsto y(u) \in \mathbb{R}^n$.

We denote by $D_u y(u)$ the differential of y with respect to u, it is a $n \times m$-Jacobian matrix.
$D_u y(u)$ represents the derivative of the state $y(u)$ with respect to the parameter u.
Example. $D_u y(u)$ represents the derivative of a temperature field $y(x)$ with respect to the spatially-distributed source term $u(x)$.

A first expression of $\nabla j(u)$

Formally, the state of the system satisfies the relation: $y(u) = (A(u))^{-1} F u$.
Of course, in practice, one never compute A^{-1}, instead the linear system is solved by a linear algebra method (e.g. a Gauss-type algorithm).
However, the cost $j(u)$ reads explicitly in function of u as:

$$j(u) = J(u; y(u)) = \|Z(A(u))^{-1} F u - z^{obs}\|^2_{R^{-1}} + \alpha_{reg} \|u - u_b\|^2_{B^{-1}} \tag{6.11}$$

From the relation $j(u) = J(u; y(u))$, we get for all $v \in \mathbb{R}^m$,

$$\boxed{< \nabla j(u), v > = < \nabla_u J(u; y(u)), v) > + < \nabla_y J(u; y(u)), D_u y(u) \cdot v >} \tag{6.12}$$

where $< \cdot, \cdot >$ denotes here the Euclidian scalar products (either in $\mathbb{R}^m$ or in $\mathbb{R}^n$) with $y(u) = (A(u))^{-1} F u$.

We have:

$$\boxed{D_u y(u) = D_u \left((A(u))^{-1} \right) F u + (A(u))^{-1} F} \tag{6.13}$$

with $D((A(u))^{-1})$ the differential of the inverse of the direct model operator.

However, the expression (6.13) of $D_u y(u)$ is not tractable. This point is partly addressed below in a simple case; it will be addressed in general cases in subsequent chapters.

6.3.4 A simple case: u in the RHS only

For sake of simplicity, let us consider from now a simplified u-parametrized model: the parameter u appears in the RHS of the model only, and not in the differential operator A.
In this case, the inverse problem solution is much easier to characterize. Indeed, in this case, explicit calculations can be derived and we can easily introduce the concept of adjoint equations.

The simplified model equation

The u-parametrized direct model equation reads:

$$\boxed{Ay = Fu} \tag{6.14}$$

In this case, we simply have: $\boxed{D_u y(u) = A^{-1}F}$, expression to be compared to (6.13).

Then, for all $v \in \mathbb{R}^m$,

$$< \nabla j(u), v >=< \nabla_u J(u; y(u)), v) > + < F^T A^{-T} \nabla_y J(u; y(u)), v >$$

From the expression of $J(u; y)$, see (6.9), we get:

$$\nabla_u J(u; y(u)) = 2\alpha_{reg} B^{-1}(u - u_b) \text{ and } \nabla_y J(u; y(u)) = 2Z^T R^{-1}(Zy(u) - z^{obs})$$

Therefore the gradient expression:

$$\boxed{\nabla j(u) = \ F^T A^{-T} \nabla_y J(u; y(u)) + \ 2\alpha_{reg} B^{-1}(u - u_b)} \tag{6.15}$$

with $\nabla_y J(u; y(u)) = 2Z^T R^{-1}(Zy(u) - z^{obs})$.

Why an adjoint equation? Because it is not tractable to compute A^{-T}.
To avoid the computation of A^{-T}, we naturally solve the following (linear) equation:

$$\boxed{A^T p = \nabla_y J(u; y(u))} \tag{6.16}$$

This is the so-called *adjoint equation*. p denotes an additional field, $p \in \mathbb{R}^n$. It is the *adjoint state*.

Then, the gradient expression not explicitly dependent on A^{-T} reads as:

$$\boxed{\nabla j(u) = F^T p(u) + 2\alpha_{reg} B^{-1}(u - u_b)} \tag{6.17}$$

with $p(u)$ the unique solution of the (6.16).

Optimality condition

The 1st order necessary condition of optimality reads: $\nabla j(u) = 0$.

$$\begin{cases} \textit{The direct equation:} & Ay = Fu \\ \textit{The adjoint equation:} & A^T p = \nabla_y J(u; y(u)) \\ \textit{The 1st order necessary condition:} & \nabla j(u) = F^T p(u) + 2\alpha_{reg}\ B(u - u_b) = 0 \end{cases}$$

with here: $\nabla_y J(u; y(u)) = 2Z^T R^{-1}(Zy(u) - z^{obs})$.

6.4 Example

Let us consider a simplified version of the model equation arising in the spatial hydrology problem, see Section **??**.

$$-\Lambda_{ref}(x)\partial^2_{xx}H(x) + \partial_x H(x) \;=\; \partial_x b(x) \text{ in } (0,L) \tag{6.18}$$

with $\Lambda_{ref} = \frac{(H_{ref}-b_{ref})}{|\partial_x H_{ref}|}$. (H_{ref}, b_{ref}) are given functions such that $h_{ref}(x) = (H_{ref} - b_{ref})(x) \geq h_{min} > 0$ a.e.
The equation is closed with non-homogeneous Dirichlet boundary conditions.

The inverse problem as those presented in Section **??** is considered: infer $b(x)$ given somme measurements $H^{obs}(x)$.

The unknown parameter $b(x)$ appears in the RHS of the equation only (through its derivative): this case fits the general basic case addressed in Section 6.3.4.

Let us solve this inverse problem by a variational approach. This consists to solve the following optimization problem:

$$b^*(x) = arg \min_{b(x)\in\mathcal{B}} j(b(x)),$$

with $j(b) = J(b\,; H^b)$, H^b the (unique) solution of (6.18) given b. The observation function $J \equiv J_{\alpha_{reg}}$may be defined as follows:

$$J_{\alpha_{reg}}(b; H) \;= \|H - H^{obs}\|_2^2 + \alpha_{reg}\|b - b_b\|_2^2,$$

with b_b a background value (first guess).

Exercise 6.1. *1) Detail the direct model equations by employing either a Finite Differences or a Finite Element method.*
(It will be noticed that if considering real-like data, the numerical Peclet number of the equation is very low, therefore a centered scheme is here suitable).
Formulate the obtained numerical scheme in matrix form as: $AH = Fb$.
2) Write the optimality system which characterizes the solution b^* of the optimal control problem.
3) Propose an algorithm to numerically solve this inverse problem.
4) Employ the Python code provided on the course Moodle page to perform numerical solutions.

Chapter 7

Bayesian inferences & equivalences in the Linear Gaussian case

The outline of this chapter is as follows.

Contents

7.1 Bayesian analysis

7.1.1 Founding calculations

Problem statement

Let u be the parameter to be estimated given M observations $z_{obs} = (z_{obs,1}, \ldots, z_{obs,M})$, with:

$$\boxed{z_{obs} = \mathcal{M}(u) + \varepsilon} \tag{7.1}$$

with $\mathcal{M}$ denoting a non linear operator and ε the error term, $\varepsilon = (\varepsilon_{obs} + \varepsilon_{mod})$.
For a sake of simplicity, it is assumed here that: $\varepsilon_{mod} = 0$.

The Bayes law

The joint probability density of u and z_{obs} reads in terms of the conditional densities as:

$$p(u, z_{obs}) = p(u|z_{obs})p(z_{obs}) = p(z_{obs}|u)p(u)$$

This provides the Bayes's law:

$$p(u|z_{obs}) = \frac{p(z_{obs}|u)}{p(z_{obs})} p(u) \tag{7.2}$$

This relation can read as:

$$\text{Posterior} \propto (\text{ Likelihood} \times \text{Prior })$$

$p(u|z_{obs})$ is the *posterior* distribution (a-posteriori density).

In the present inverse problem context, the posterior $p(u|z_{obs})$ contain all information on the sough quantity u (given z_{obs}).
In practice, values of interest may be: the most probable value $u^* = arg\max_u p(u|z_{obs})$, i.e. the Maximum A-Posteriori (MAP), or the posterior mean $\bar{u} = \text{mean}(p(u|z_{obs}))$, or quantiles of $p(u|z_{obs})$.

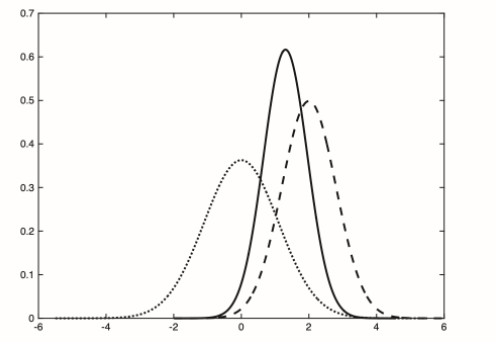


Figure 7.1: The distributions in a trivial scalar/univariate Gaussian case: prior (dotted), likelihood (dashed) and the resulting posterior (solid) Gaussian distributions. Bayes' law: Posterior $\propto$ Prior $\times$ Likelihood. Figure extracted from [1].

Remark 7.1.

- *The prior distribution $p(z_{obs}|u)$ can be numerically approximated by a Monte-Carlo method by performing the direct model $\mathcal{M}$ a very large number of times (let us say $O(10^4)$ and more). This implies to tackle a low CPU-time consuming model $\mathcal{M}(u)$.*

- *If a lot of data are available, the choice of the prior doesn't matter so much.*
  On the contrary, if not so much data are available, the choice of a relevant prior (the background information) becomes crucial since the result (the posterior) highly depends on the prior...

- *For large scale real-world problems, the priors may be quite subjective, as a consequence the resulting posteriors become subjective too...*

- *Since the posterior $p(u|z_{obs})$ results from the data z_{obs}, it is expected that it is less variable than the prior $p(u)$....*

7.1.2 The most probable parameter u

As estimator let us consider *the most probable parameter value* u.
Given the observations z_{obs}, given the background value u_b, the most probable parameter satisfies:

$$\boxed{u^* = arg\max_u \left(p(u|z_{obs} \text{ and } u_b)\right)} \tag{7.3}$$

Since the function $(-\log)(\cdot)$ is monotonic decreasing therefore convex, the optimization problem above can be equivalently written by minimizing the following functional:

$$\boxed{j(u) = -\log(p(u|z_{obs}) + c} \tag{7.4}$$

with c denoting any constant value. Then the most probable parameter u satisfies:

$$\boxed{u^* = arg\min_u j(u)}$$

Assuming that the observation errors and the background errors are uncorrelated (this is a very reasonable assumption), we get:

$$p(z_{obs} \text{ and } u_b|u) = p(z_{obs} \text{ and } u_b)p(u)$$

Using the Bayes law (7.2), it follows:

$$j(u) = -\log p(z_{obs}|u) - \log p(u) + \text{ c} \tag{7.5}$$

for any constant c.

The calculations above are valid for any distributions $p(u)$ and $p(z_{obs}|u)$.
From now, if considering Gaussian PDFs, the choice of the log function in the definition of $j(u)$ turns out to be judicious...

7.2 Assuming Gaussian PDFs

Let us assume from now that $\varepsilon_{obs} \sim \mathcal{N}(0, \sigma_{obs})$.

7.2.1 Scalar / univariate case

For sake of simplicity, we here consider the *scalar / univariate case*: the parameter u is a scalar function.

The prior distributions

Let us assume the prior distribution $p(u)$ is Gaussian: $p(u) \sim \mathcal{N}(u_b, \sigma_u^2)$, that is

$$p(u) = \frac{1}{(2\pi)^{1/2}\sigma_u} \exp\left(-\frac{1}{2\sigma_u^2}(u - u_b)^2\right) \tag{7.6}$$

We get M observations: $z_{obs} = (z_{obs,1}, \ldots, z_{obs,M})$.
Assuming $p(z_{obs}|u) \sim \mathcal{N}(\overline{z_{obs}}, \sigma_{obs}^2)$ *and that the* M *data are all independent*, we get:

$$p(z_{obs}|u) = \Pi_{m=1}^{M} \frac{1}{(2\pi)^{1/2}\sigma_{obs}} \exp\left(-\frac{1}{2\sigma_{obs}^2}(z_{obs,m} - \overline{z_{obs}})^2\right)$$

that is:

$$p(z_{obs}|u) \propto \exp\left(-\frac{1}{2\sigma_{obs}^2} \sum_{m=1}^{M} (z_{obs,m} - \overline{z_{obs}})^2\right) \tag{7.7}$$

Recall that given a Gaussian prior $p(u)$, *if* the model operator $\mathcal{M}$ is *linear* then the likelihood $p(z_{obs}|u)$ is Gaussian.
On the contrary if $\mathcal{M}$ is non linear then the likelihood $p(z_{obs}|u)$ is non Gaussian.

Resulting posterior expression

If both $p(u)$ and $p(z_{obs}|u)$ are Gaussian then the posterior $p(u|z_{obs})$ is Gaussian too, as the product of two Gaussians.
Indeed, by applying the Bayes law (7.2), the posterior reads as:

$$p(u|z_{obs}) \propto \exp\left(-\frac{1}{2\sigma_u^2}(u - u_b)^2 - \frac{1}{2\sigma_{obs}^2} \sum_{m=1}^{M} (z_{obs,m} - \overline{z_{obs}})^2\right) \tag{7.8}$$

which is equivalent too (see e.g. [2] Chap. 3 for detailed calculations):

$$p(u|z_{obs}) \propto \exp\left(-\frac{1}{2\sigma_p^2}(u - \overline{u_p})^2\right) \tag{7.9}$$

$$\text{with } \overline{u_p} = \sigma_p^2 \left(\frac{u_b}{\sigma_u^2} + \sum_{m=1}^{M} \frac{z_{obs,m}}{\sigma_{obs}^2}\right) \text{ and } \sigma_p^2 = \left(\frac{1}{\sigma_u^2} + \frac{M}{\sigma_{obs}^2}\right)^{-1} \tag{7.10}$$

7.3 The Maximum A-Posteriori (MAP) in Gaussian cases: equivalences with the BLUE & the variational solution

7.3.1 Computing the MAP

Recall (7.3)-(7.5): the most probable estimation (=the MAP) satisfies:

$$u^* = arg\max_u p(u|z_{obs}) = arg\min_u j(u) \tag{7.11}$$

by setting adequate constant values.

By using the Gaussian forms distributions, see (7.6)(7.7) in the scalar case, it follows that:

$$\boxed{u^* = arg\min_u j(u) \text{ with } j(u) = \frac{1}{2}\|\mathcal{M}(u) - \overline{z_{obs}}\|^2_{\sigma^{-1}_{obs}} + \frac{1}{2}\|u - u_b\|^2_{\sigma_u^{-1}}} \tag{7.12}$$

In summary, computing the most probable posterior value (= the MAP) involves maximizing a product. Next, by considering the log-likelihood function and under Gaussian assumptions leads to the minimization of the quadratic cost function $j(u)$ defined in (7.12).

Remark 7.2. *The term $\|u - u_b\|^2_{\square}$ in the definition of $j(u)$, see (7.12) can be read as a Tikhonov regularization term, see Section 2.2.1. In the present probabilistic point of view, this "regularization" term corresponds to the prior $p(u)$ whose is assumed to be Gaussian.*

7.3.2 Equivalences in the Linear Quadratic Gaussian (LQG) case

Let us recall the linear estimation problem considered in this chapter: estimate u satisfying $z_{obs} = (\mathcal{M}(u) + \varepsilon)$, with $\mathcal{M}$ denoting a non linear operator, ε the errors term, $\varepsilon = (\varepsilon_{obs} + \varepsilon_{mod})$. For a sake of simplicity, it is assumed here that $\varepsilon_{mod} = 0$.

Let us assume that:

- the model $\mathcal{M}$ is linear and $Z \equiv \mathcal{M}$, (therefore $Z \in \mathcal{M}_{n\times m}$),
- the error model is Gaussian with $\varepsilon_{obs} \sim \mathcal{N}(0, R^{-1})$.

We then solve:

$$z_{obs} = Zu + \varepsilon_{obs} \tag{7.13}$$

Then it has been demonstrated that:

- the Best Linear Unbiased Estimator (BLUE), which is the simplest statistic estimator, is equivalent to minimize the following quadratic functional (see Prop. 5.2):

$$j(u) = \left(\|z_{obs} - Zu\|^2_{R^{-1}} + \|u - u_b\|^2_{B^{-1}}\right) \tag{7.14}$$

  where the norms R^{-1} and B^{-1} are defined as $R = (cov(\varepsilon_{obs}))$ and $B = (cov(\varepsilon_b))$ respectively.

- if the prior distribution $p(u)$ is Gaussian, then the most probable solution u^* (= the MAP) coincides with the BLUE, provided we use the same B^{-1} and R^{-1} norms as above.
  The MAP is nothing else than the (unique) optimum of the Quadratic function $j(u)$ defined by (7.14), considering the appropriate covariances matrices R and B[1].

Recall that the *VDA method* aims at minimizing the functional $j(u)$ above.

Finally, in the LQG case, both the VDA solution and the sequential KF estimation are fully interpretable through Bayesian analysis since the three approaches (deterministic VDA, statistic BLUE / sequential KF, Bayesian estimation) mathematically yield the same estimation u^*.
However, each approach lead to different algorithm to compute u^*, thus coming with distinct advantages and drawbacks.

[1] In the LQ case, the cost function $j(u)$ is strictly convex therefore admitting a unique minimum. This point is mathematically shown in a subsequent chapter.

7.4 Numerical computations

7.4.1 Algorithm

In the computational point of view, given a Gaussian prior $p(u)$ and assuming a Gaussian likelihood $p(z_{obs}|u)$, the posterior $p(u|z_{obs})$ is numerically approximated as follows.

- Define a sufficiently fine grid (sampling) of the parameter space $\mathcal{U}$, $\mathcal{U} \subset \mathbb{R}^n$.
- Compute an approximation of the prior $p(u)$ by evaluating N_u values $p(u_n)$, $n = 1, \ldots, N_u$. We have: $N_u = O(n_0^n)$ with n_0 large enough (that is $n_0 = O(10)$ at least ?).
- Compute an approximation of the likelihood distribution $p(z_{obs}|u)$ as follows.
  - Perform the N_u model outputs $\mathcal{M}(u_n)$, $n = 1, \ldots, N_u$,
  - Given a fixed Gaussian likelihood form as $\mathcal{N}(\overline{z_{obs}}, \sigma_{obs})$, evaluate the probability $p(z_{obs}|u_n)$ as:

$$p(z_{obs}|u_n) = \frac{1}{(2\pi)^{1/2}\sigma_{obs}} \exp\left(-\frac{1}{2\sigma_{obs}^2}(\overline{z_{obs}} - \mathcal{M}(u_n))^2\right) \text{ for } n = 1, \ldots, N_u,$$

  - Evaluate the approximation of the likelihood $p(z_{obs}|u)$ as:

$$p(z_{obs}|u) \approx \Pi_{n=1}^{N_u} p(z_{obs}|u_n)$$

- Deduce the approximation of the posterior distribution $p(u|z_{obs})$ as:

$$p(u|z_{obs}) \propto \text{ Prior } p(u) \times \text{ Likelihood } p(z_{obs}|u)$$

7.4.2 Pros and cons

$\oplus$ The Bayesian analysis provides the richest information one can expect on the tackled inverse problem: the complete posterior distribution $p(u|z_{obs})$ from which one can next deduce the most probable value (= the MAP), the posterior mean and quantiles.

$\oplus$ The algorithm can be applied by performing the model $\mathcal{M}(\cdot)$ as a black box only. In this sense, it is a non-intrusive method.

$\ominus$ The approach is not tractable neither for CPU-time consuming models $\mathcal{M}(\cdot)$ nor for non tiny parameter dimension n.

$\ominus$ The posterior directly results from both the prior and the likelihood; and the latter can be very badly known...
Moreover, even if the prior distribution of u is supposed to be Gaussian (which may be a reasonable assumption), the resulting likelihood $p(z_{obs}|u)$ is not Gaussian in the case of a non linear operator $\mathcal{M}$.

7.5 Examples

The reader can find numerous well documented examples with corresponding Python codes available on the web, e.g. on the https://towardsdatascience.com web site [2], [3].

[2] https://towardsdatascience.com/how-to-use-bayesian-inference-for-predictions-in-python-4de5d0bc84f3
[3] https://towardsdatascience.com/estimating-probabilities-with-bayesian-modeling-in-python-7144be007815

Towards large dimensional problems

In the case of large dimensional problems, i.e. with m =dim(u) >>, the Bayesian analyses and the KF-based filters are barely tractable since too highly CPU-time and memory consuming.
On the contrary, the VDA method "naturally" extends to large dimension cases and to non linear models too. Indeed, the corresponding algorithms (3D-Var, 4D-Var and so on, see subsequent chapters) are well-suited for large dimensional (and non linear) problems since based on local gradient-based optimisation algorithms. However, uncertainties are not provided by the VDA method.

Chapter 8

DA by artificial neural networks

8.1 Artificial Neural Networks (ANNs)

8.1.1 Introduction

In essence, *well-trained ANNs can be seen as highly effective multi-scale interpolators.*
Additionally, *supervised learning applied to physically-based data provides an alternative approach for building effective models and solving inverse/identification problems.*

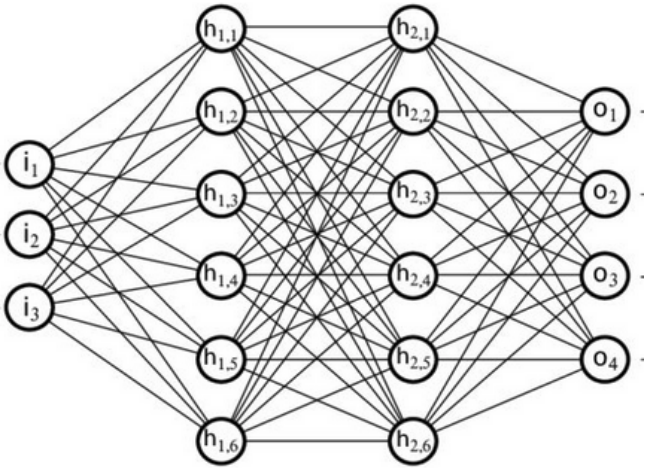


Figure 8.1: A small ANN with 2 hidden layer ($L = 2$). For each connection correspond a weight parameter value and a bias value. Image extracted from [.].

8.1.2 ANNs structure

Let us consider a dataset $\mathcal{D}$ containing the pairs (X_s^{obs}, Y_s^{obs}), $s = 1, \cdots, N_s$,

Let us denote by $\mathcal{N}_\theta$ an ANN composed of L hidden layers with $\theta = (W, b) \in \mathbb{R}^{N_\theta}$ as its parameters Let us denote: by f_l the l-th layer function, $f_l : x_{l-1} \in \mathbb{R}^{N_{l-1}} \to x_l \in \mathbb{R}^{N_l}$.

$$\boxed{\begin{array}{l}\mathcal{N}_\theta : x \in \mathbb{R}^n \mapsto y(\theta)(x) \in \mathbb{R}^p \\ \text{with } \mathcal{N}_\theta(x) = \left(f_{L+1} \circ \ldots \circ f_1\right)(x)\end{array}} \tag{8.1}$$

8.1.3 ANNs training: the optimization problem

Let us consider a dataset $\mathcal{D}$ containing data pairs (X_s^{obs}, Y_s^{obs}), $s = 1, \cdots, N_s$.

$$J_{obs}(\mathcal{D}) = \|Y^{obs} - \mathcal{N}_\theta(X^{obs})\|^2_{2,N_s} = \frac{1}{N_s}\sum_{s=1}^{N_s}\left(Y_s^{obs} - \mathcal{N}_\theta(X_s^{obs})\right)^2 \tag{8.2}$$

Training an ANN consists to minimize this misfit functional $J(\mathcal{D})_{obs}$ *with respect to the NN parameters* θ. Then we define the functional to be minimized as:

$$j_{\mathcal{D},obs}(\theta) = J_{obs}(\mathcal{D}) \tag{8.3}$$

Thus, training the ANN consists to solve the following differentiable optimization problem:

$$\theta^* = \min_\theta j_{\mathcal{D},obs}(\theta) \tag{8.4}$$

For deep ANNs, θ is extremely high dimensional e.g. $\mathcal{O}(10^q)$ with $q = 6$ and more.

After training,

$$\mathcal{N}_{\theta^*} \approx \mathcal{F} \text{ where } \mathcal{F} : X^{obs} \mapsto Y^{obs} \tag{8.5}$$

8.1.4 Trained ANNs: surrogate models

8.2 ANNs to solve inverse problems

Let us consider the same general u-parametrised PDE-based model as in (6.2):

$$A(u;y)(x) = B(u)(x) \text{ for } x \in \Omega \tag{8.6}$$

The physical-based model $A(\cdot;\cdot)$ is a-priori non-linear both in u and y.

8.2.1 Fully-parametrized ANN

Let us consider the pair $(x;u)$ as the input variable of the ANN: x the space variable, u the PDE parameter.
Note that u may be a spatially distributed parameter. In this case, we have $u(x)$.

Given an output quantity of interest $g(y)$, y the model output (= the state of the system), the ANN can be schematized as on Fig. 8.2.

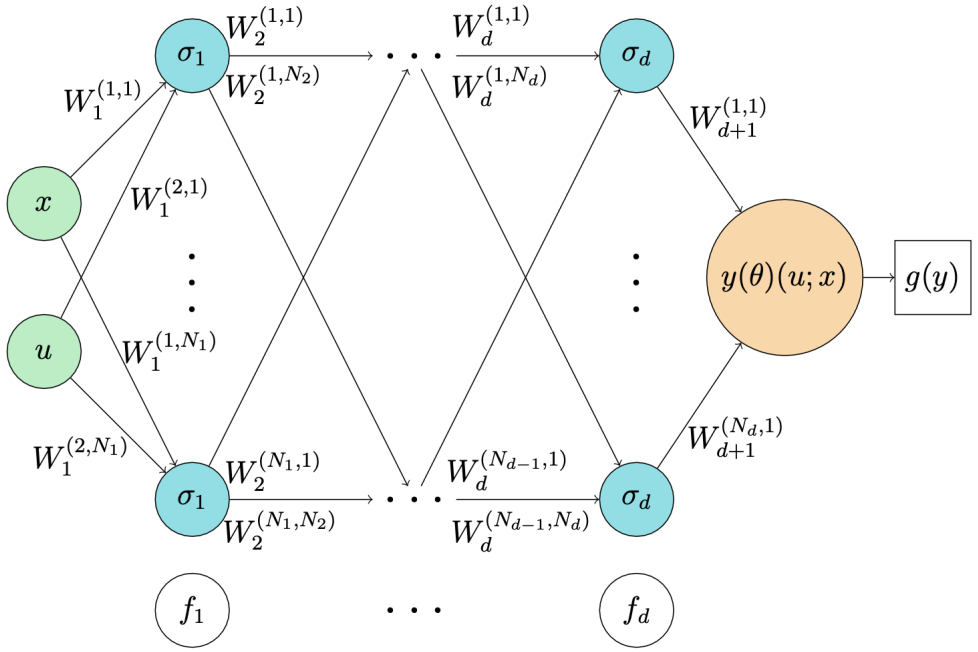


Figure 8.2: ANN to approximate the output of a u-parametrized model : the fully-parametrized ANN version.

(For a sake of clarity, the biases b_l are not indicated on the figures).

After learning, the trained ANN $\mathcal{N}_{\theta^*}$ is expected to be an approximation of the model operator $\mathcal{M}$ defined as:

$$\mathcal{N}_{\theta^*} \approx \mathcal{M} \text{ with } \mathcal{M} : (u;x) \mapsto y(u;x) \tag{8.7}$$

with $y(u;x)$ satisfying the model equation (8.6).

Moreover, under the assumption that u is very low dimensional, $\dim(u) = O(1)$, one should be able to infer values of u given data $y^{obs}(x)$.
Indeed, in this last case, a basic optimization procedure enables to infer u from y^{obs}, that is to solve the inverse problem consisting to identify u given $y(x)$.

8.2.2 Semi-parametrized ANN

For larger dimensional parameter u, say up to $\dim(u) = O(10^q)$ $q \approx 2$, a so-called semi-parametrized version of NN can be a good approach. This consists to build an ANN as indicated on Fig. 8.3.

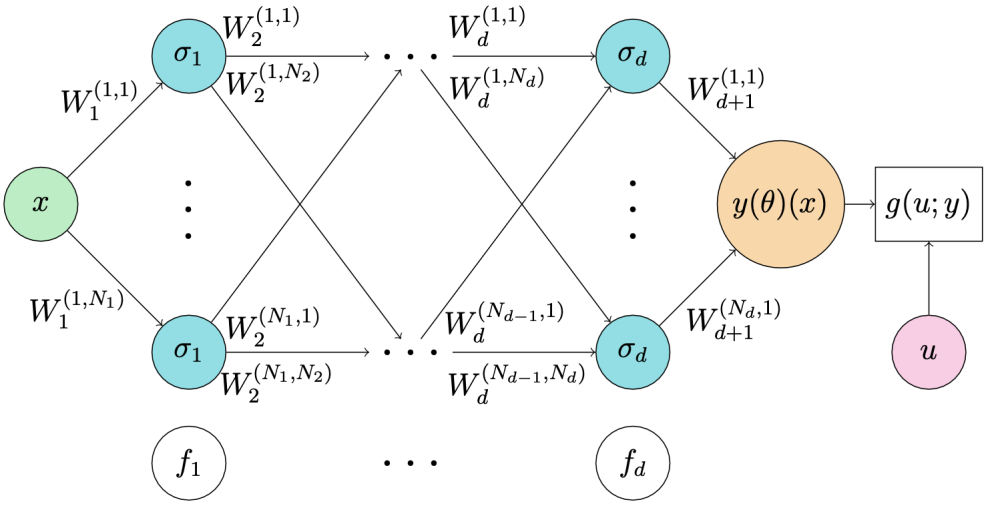


Figure 8.3: ANN to approximate a parametrized PDE-based model : the semi-parametrized version.

8.3 Physics-Informed Neural Networks (PINNs)

An important drawback of ANNs is their lack of explainability and reliability.
PINNs consist to minimize both the misfit to the observations (the usual term J_{obs}) and *an additional term: the residual of a given physical model* (term denoted by J_{res}).

8.3.1 Basic formalism

The residual of the model reads here:

$$r(u; y) = A(u; y) - L(u) \tag{8.8}$$

The loss function J to be minimized is then constituted by the standard misfit term J_{obs} as in (8.2), plus the residual functional $J_{res}(u; y)$ defined by:

$$J_{res}(u; y) = \|r(u; y)\|_{2,\mathcal{X}_{col}} \tag{8.9}$$

where $\mathcal{X}_{col}$ denotes a set of points within the domain Ω. It is may be perceived as collocation points where the residual is evaluated.

The total loss function J, $J : \mathcal{U} \times \mathcal{Y} \to \mathbb{R}$, then reads:

$$J_\alpha(u; y) = J_{\mathcal{D},obs}(y) + \alpha J_{res}(u; y) \tag{8.10}$$

Training a PINNs as proposed in [43] consists to solve the optimization problem (8.4), that is

$$\theta^* = \min_\theta j_{\mathcal{D},obs}(\theta),$$

with the loss function defined by $j_{\mathcal{D},obs}(\theta) = J_\alpha(u; y(\theta))$, $J_\alpha(u; y)$ defined by (8.10), and with a semi-parametrized architecture as those indicated on Fig. 8.4 or 8.4.

PINNs rely on Automatic Differentiation Finally, the most important remark relies on the way to compute the residual $r(u; y) = A(u; y) - L(u)$.
Automatic Differentiation of the ANN can provide any partial derivative $\partial^q_{x_j \ldots x_l} y$ therefore simply evaluating the residual (8.8). This is very likely the most important trick of the PINNs concept, [43, 37].

Minimizing the residual vs the model error

Given u, we have an unique (exact) solution y^* satisfying: $r(u; y^*) = A(u; y^*) - L(u) = 0$.
Given an approximation $\tilde{y}$ of y^*, we have of course: $r(u; \tilde{y}) = A(u; \tilde{y}) - L(u) \neq 0$.
We denote the error by $\varepsilon(y)$: $\varepsilon(y) = (y^* - \tilde{y})$.

Let us assume that $A(\cdot; y)$ is linear in y. In this case, we have:

$$r(u; \tilde{y}) = A(u)\tilde{y} - A(u)\tilde{y} = A(u)\varepsilon(y)$$

Therefore

$$\varepsilon(y) = A^{-1}(u)\ r(u; \tilde{y}) \tag{8.11}$$

$$\|\varepsilon(y)\|_2 \leq \|A^{-1}(u)\|_2 \|r(u; \tilde{y}))\|_2 \leq \frac{1}{\max_i |\lambda_i(A)|} \|r(u; \tilde{y}))\|_2 \tag{8.12}$$

This estimation shows that if the residual vanishes then the error vanishes too. However, a "small residual" does not necessarily implies a "small error". Indeed this depends on the spectrum of A.

8.3.2 PINNs for direct modeling

The architecture indicated on Fig. 8.4 relies on the direct-model without considering its parametrisation: u is here given, fixed. Here, we seek to solve the (formal) equation $A(y) = L$ in Ω by employing a PINNs.

After training, the PINNs is supposed to provide a surrogate model, such that:

$$\mathcal{N}_{\theta^*}(x) \equiv \tilde{y}(\theta^*)(x) \approx y^*(x) \tag{8.13}$$

with $y^*(x)$ the (unique) solution of the model, see Fig. 8.4.

The architecture of the PINNs for direct modeling is indicated on Fig. 8.4.

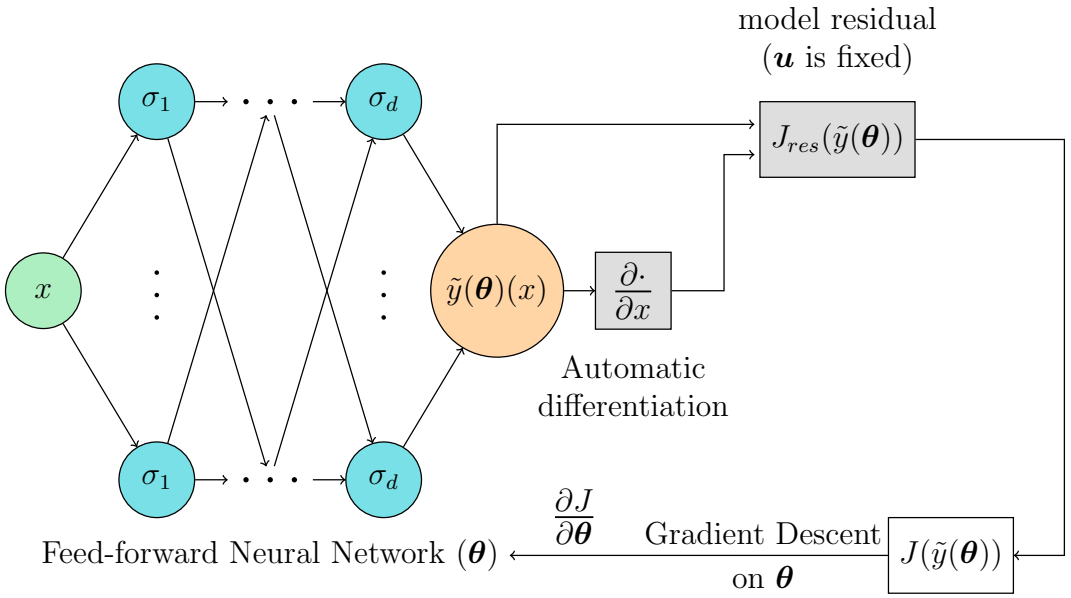


Figure 8.4: PINNs-like architecture to build a surrogate direct model. After training, the NN output approximates the observations while minimizing the residual of a physical model.

8.3.3 PINNs for inverse modeling

Here, we seek to solve the (formal) u-parametrized equation $A(u; y) = L$ in Ω by employing a PINNs *and* to estimate the parameter u^* corresponding to the given observations set.

After training, the PINNs is supposed to provide a surrogate model, such that:

$$\mathcal{N}_{\theta^*}(u; x) \equiv \tilde{y}(\theta^*)(u; x) \approx y^*(u^*; x) \tag{8.14}$$

with $(u^*, y^*)(x)$ the solution of the inverse problem.

The architecture of the PINNs to solve this inverse problem is indicated on Fig. 8.4.

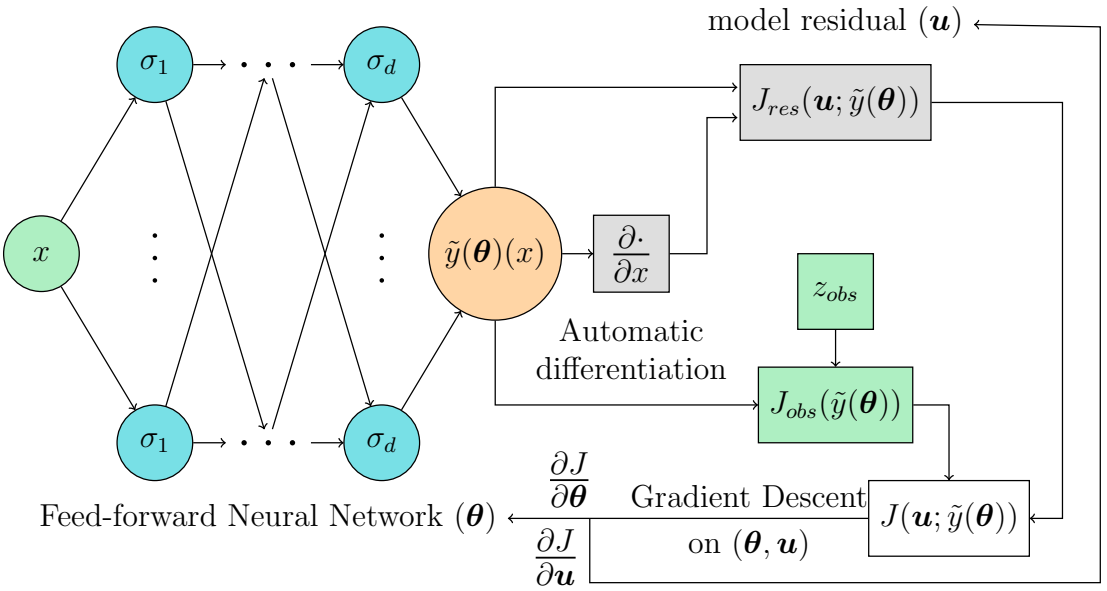


Figure 8.5: PINNs-like architecture to solve a parameter identification problem. After training, both u^* and $y(u^*)$ are estimated.

8.4 Examples

A reference library addressing PINNs-like architecture is the DeepXDE library [1]. Various examples of direct and inverse problems solved by PINNs-like architectures are proposed.

For examples locally developed at IMT-INSA Toulouse, please consult the supplementary material.

[1] https://deepxde.readthedocs.io/en/latest

Part III

Variational Approaches

Chapter 9

Optimal Control of ODEs

The outline of this chapter is as follows[1]

Contents

[1]Recall that the sections indicated with a * are "to go further sections". They can be skipped in a first reading or if the reader is not particularly interested in deeper mathematical basis, mathematical proofs.

9.1 Example: dynamic control of a vehicle

This problem is numerically solved by the code provided on the INSA Moodle page of the course.

9.1.1 The model

Model equation in the original position variable $x(t)$

At instant t, the vehicle position is represented by $x(t)(m)$, its velocity by $x'(t)(ms^{-1})$.
The goal is to control the vehicle trajectory by acting on a command $u(t)$ e.g. the engine power.
We have $x : [0, T] \to \mathbb{R}$ and $u : [0, T] \to \mathbb{R}$.
Let m (kg) be the vehicle mass, u be the pedal position (in %).
The dynamic trajectory model may simply reads as:

$$m\, x"(t) = -K\ x'(t) + G\ u(t) \text{ in } (0, T) \tag{9.1}$$

with K a friction coefficient (Ns/m) and G a gain parameter $(ms^{-1}(\%\text{pedal})^{-1})$.

Model equation in variable $y(t) = x'(t)$ (velocity variable)

By making the change of variable $y(t) = x'(t)$, the state equation simply reads:

$$y'(t) = -k_1 y(t) - k_2\ u(t) \text{ in } (0, T) \tag{9.2}$$

with $k_\square$ constant parameters of the model.

9.1.2 Inverse problems

- Given a target velocity z_{target}, identify a value of $u(t)$ such that the vehicle reaches this velocity value z_{target} at time T given, by consuming a minimum of energy (minimal value of u).

- A more dynamic version of the problem would be as follows. Given a target velocity $z_{target}(t)$, identify a value of $u(t)$ such that the vehicle sticks as close as possible to $z_{target}(t)$, moreover by consuming a minimum of power, moreover by considering bounded accelerations, etc.

One can mathematically translate the inverse problem above by minimizing the following functional (called cost function):

$$j_\alpha(u) = \frac{1}{2}\int_0^T |y^u(t) - z_{target}(t)|^2 dt + \alpha\frac{1}{2}\int_0^T \|u(t)\|_N^2 dt \tag{9.3}$$

with y^u the unique solution of the model equation, given u.

The optimal control problem reads:

$$\min_{u(t)\in\mathcal{U}} j_\alpha(u) \tag{9.4}$$

Remarks

- The cost function $j(u)$ depends on u explicitly through its second term, but also through $y^u(t)$ in its first term: this is why it is a optimal control problem and not simply a standard optimization problem.

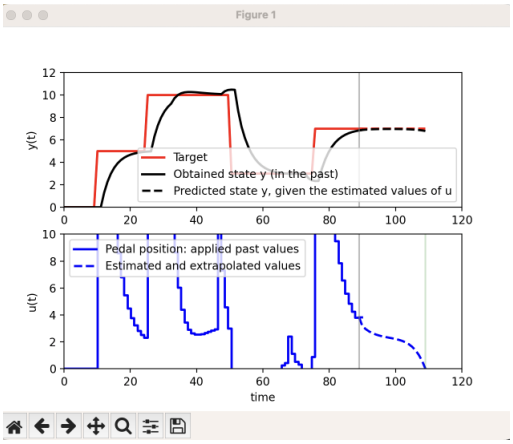


Figure 9.1: Optimal control of a vehicle: given a velocity target $z_{target}(t)$, what is the optimal control value $u(t)$ while imposing "reasonable" accelerations ? Figure plotted by the ocmputational code provided on the webpage of the course.

9.2 Introductory remarks

9.2.1 ODE solution behaviors: simple examples

The simple ODE-based model: spring - mass system

Let us consider here a similar basic example: the dynamics of a spring - mass system, see Fig. 9.2.
The mass m is submitted to a force $f(y)$, which is supposed here to be equal to:$-[k_1(y-L)+k_2(y-L)^3]$.

Given the external force $u(t)$, the spring position $y(t)$ is described by the differential equation:

$$my''(t)+k_1(y(t)-L)+k_2(y(t)-L)^3=u(t) \text{ for } t\geq 0$$

Different states of the system according to different control values

Case $u(t)=0$ i.e. without any external action.
In this case the equations reads:

$$y''(t)+y(t)+2y(t)^3=0$$

It is a particular case of the Duffing equation.
Its solutions $y(t)$ satisfy: $y(t)^2+y(t)^4+y'(t)^2=c$, c a constant.
All solutions are periodic and can be represented by an algebraic curve. The phase diagram and the trajectory are plotted on Fig. 9.2.

Case $u(t)=-y'(t)$**.** In this case, the applied external force aims at damping the spring. The equation reads:

$$y''(t)+y(t)+2y(t)^3+y'(t)=0$$

The numerical solution is computed, then the phase diagram and trajectory are obtained, see Fig. 9.2.

Using the Lyapunov theory, it can be shown that the origin is asymptotically stable. The spring position and the velocity reach the equilibrium position in infinite time, not in finite time.

Case $u(t) = -(y(t)^2 - 1)y'(t)$**.** With this control expression, the model is a particular case of the classical *Van der Pol* equation:

$$y''(t) + y(t) + 2y(t)^3 + (y(t)^2 - 1)y'(t) = 0$$

Two different solutions are computed and plotted on Fig. 9.2 (phase diagram and trajectories).
Using the Lyapunov theory, it can be proved that it exists a periodic solution which is attractive (plotted on Figure 9.2).

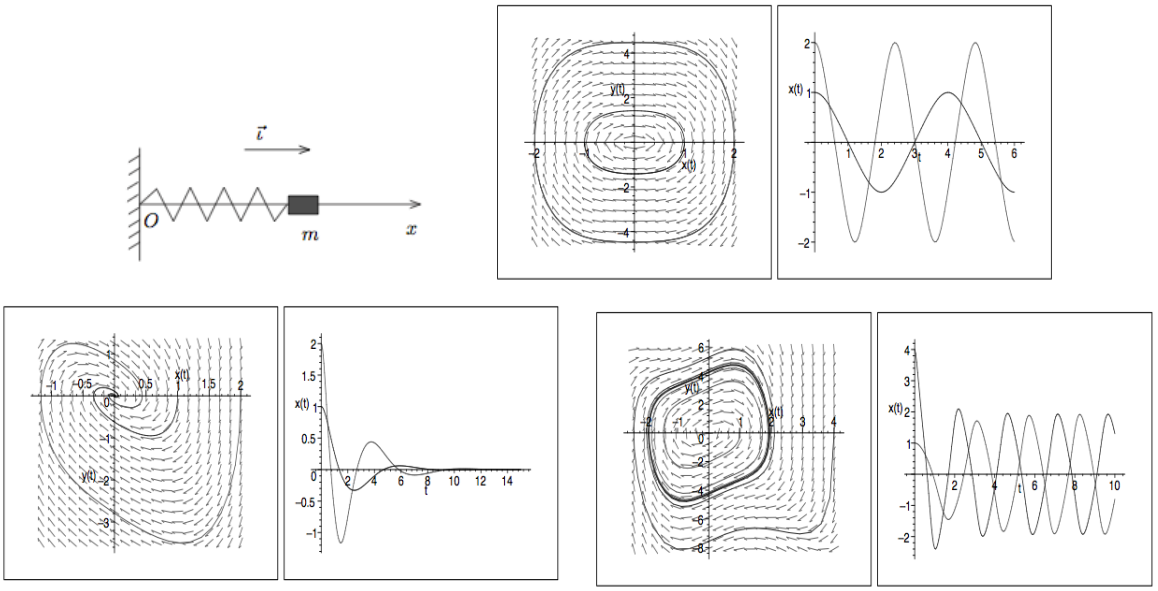


Figure 9.2: The spring-mass system example from [48]. (Up)(L) The spring-mass system. (Up)(R) The solution (state of the system) without control. (Down) The solution: (L) with damp control, (R) with another control value leading to the Van der Pol equation.

These three examples simply illustrate the wide range of behaviours which can be obtained by simply changing the control expression, even in the case of a gentle scalar equation.

9.3 The Linear-Quadratic (LQ) problem

9.3.1 The general linear ODE-based model

Let A, B and S be three mappings defined from $I =]0, T[$ into $\mathcal{M}_{n,n}(\mathbb{R})$, $\mathcal{M}_{n,m}(\mathbb{R})$ and $\mathbb{R}^n$ respectively.

$$\left\{ \begin{array}{l} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) + S(t) \text{ for } t \in I = [0, T] \\ \text{with the initial condition } y(0) = y_0. \end{array} \right. \tag{9.5}$$

Explicit expression of the solution

A classical result states that (9.5) has one and only one solution $y(t)$, $y(t)$ continuous from I into $\mathbb{R}^n$. Moreover an explicit expression of y in integral form holds.

Let us consider for sake of simplicity the case $S = 0$. In this case, we have:

$$y(t) = M(t)y_0 + M(t) \int_0^t M(s)^{-1} B(s)u(s)ds \tag{9.6}$$

with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t)$, $M(0) = Id$.
Note that if $A(t) = A$ constant then $M(t) = \exp(tA)$.

The control-to-state map $\mathcal{M}(u)$

Let us introduce the control-to-state map (model operator) $\mathcal{M}(u)$:

$$\mathcal{M} : \ u(t) \mapsto y^u(t) \equiv y(u(t))$$

with $\mathcal{U} \subset L^\infty(I, \mathbb{R}^m)$ and $\mathcal{Y} \subset C^0(I, \mathbb{R}^n)$.

The following central property holds.

Proposition 9.1. *In the linear case, the control-to-state operator $\mathcal{M}(u(t))$ is affine for all t in $[0, T]$.*

Proof. Let $y(t)$ be the (unique) solution associated to $u(t)$: $y(t) = \mathcal{M}(u(t))$. The result follows straightforwardly from the explicit expression (9.6) of the solution $y(t)$, here in the case $S = 0$.
□

9.3.2 Quadratic cost functions

The choice of the cost function to be minimized is part of the problem definition. Since it will be minimized, convexity properties are expected.
Moreover if using computational gradient-based methods, differentiability properties are expected too. Recall that "quadratic $\Rightarrow$ differentiable and strictly convex".

In the present dynamical system context, the typical objective function reads as:

$$\boxed{J(u;y) = \frac{1}{2}\int_0^T \|y(t)\|^2_W dt + \frac{1}{2}\int_0^T \|u(t)\|^2_U dt + \frac{1}{2}\|y(T)\|^2_Q} \tag{9.7}$$

The cost function $j(u)$ is next defined from the objective function $J(u;y)$ by:

$$\boxed{j(u) = J(u\,; y^u)} \tag{9.8}$$

with y^u the unique solution of the (linear) model, given u.

Note that by definition the objective function $J(u;y)$ is here quadratic in its primal variables $(u;y)$. On the contrary, $j(u)$ is not quadratic in u!
However, it will be demonstrated that since the model operator $\mathcal{M}$ is affine, see Prop. 9.1, $j(u)$ is strictly convex.

In practice, W, U and Q are often diagonal positive matrices whose the diagonal coefficients are tuned to define the relative importance of the terms.

9.3.3 Linear-Quadratic (LQ) optimal control problem

The optimal control problem defined by (9.5)-9.7) reads as follows.
Given y_0 and T, find $u^*(t)$ such that

$$\boxed{u^* = arg\min_u j(u)} \tag{9.9}$$

with $j(u) = J(u\,; y^u)$, $y^u(t)$ the solution of the linear ODE (9.5).

It is a *Linear-Quadratic (LQ) optimal control problem.*

9.4 Numerical methods for optimal control problems

9.4.1 Two classes of numerical methods: direct, indirect

Basically, it exists two classes of numerical methods to solve an optimal control problem (whatever if linear or not): the *direct methods* and the *indirect methods*.

- *Direct methods* simply consist in discretizing the state y and the control u, to reduce the problem to a standard discrete optimization problem.
  Next, the minimization relies on nonlinear programming algorithms such as e.g. the classical Sequential Quadratic Programming (SQP) algorithm.
- *Indirect methods* consist in numerically solving a problem resulting from the so-called *maximum principle*.
  The latter relies on the *necessary first-order optimality* conditions and on the *Hamiltonian*. These concepts are presented in next sections.

In short, pros and cons of each approach are as follows.

- Direct methods are easy to implement. They are tractable in small dimensions only, not in large dimensions.

- Indirect methods require to derive the optimality system relying on the Hamiltonian and the adjoint equations (see next sections). They are efficient in all dimensions, large ones included.

9.4.2 Direct methods

Direct methods consist to:

- write the optimal control problem equations in a discrete form,
- solve the optimization problem by a standard differentiable optimization algorithm e.g. the classical Sequential Linear Quadratic (SQL) algorithm.

$$\text{argmin}_{(u_1,\cdots,u_M)} j(u_1,\cdots,u_M) \tag{9.10}$$

with $u_h = (u_1,\ldots,u_M)$, $j(u_h) = J(u_h; y_h(u_h))$.

For a sake of simplicity, let us consider the basic explicit Euler scheme to solve the equation $y'(t) = A(t)y(t) + B(t)u(t)$.
The discrete system reads:

$$y^{n+1} = (1 + hA(t^n))y^n + hB(t^n)u^n \text{ for } n = 0\cdots(N-1); \quad y^0 = y_0 \tag{9.11}$$

The finite dimensional optimization problem reads:

$$\boxed{\left\{\begin{array}{c} \min_{u_h=(u_1,\cdots,u_M)} j(u_h) \\ \text{under the } N \text{ constraints (9.11) on } y_h = (y^1,\ldots,y^N) \\ (= \text{the numerical scheme equations}) \\ \oplus \text{ potential equality-inequality constraints on } u_h. \end{array}\right.} \tag{9.12}$$

Such (finite dimensional) optimization problem, with (equality-inequality) constraints, can be solved by the Sequential Quadratic Programming (SQP) algorithm.

Recalls on the SQP algorithm Sequential Quadratic Programming (SQP) algorithms denote iterative methods for constrained nonlinear optimization problems. The objective function and the constraints are supposed to be C^2.
The SQP algorithms principle is as follows. One solves a sequence of optimization subproblems; each of them optimizes a *quadratic representation of the objective function*, under the constraints which are linearized.

- If the problem is unconstrained, then the method reduces to the Newton method: the optimum is such that it makes vanish the gradient.

- If the problem has equality constraints and no inequality constraints, then the method is equivalent to apply Newton's method to the first-order optimality conditions (KKT condition).

Pros and cons are as follows.

$\oplus$ The algorithm is available on any well built optimization library or computational system.

$\ominus$
- Each iteration may demand a lot of computational time.
- The algorithm requires cost functions twice differentiable.
- The algorithm may converge to local minima only.

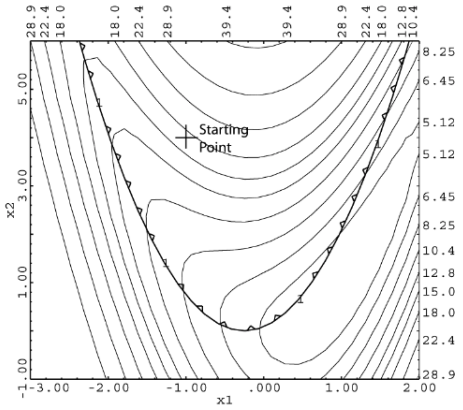


Figure 9.3: The SQP algorithm for an example. Image extracted from [].

9.4.3 Numerical solution of the optimal vehicle dynamic

Exercise 9.2. *1) Detail the equations to solve by a direct method the vehicle dynamic problem described in Section 9.1.*
2) Detail the numerical algorithm implemented into the Python code provided on the Moodle course page. Follow the instructions (supplementary material).

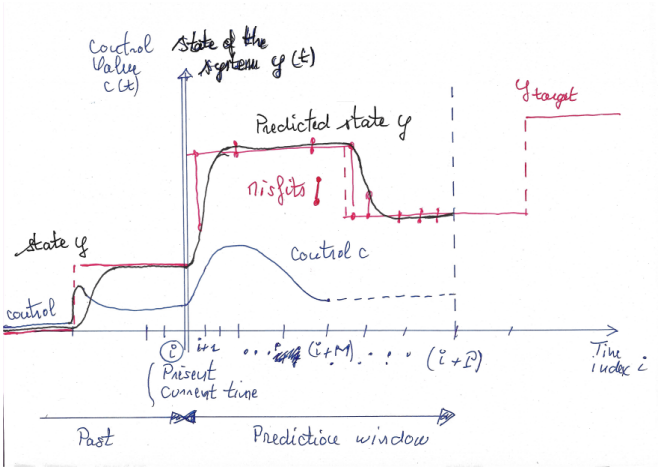


Figure 9.4: The optimal control policy is here simply imposed by using a basic direct method since the problem is of small dimension (a scalar control variable) and the model is very low CPU-time consuming.

9.5 Open-loop control: the Pontryagin principle & Hamiltonian

To address large dimensions systems, typically $y(t) \in \mathbb{R}^n$ with $n = O(10^p)$, $p \approx 4$ and more), indirect methods are more adapted or even required. However, indirect methods require to derive the optimality system which relies on the Hamiltonian and the adjoint equation concepts.

Open-loop control vas closed-loop control. The Pontryagin's principle states optimality condition useful for open-loop control only, Fig. 9.5.
In automatic, the expected information is a closed-loop control law. Such feedback laws are not required in the context of Data Assimilation.

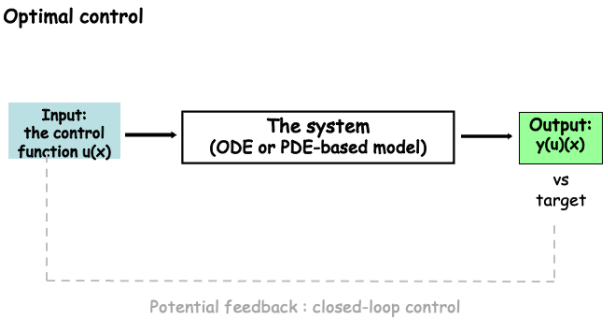


Figure 9.5: Optimal control of a system: open loop control vs closed loop control.

9.5.1 Existence and uniqueness of the solution in the LQ case *

Theorem 9.3. *Let $j(u)$ be defined by (9.7). It exists a unique $u \in M$ minimizing $j(u)$ with the "constraint" $y(t)$ solution of (9.5).*
In other words, it exists a unique optimal control $u(t)$ and a corresponding trajectory $y(t)$ to the LQ problem.

Proof A) Existence.
It is based on the convergence of minimizing sequence (calcul of variations, D. Hilbert, 1900 approx.).

B) Uniqueness.
Recall that, see (9.7)(9.9):

$$j(u) = \frac{1}{2}\int_0^T \|y^u(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u(t)\|_U^2 dt + \frac{1}{2}\|y^u(T)\|_Q^2$$

We prove that $j(u)$ is strictly convex that is $\forall (u_1, u_2) \in M^2,\ \forall t \in]0,1[$,

$$j(tu_1 + (1-t)u_2) < tj(u_1) + (1-t)j(u_2)$$

unless $u_1 = u_2$.

For all t, $\|u(t)\|_U$ is a norm hence convex but not strictly convex... Proof of this assertion: the triangle inequality.
However, in a Hilbert space, the square of a norm (eg $\|u(t)\|_U^2$) is strictly convex, see e.g. [4] Chap. 10, p118.

Moreover, it has been previously proved that the control-to-state operator $\mathcal{M}$ is affine therefore convex, see Prop. 9.1.
In other respects $\|.\|_W$ and $\|.\|_Q$ are semi-norms therefore convex.

Finally the cost function $j(u)$ is strictly convex and the uniqueness follows.
Indeed, let u_1 and u_2 be such that: $j(u_k) = \inf_{v \in M} j(v),\ k = 1, 2$.
We have: $j(tu_1 + (1-t)u_2) < tj(u_1) + (1-t)j(u_2)$.

Hence: $j(tu_1 + (1-t)u_2) < \inf_{v \in M} j(v)$ unless $u_1 = u_2$, which must be the case. $\square$

Note that the strict convexity of $j(\cdot)$ is due to the quadratic term $\|u(t)\|_U^2$.

9.5.2 The Pontryagin minimum principle

In the case of *non-linear* state equation, the cost function is non-convex and the Pontryagin minimum principle (also called maximum principle) states a *necessary condition* of optimality.

In the LQ case, the Pontryagin minimum principle is a *necessary and sufficient condition* of optimality. The Pontryagin minimum principle relies on the concepts of Hamiltonian[2] and *adjoint equation*.

Let us recall the equations in the LQ case, see (9.5)(9.7). The model reads:

$$\begin{cases} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) + S(t) \text{ for } t \in I = [0,T] \end{cases} \tag{9.13}$$

$$J(u;y) = \frac{1}{2}\int_0^T \|y(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u(t)\|_U^2 dt + g(y(T)) \tag{9.14}$$

Note that we consider here the term in $y(T)$ slightly more general than before with $g(\cdot)$ any function defined from $\mathbb{R}^n$ into $\mathbb{R}$, C^1 and convex.

The goal is to characterize the optimal control satisfying: $u^* = arg\min_u j(u)$ with $j(u) = J(u;y^u)$.

[2]The so-called "Hamiltonian" in control theory is a particular case of the Hamiltonian in mechanics; it is inspired by the Lagrangian you have studied during your optimization course.

Theorem 9.4. *The trajectory $y(t)$ associated with the control $u(t)$, is optimal for the LQ problem (9.13)(9.14) if there exists an adjoint field $p(t)$ which satisfies:*

$$-p'(t) = p(t)A(t) - y^T(t)W \text{ for almost } t \in [0,T] \tag{9.15}$$

with the Final Condition: $p(T) = -\nabla g^T(y(T))$.
By convention, $p(t)$ is here a line vector, on the contrary to $y(t)$.
Furthermore, the optimal control u satisfies:

$$Uu(t) = B^T(t)\ p^T(t) \text{ for almost } t \in [0,T] \tag{9.16}$$

Remark 9.5.

- *Note that Eq. (9.15) written in $q \equiv p^T$, q a column vector as y, reads:*

  $$-q'(t) = A^T(t)q(t) - Wy(t)$$

  which better highlights the terminology "adjoint equation".

- *This theorem provides an expression of the optimal control u^* not explicitly depending on the state y but in function of an auxiliary field p called the adjoint field.*
  The adjoint field $p(t)$ is solution of a linear equation "similar" to the model one with $-A^T$ instead of A in particular. it is therefore reverse in time, therefore starting from a condition at $t = T$. p depends on y through the adjoint equation.

- *An explicit expression of the optimal control u^* in function of y would provide a feedback law which is required for closed-loop control. Such law is derived in the LQ case in a subsequent section.*

Proof of the theorem. The proof of the theorem is based on *calculus of variations.*
The calculations are similar to those proving the general Theorem 10.7 tackling non-linear stationary PDEs.
It has been proved that the cost function $j(u)$ is strictly convex and the optimal control u^* exists and is unique, see Theorem 9.3.

Let δu be a perturbation to u^*. We denote by u_δ the perturbed optimal control, $u_\delta = (u^* + \delta u)$, and by y_δ the corresponding perturbed solution.

*) y_δ starts from the same I.C. y_0 as y, that is: $\delta u(0) = 0 = \delta y(0)$.

*) We denote by δy the difference $(y_\delta - y^u)$: $y_\delta = y^u + \delta y$.
The perturbed solution satisfies: $y'_\delta(t) = A(t)y_\delta(t) + B(t)u_\delta(t) + S(t)$.
By linearity of the equation, we obtain: $\delta y'(t) = A(t)\delta y(t) + B(t)\delta u(t)$.
Therefore the expression of the solution perturbation:

$$\delta y(t) = M(t) \int_0^t M(s)^{-1} B(s) \delta u(s) ds \tag{9.17}$$

with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t), \;\; M(0) = Id.$
Recall that if $A(t) = A$ constant then $M(t) = \exp(tA)$.

*) Recall that: $j(u_\delta) = \frac{1}{2} \int_0^T \|y_\delta(t)\|^2_W dt + \frac{1}{2} \int_0^T \|u_\delta(t)\|^2_U dt + g(y_\delta(T))$.

The optimal control u^* is uniquely determined by the Euler condition $\nabla j(u) = 0$, with:

$$\nabla j(u) \cdot \delta u = \int_0^T < Wy(t), \delta y(t) > dt + \int_0^T < Uu(t), \delta u(t) > dt + \nabla g(y(T))\, \delta y(T) \tag{9.18}$$

(Recall that W and U are symmetric).
The relation $\nabla j(u) \cdot \delta u = 0$ provides the following relation of u in function of y

and δy: for all $\delta u(t)$,

$$\int_0^T < \underbrace{Uu(t)}_{\text{sough term}} , \delta u(t) > dt = - \underbrace{\int_0^T < W\delta y(t), y(t) > dt}_{\text{term in } \delta y(t)...} - \underbrace{\nabla g(y(T)), \delta y(T)}_{\text{final time term}} \quad (9.19)$$

*) Let us inject the expression of $\delta y(t)$ in function of $\delta u(t)$, see (9.17), in the term in $\delta y(t)$ above, see (9.19). We obtain:

$$\underbrace{\int_0^T < W\delta y(t), y(t) > dt}_{\text{term in } \delta y(t)...} = \int_0^T < WM(t) \int_0^t M(s)^{-1} B(s)\delta u(s)ds, y(t) > dt \quad (9.20)$$

$$= \int_0^T < WM(r) \int_0^T M(s)^{-1} B(s)\delta u(s)ds, y(r) > dr$$

$$- \int_0^T \int_0^t y^T(s) WM(s)ds \; M(t)^{-1} B(t)\delta u(t)dt \quad (9.21)$$

after integration by parts.

*) By construction, the expression of the adjoint $p(t)$, solution of (9.15) with the F.C. $p^T(T) = -\nabla g(y(T))$, reads:

$$p(t) = \Lambda(T)M^{-1}(t) + \left(\int_0^t y^T(s) WM(s)ds \right) M^{-1}(t)$$

with $\Lambda(T) = -\nabla^T g(y(T))M(T) - \int_0^T WM(s)y(s)ds$,
with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t), \;\; M(0) = Id.$
Recall that if $A(t) = A$ constant then $M(t) = \exp(tA)$.
Note that in the expression of $p(t)$ above, the F.C. $p^T(T) = -\nabla g(y(T))$ is of course retrieved.

By injecting this expression of $p(t)$ (and $\Lambda(t)$) in (9.21), we get:

$$\underbrace{\int_0^T < W\delta y(t), y(t) > dt}_{\text{term in } \delta y(t)...} = etc \quad (9.22)$$

*) Let us address now on the final time term in (9.19).
By construction of the F.C. of the adjoint field and using the expression of $\delta y(t)$ in function of $\delta u(t)$, see (9.17), at time $t = T$, we get:

$$\underbrace{\nabla g(y(T)), \delta y(T) >}_{\text{final time term}} = etc$$

ToDo: terminer les calculs...

*) By combining the expressions above, we finally obtain the relation: for all $\delta u(t)$,

$$\int_0^T < \underbrace{Uu(t)}_{\text{sough term}} , \delta u(t) > dt = \int_0^T < B^T(t)\ p^T(t), \delta u(t) > dt \tag{9.23}$$

Therefore the expression:

$$u(t) = U^{-1}(t) B^T(t)\ p^T(t)$$

□

9.5.3 The Hamiltonian

Let us consider the linear direct model without source term for sake of simplicity ($S = 0$):

$$\boxed{\begin{cases} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) \text{ for } t \in (0,T) \\ \text{with the Initial Condition: } y(0) = y_0. \end{cases}} \tag{9.24}$$

$$\boxed{j(u(t)) = \frac{1}{2}\int_0^T \left(\|y(t)\|_W^2 + \|u(t)\|_U^2\right) dt + g(y(T))} \tag{9.25}$$

In this optimal control context, the *Hamiltonian* H is the functional defined as:

$$\boxed{H(y,p,u)(t) = (p^T, (Ay + Bu))(t) - \frac{1}{2}(\|y(t)\|_W^2 + \|u(t)\|_U^2)} \tag{9.26}$$

with $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$.

Let us calculate the partial derivatives of $H(y,p,u)$. We have:

$$\begin{cases} \partial_y H(y,p,u)(t) \cdot \delta y(t) = (p^T, A\delta y)(t) - (Wy, \delta y)(t) \\ \partial_p H(y,p,u)(t) \cdot \delta p(t) = ((Ay + Bu), \delta p^T)(t) \\ \partial_u H(y,p,u) \cdot \delta u(t) = (p^T, B\delta u)(t) - (Uu, \delta u)(t), \end{cases} \tag{9.27}$$

where $(\cdot,\cdot)$ denotes the scalar product in the adequate Euclidian spaces.

Consequently, the necessary and sufficient conditions of the LQ problem solution stated in Theorem 9.4 correspond to *stationary conditions of the Hamiltonian* in the following sense. For all t,

$$\boxed{\left\{\begin{array}{rcll} y'(t) & = & \partial_p H(y,p,u)(t) & = A(t)y(t) + B(t)u(t) \\ -p'(t) & = & \partial_y H(y,p,u)(t) & = p(t)A(t) - y^T(t)W \\ 0 & = & \partial_u H(y,p,u)(t) & \Leftrightarrow Uu(t) = B^T(t)p^T(t) \end{array}\right.} \tag{9.28}$$

with the Final Condition (F.C.): $\boxed{p(T) = -\nabla g^T(y(T))}$.
(Recall that p is a line vector in $\mathbb{R}^n$).

These three relations constitute the so-called *optimality system.*
The first two equations are the state equation and the adjoint state equations respectively (with the F.C. depending on $y(T)$): they constitute the so-called *Hamiltonian equations.*
The Hamiltonian equations are accompanied by the last equation which is the optimality condition on u, a necessary and sufficient condition in the present LQ case.
In general, these three equations are fully coupled.

The Hamiltonian: the conserved quantity in time

Exercise 9.6. *Let (y^*, p^*, u^*) be the solution of the LQ problem (9.24). Show that the mapping $t \mapsto H(y^*, p^*, u^*)(t)$ is constant.*

In some contexts, the Hamiltonian denotes the energy of the system.

Remark 9.7. • *The solution of the LQ problem exists and is unique, see Theorem 9.3. Moreover, the stationary conditions of the Hamiltonian $H(y, p, u)$ defined by (9.26) corresponds to the necessary and sufficient conditions of the LQ problem solution which is unique, see Theorem 9.4 and Eq. (10.33).*

If the model is non linear or if the objective function $J(u; y)$ is not quadratic then Theorem 9.4 does not hold anymore.
However, in this case, the stationary conditions of the Hamiltonian $H(y, p, u)$ corresponds to necessary conditions of solution(s). In this case, the corresponding triplet(s) (y, p, u) represent optimal control policies for the dynamical system.

- *Some links can be done between the Hamiltonian and the Lagrangian. For the control of PDEs (see next chapter), to obtain the adjoint equation, a Lagrangian will be introduced.*

9.6 The fundamental equations at a glance

The Linear model

The linear model (without source term $s(t)$) reads:

$$\begin{cases} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) \text{ for } t \in I = [0,T] \\ \text{with the initial condition: } y(0) = y_0. \end{cases} \tag{9.29}$$

The model operator (control-to-state map) $\mathcal{M}(u)$ is defined as: $\mathcal{M}(u) = y^u$. This operator $\mathcal{M}(u(t))$ is here affine in $u(t)$, for all t in $[0,T]$.

The cost functional $j(u)$ is defined from quadratic terms. First, the observation function is defined:

$$j(u(t)) = J(u(t), y^u(t)) = \frac{1}{2}\int_0^T \|y^u(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u(t)\|_U^2 dt + \frac{1}{2}\|y^u(T)\|_Q^2 \tag{9.30}$$

Given the observation" functional $J(u;y)$, the cost function is defined as:

$$j(u) = J(u\,; y^u) \tag{9.31}$$

with $y^u(t)$ the unique solution of (9.29), given u.

The LQ optimal control problem

Given the I.C. y_0 and the final time T, find $u^*(t)$ such that:

$$j(u^*) = \min_u j(u) \tag{9.32}$$

The Hamiltonian is the conserved quantity in time. Its expressions is:

$$H(y,p,u) = p(Ay + Bu) - \frac{1}{2}(\|y\|_W^2 + \|u\|_U^2) \tag{9.33}$$

with $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $u(t)$ the control, $y(t)$ the state of the system and $p(t)$ the adjoint state, solution of the adjoint model.

The adjoint model reads:

$$-p'(t) = p(t)A(t) - y(t)^T W(t) \text{ for } t \in [T,0] \tag{9.34}$$

with the final condition: $p^T(T) = -Qy(T)$. (p is a line vector).

The Pontryagin maximum principle states that if the control $u(t)$ is defined as:

$$u(t) = U(t)^{-1}B(t)^T p(t)^T \text{ for almost } t \in [0,T] \tag{9.35}$$

then the state $y^u(t)$ associated to this control $u(t)$, is optimal for the LQ problem.

The Pontryagin maximum principle can be read as follows: the equations below have to be satisfied.

$$\left\{ \begin{array}{rcll} y'(t) & = & \partial_p H(y,p,u) & = Ay(t) + Bu(t) \\ -p'(t) & = & \partial_y H(y,p,u) & = p(t)A - y^T(t)W \\ 0 & = & \partial_u H(y,p,u) & \Longleftrightarrow Uu(t) = B^T p^T(t) \end{array} \right. \tag{9.36}$$

with the final condition: $p(T) = -\nabla^T g(y(T)) = -y(T)^T Q$. This is the **optimality system**.

Chapter 10

Optimal Control of Stationary PDEs: Adjoint Method, VDA

The outline of this chapter is as follows[1].

Contents

[1]Recall that the sections indicated with a * are "to go further sections". They can be skipped in a first reading or if the reader is not particularly interested in deeper mathematical basis, mathematical proofs.

10.1 General non-linear case in infinite dimension

10.1.1 The direct model

$$\boxed{\begin{cases} \text{Given } u \in U, \text{ find } y \in V \text{ such that:} \\ A(u;y) = F(u) \text{ in } \Omega \\ \text{with boundary conditions on } \partial\Omega \end{cases}} \tag{10.1}$$

where A is an elliptic operator (with respect to the state y), defined from $U \times V$ into V' (dual of V).
$A(\cdot;\cdot)$ is a-priori non-linear, both with respect to the parameter u and with respect to the state y.
F is defined from U into V'.

Assumption 10.1. *The state equation (10.1) is well posed in the Hadamard sense: it has an unique solution $y \in V$, moreover this solution is continuous with respect to the parameters (in particular with respect to u).*

Optimal control terminology: distributed control, boundary control

- If the control u appears in the "bulk" (i.e. in Ω) then one says that it is a *distributed control*.
- If u appears on the boundary conditions only, then one says that it is a *boundary control*.

10.1.2 Examples

Example 1)

$$A(u;y) = -div(\lambda \nabla y) \text{ and } F(u) = u$$

with mixed boundary conditions: $y = 0$ on Γ_0 ; $-\lambda \partial_n y = \varphi$ on $\partial\Omega/\Gamma_0$, with φ given.
$\lambda \in L^\infty(\Omega)$, $\lambda > 0$ a.e.
The control u is spatially distributed; it constitutes the source term (the RHS). Here, u is a distributed control since defined in Ω; it represents an external force (since in the RHS).

Exercice 10.2.
a) Write the state equation (and recall the adequate functional spaces). Prove that it has one and only one (weak) solution in V.
b) Prove that the unique solution y is continuous with respect to u i.e. the operator $\pi : u \in U \mapsto y^u \in V$ is continuous.

Example 2)

$$A(u;y) = -div(\lambda(y;u)\nabla y)$$

with mixed boundary conditions (independent of u).
In this still classical case, the PDE is non linear due to the term $\lambda(y;\cdot)$.

Example 3)

$A(u;y) = -div(\lambda \nabla y)$ and $F(u) = f$. A and F are here independent on the control u, however the boundary conditions are, as:

$$y = 0 \text{ on } \Gamma_0 \text{ and } -\lambda \partial_n y = u \text{ on } \partial\Omega/\Gamma_0$$

(λ, f) are given in adequate functional spaces.
In this case u is a boundary control, it represents the flux at boundary.

10.1.3 The objective and cost function terms (misfit to data)

If enriched by a regularization term, the objective function reads as:

$$\boxed{J(u\,;y) \;=\; J_{obs}(y) \;+\alpha_{reg}\; J_{reg}(u)} \tag{10.2}$$

with J_{reg} the regularization term.
Note that $J : U \times V \to \mathbb{R}$ with $J_{obs} : V \to \mathbb{R}$ and $J_{reg} : U \to \mathbb{R}$.

The cost function $j(u)$ is finally defined as:

$$\boxed{\begin{array}{c} j(u) = J(u\,;y(u)) \\ \text{where } y(u) \text{ (also denoted } y^u\text{) is the (unique) solution of the direct model (10.1).} \end{array}} \tag{10.3}$$

We have: $j : U \to \mathbb{R}$.

LQ problems vs real-world problems Most of the control problems are not LQ problems for one of at least one of the following reason:

- The model is not linear.
- The regularization term $\|J_{reg}(u)\|$ is higher order only (e.g. of the form $\|\nabla u\|^2$) therefore not strictly convex in u (but in $\|\nabla u\|$ only).
- Even if the direct model is linear then we generally do not have measurements everywhere in Ω, that is at each numerical grid points/nodes!

the actual misfit term may have one of the following form:

$$\int_{\omega}(Z(y)-z^{obs})^2 \;\; dx \text{ or } \sum_{m=1}^{M}(Z(y)(x_m)-z^{obs}(x_m))^2 \text{ or } \int_{\Omega}(Z(y)-\mathcal{F}(z^{obs}))^2 \;\; dx$$

where ω is a non-convex subset of the complete domain Ω, M is the total number of point-wise data and $\mathcal{F}$ denotes e.g. an uncertain low-pass band filter.

In one of these cases, the cost function is not strictly convex anymore and the uniqueness of a minimum u^* is not guaranteed anymore.

The minimization problem is often ill-conditioned: in the vicinity of a minimum (potentially local only), the cost function presents "nearly flat valleys", Fig. 10.1.

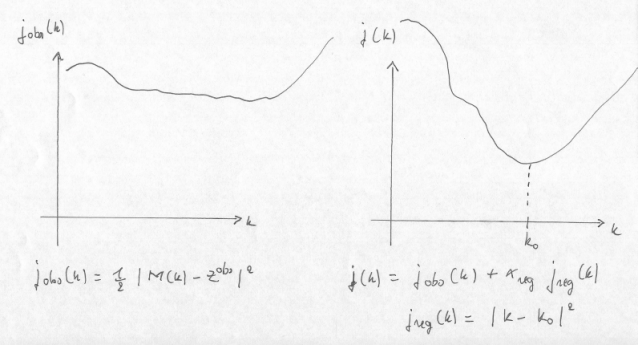


Figure 10.1: Tikhonov regularization. (L) A typical "poorly" convex functional $j_{misfit}(\cdot)$: ill-conditioned minimisation problem. (R) Regularized functional $j(.) = (j_{misfit}(.) + \alpha_{reg} j_{reg}(.))$, with j_{reg} is here strictly convex in u and defined from a *prior* value u_0.

10.1.4 Optimal control problem, VDA problem

Variational Data Assimilation (VDA) simply relies on the optimal control of the model (here a PDE) with the cost function $j(u)$ measuring the discrepancy between data and model outputs.

As previously, the control-to-state mapping (the "model operator") reads as: $\mathcal{M} : u \in U \mapsto y^u \in V$ with y^u the (unique) solution of the direct model (10.1).

The optimal control problem reads:

$$\boxed{\begin{cases} \text{Find } u^* \in U_{ad} \text{ such that:} \\ j(u^*) = \min\limits_{U_{ad}} j(u) \\ \text{with the cost function } j \text{ defined by (10.3) .} \end{cases}} \tag{10.4}$$

Problem (10.4) can be re-read as:

$$\boxed{\begin{cases} \text{Minimize } j(u) = J(u; y^u) \text{ in } U_{ad} \\ \text{under the "model constraint" (10.1)} \end{cases}} \tag{10.5}$$

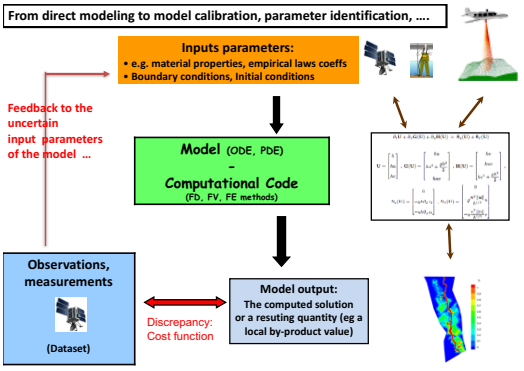


Figure 10.2: Principle of VDA: control some uncertain input model parameters u to make fit the model output y with data.

10.1.5 On the numerical resolution in the general context

Let us recall the following points. The VDA problem consists to solve the optimization problem (10.5).
If the dimension of the (discrete) parameter u is large, moreover if the computation of the cost function $j(u)$ is CPU-time consuming (this the case e.g. if considering a 3d PDE model), then this optimization problem cannot be solved by a global optimization algorithm such as e.g. MCMC / Monte-Carlo type method. In this case, it has to be solved by descent algorithm aiming at computing a local minimum only. Descent algorithms require the information of the gradient $\nabla j(u)$, see some details in Appendix.

The computation of the gradient $\nabla j(u)$ in the large dimensional case (u is of large dimension), is tricky.
Different ways to compute the gradient $\nabla j(u)$ are discussed in next sections, including those by introducing the adjoint model.

10.2 Back to mathematical foundations

10.2.1 Differential calculus in infinite dimensions

Please consult the Appendices and the supplementary material.

10.2.2 Weak forms and dual space representation

Here, considering the weak form of the equations enables to naturally and rigorously derive the adjoint equations including in the presence of non trivial boundary conditions (which is often the case in real-world problems).

$$a(u;y,z) : U \times V \times V \to \mathbb{R} \; ; \quad a(u;y,z) =< A(u;y), z >_{V' \times V}$$
$$b(u;z) : U \times V \to \mathbb{R} \; ; \quad b(u;z) =< F(u), z >_{V' \times V}$$

$$\boxed{\begin{cases} \text{Given } u \in U, \text{ find } y \in V \text{ such that:} \\ a(u;y,z) = b(u;z) \text{ for all } z \in V \end{cases}} \tag{10.6}$$

By using the Riez-Frechet representation theorem (see Section **??** in Appendix), (10.6) is equivalent to:

$$\boxed{A(u;y) = F(u) \text{ in } V'} \tag{10.7}$$

Exercice 10.3. *Apply this general presentation to the toy BVP.*

10.2.3 Differential $j'(u)$ vs gradient $\nabla j(u)$

Recall that $j : U \to \mathbb{R}$, then $j'(u) \in \mathcal{L}(U; \mathbb{R})$.
Let us denote U_h the discrete control space with $dim(U_h) = m$ (there is m discrete control variables). We assume that $U_h \subset U$.

The gradient $\nabla j(u)$ to be computed, $\nabla j(u) \in \mathbb{R}^m$, is related to the differential $j'(u)$ by the relation:

$$\boxed{< \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u \text{ for all } \delta u \in U_h \subset \mathbb{R}^m} \tag{10.8}$$

Of course, the functional $j(\cdot)$ is here assumed to be differentiable. In the sequel sufficient conditions are presented to have $j(\cdot)$ continuously differentiable that is of class C^1.

10.3 Equations derivation from the Lagrangian

10.3.1 The Lagrangian

All calculations below are formal: we do not pay attention to functionals spaces. The equations may be read as being discrete systems too.

The optimization problem (10.5) may be viewed as a (differentiable) optimization problem with the model (10.1) being an equality constraint. Then, it is natural to write the corresponding Lagrangian $\mathcal{L}$:

$$\boxed{\mathcal{L}(u; y, p) = J(u; y) - < A(u; y) - F(u), p >} \tag{10.9}$$

with p the Lagrangian multiplier.

- If considering that $A(u; y)$ and $F(u)$ denote the PDE terms in finite dimension, e.g. $A(u; y)$ the rigidity matrix in FEM and $F(u)$ the RHS vector, then $< \cdot, \cdot >$ simply denotes the Euclidian scalar product.

- If considering that $A(u; y)$ and $F(u)$ represent the operators of the PDE, e.g. $A(u; y) = -div(u \nabla y)$, then $< \cdot, \cdot >$ denotes the dual product $< \cdot, \cdot >_{V' \times V}$ with V the state y belongs to, V a Hilbert space.
  In this case, p is a dual variable belonging to V too.

No constraint are here imposed to the control u (neither equality nor inequality ones).

10.3.2 The optimality system

The stationary point(s) of the Lagrangian provide the necessary optimality condition. These points are determined by the relation: $\nabla\mathcal{L}(u;y,p)=0$. This reads:

$$\boxed{\left\{\begin{array}{ll}\partial_u\mathcal{L}(u;y,p)\cdot\delta u=0 & \forall\delta u\\ \partial_y\mathcal{L}(u;y,p)\cdot\delta y=0 & \forall\delta y\\ \partial_p\mathcal{L}(u;y,p)\cdot\delta p=0 & \forall\delta p\end{array}\right.} \tag{10.10}$$

The last equation of (10.10) provides the direct model: $A(u;y)=F(u)$.

The second equation of (10.10) provides the following linearized equation:

$$\partial_y J(u;y)\cdot\delta y- <\partial_y A(u;y)\cdot\delta y,p>=0 \quad \forall\delta y$$

Therefore: $<\partial_y J(u;y)-[\partial_y A(u;y)]^*\cdot p,\delta y>=0 \ \ \forall\delta y$.
Therefore:

$$[\partial_y A(u;y)]^*\cdot p=\partial_y J(u,y)$$

This is the so-called adjoint equation.

The first equation of (10.10) reads: $\partial_u J(u;y)\cdot\delta u- <(\partial_u A(u;y)-F'(u))\cdot\delta u,p>=0\ \forall\delta u$.

It will be shown later that this equation is the necessary condition which reads: "the gradient equals 0".

Using the particular decomposition of $J(u;y)$ introduced in (10.2), we have:

$$\partial_y J(u,y)\cdot\delta y=J'_{obs}(y)\cdot\delta y \text{ and } \partial_u J(u;y)\cdot\delta u=J'_{reg}(u)\cdot\delta u \tag{10.11}$$

In summary, we have the set of equations:

$$\boxed{\left\{\begin{array}{lrcll}\text{Given } u\text{, find } y \text{ s.t. } : & A(u;y) & = & F(u) & \text{(Direct model)}\\ \text{Given } (u,y)\text{, find } p \text{ s.t. } : & [\partial_y A(u;y)]^*\cdot p & = & J'_{obs}(y) & \text{(Adjoint model)}\\ \text{Given } (y,p)\text{, find } u \text{ s.t. } : & [\partial_u A(u;y)-F'(u)]^*\cdot p & = & J'_{reg}(u) & \text{(1st order condition)}\end{array}\right.} \tag{10.12}$$

This set of equations constitutes the so-called optimality system.

10.3.3 Using weak forms

Let V be the state space ($y \in V$), V a Hilbert space. Let $< \cdot, \cdot >_{V' \times V}$ be the dual product. Then, the Lagrangian reads:

$$\begin{array}{rcl} \mathcal{L}(u; y, p) & = & J(u; y) - [a(u; y, p) - b(u; p)] \\ \mathcal{L} & : & U \times V \times V \to \mathbb{R} \end{array}$$

Next, the last equation of (10.10) provides the state equation (the direct model in weak form):

$$a(u; y, \delta p) \;=\; b(u; \delta p) \quad \forall \delta p \in V$$

The second equation of (10.10) provides the following linearized equation:

$$\partial_y a(u; y, p).\delta y \;=\; \partial_y J(u, y).\delta y \quad \forall \delta y \in V$$

It is the adjoint equation.

The first equation of (10.10) reads:

$$\partial_u J(u; y) \cdot \delta u - [\partial_u a(u; y, p) - \partial_u b(u; p)] \cdot \delta u = 0 \;\; \forall \delta u \in U$$

It will be shown that: $j'(u) \cdot \delta u = \partial_u J(u; y) \cdot \delta u - [\partial_u a(u; y, p) - \partial_u b(u; p)] \cdot \delta u$. Therefore this last equation reads: $j'(u) = 0$.

These three equations summarizes as:

$$\left\{ \begin{array}{rcll} a(u; y, z) & = & b(u; z) & \forall z \in V \\ \partial_y a(u; y, p) \cdot z & = & \partial_y J(u, y) \cdot z & \forall z \in V \\ \partial_u J(u; y) \cdot \delta u - [\partial_u a(u; y, p) - \partial_u b(u; p)] \cdot \delta u & = & 0 & \forall \delta u \in U \end{array} \right. \tag{10.13}$$

Exercice 10.4. *Apply the general expressions above to the equations of the programming practical.*

10.4 Mathematical purposes *

This is a "to go further section".

10.4.1 Differentiability of the cost function

the following question is of main interest in order to address the optimal control problem:

Is the cost function (continuously) differentiable ?

A useful result to address this question of differentiability is the *implicit function theorem*.

10.4.2 Existence and uniqueness of the optimal control in the LQ case

Warm up with a basic linear finite dimensional problem

Let us consider a problem such that M, the control-to-state map, is defined from $\mathbb{R}^m$ onto $\mathbb{R}^n$ as $M : u \mapsto y^u$, with y^u the unique solution of the state equation given u.
Moreover, *we assume that M is linear.*

An example is as follows. The control appears in the RHS of the (finite dimensional) state equation as:

$$\boxed{Ay = Fu \text{ in } \mathbb{R}^n} \tag{10.14}$$

with $A \in \mathcal{M}_{n\times n}$ a non singular real matrix, F a rectangular matrix, $F \in \mathcal{M}_{n\times m},\ m < n$, of maximal rank m.

We consider the usual quadratic observation function $J(u; y) = \|Zy - z^{obs}\|_2^2$, with the observation operator Z a non-singular linear matrix Z of $\mathcal{M}_{n\times n}$.
For a sake of simplicity, we set here $z^{obs} = 0$.

The cost function is defined as usual as: $j(u) = J(u; y^u)$.
The optimal control problem aims at solving $\boxed{u^* = arg \min_{u\in R^m} j(u)}$.

Exercise 10.5. *Show that this optimal control problem admits an unique solution u^*, even without regularization term in $J(u; y)$.*

10.5 Gradient computation: methods for small dimension cases

10.5.1 Recall: why and how to compute the cost function gradient?

Problem statement

In the discrete context, the dimension of the gradient $\nabla j(u)$ equals m, m the dimension of the discrete control variable.
Descent algorithms require scalar products of the form $< \nabla j(u), \delta u >$ for at least m directions δu.

Composite control variable case In the case the control variables includes different natures of components e.g. $u = (u_1, u_2)$ then we have:

$$\nabla j(u) = (\frac{\partial j}{\partial u_1}(u), \frac{\partial j}{\partial u_2}(u))^T$$

$$j'(u) \cdot \delta u = \frac{\partial j}{\partial u_1}(u) \cdot \delta u_1 + \frac{\partial j}{\partial u_2}(u) \cdot \delta u_2$$

Small dimensional vs large dimensional case In practice we will have to distinguish small dimensional cases ($m = O(1)$) to large dimensional cases ($m = O(10^2)$ and much more).
The challenging case will be the large dimensional one of course. That is a key question will be:

> *How to compute the (scalar) values $< \nabla j(u), \delta u >$ for a large number of directions δu i.e. with m large ?*

In the next paragraphs, we first present methods to compute $< \nabla j(u), \delta u >$ which are tractable for m very small only, i.e. for small dimensional inverse problems only.

10.5.2 Computing the gradient without adjoint model

Two natural options arise to compute the differential $j'(u)$, therefore the gradient $\nabla j(u)$.

Option 1: the Finite Difference gradient

$$j'(u) \cdot \delta u \approx \pm \frac{j(u \pm \varepsilon \delta u) - j(u)}{\varepsilon} \text{ at order 1 in } \varepsilon \tag{10.15}$$

$$j'(u) \cdot \delta u \approx \frac{j(u + \varepsilon \delta u) - j(u - \varepsilon \delta u)}{\varepsilon} \text{ at order 2 in } \varepsilon \tag{10.16}$$

Advantages and drawbacks of the FD approach.
$\oplus$: simple to implement, non-intrusive.
$\ominus$: requires $(m + 1)$ evaluations of $j(u)$ therefore $(m + 1)$ resolutions of the direct model. This is generally not possible for m large.
$\ominus$: The accuracy depends on the choice of ε (and an optimal value of ε is a-priori unknown).

Option 2: expression of $j'(u)$ based on the Tangent Linear Model (TLM)

The straightforward expression of $j'(u)$ (differential calculations) Let u_0 in U, for all $\delta u \in U$,

$$\boxed{j'(u_0)\cdot\delta u \;=\; \frac{\partial J}{\partial u}(u_0;y(u_0))\cdot\delta u + \frac{\partial J}{\partial y}(u_0;y(u_0))\cdot w^{\delta u}} \tag{10.17}$$

where $w^{\delta u}$ denotes the derivative of the state y with respect to u in the direction δu:

$$w^{\delta u} = \frac{dy}{du}(u_0)\cdot\delta u \tag{10.18}$$

In the case that $J(u;y)$ has the particular form (10.2), we have: $\partial_y J(u_0;y) = J'_{obs}(y)$ and $\partial_u J(u_0;y) = \alpha_{reg}J'_{reg}(u_0)$. Therefore:

$$\boxed{j'(u_0)\cdot\delta u \;=\; J'_{obs}(y(u_0))\cdot w^{\delta u} + \alpha_{reg}J'_{reg}(u_0)\cdot\delta u} \tag{10.19}$$

The TLM The TLM consists to differentiate the state equation with respect to the control variable u. By simple differentiation, we obtain:

$$\frac{\partial A}{\partial u}(u;y^u)\cdot\delta u + \frac{\partial A}{\partial y}(u;y^u)\cdot(\frac{dy}{du}(u)\cdot\delta u) = F'(u)\cdot\delta u \tag{10.20}$$

Therefore the TLM:

$$\boxed{\left\{\begin{array}{l}\text{Given } u_0\in U \text{ and } y^{u_0} \text{ the corresponding solution of the state equation (10.1),}\\ \text{ given a direction } \delta u\in U, \text{ find } w^{\delta u}\in V \text{ such that:}\\ \dfrac{\partial A}{\partial y}(u_0;y^{u_0}).w^{\delta u} = \left[F'(u_0) - \dfrac{\partial A}{\partial u}(u_0;y^{u_0})\right]\cdot\delta u \text{ in } \Omega\\ \text{with corresponding linearized boundary conditions on } \partial\Omega\end{array}\right.} \tag{10.21}$$

Remark 10.6.

- *For each new value of δu, only the RHS of the TLM changes.*
  As a consequence if the numerical solver relies on a factorization of the LHS, the latter can be done once for all.

- *In the case of a linear model, that is the map* $y \mapsto A(\cdot; y)$ *is linear, the TLM simplifies as:*

$$A(u_0; w^{\delta u}) = \left[F'(u_0) - \frac{\partial A}{\partial u}(u_0; y^{u_0}) \right] \cdot \delta u \text{ in } \Omega$$

In this case, the differential operator therefore the numerical solver, are the same to the direct model solver. Only the RHS changes compared to the direct model.

The *weak form* of the TLM is as follows:

$$\left\{\begin{array}{l} \text{Given } u \in U \text{ and } y^{u_0} \text{ solution of (10.6)},\\ \text{ given } \delta u \in U, \text{ find } w \in V \text{ such that:}\\ \dfrac{\partial a}{\partial y}(u; y^{u_0}, z).w = [\dfrac{\partial b}{\partial u}(u; z) - \dfrac{\partial a}{\partial u}(u; y^{u_0}, z)].\delta u \text{ for all } z \in V \end{array}\right. \tag{10.22}$$

Solving the TLM provides $w^{\delta u} = \dfrac{dy}{du}(u).\delta u$,
that is the derivative of the state y with respect to the control u in the direction δu.

Recall that: $\mathcal{M} : u \in U \mapsto y^{u_0} \in V$. Therefore $\dfrac{dy}{du}(u) \in \mathcal{L}(U; V)$ and $w^{\delta u} \in V$.

Note that of course if the direct model is linear then we simply have: $\frac{\partial A}{\partial y}(u_0; y^{u_0}).w = A(u_0; w)$.

Advantages and drawbacks of the TLM-based expression.
$\ominus$: If the direct model is non-linear, the TLM has to be implemented (intrusive approach).
Note that if the non-linear model is solved by the Newton-Raphson method (or if the direct model is linear), then the RHS only has to be coded.
The TLM has to be solved m times to obtain $w^{\delta u}$ in each direction δu. Therefore if m is large and the CPU time for each resolution is large than the TLM approach to compute $w^{\delta u}$ is prohibitive.
$\oplus$: The accuracy of $w^{\delta u}$, therefore of the gradient, is fully controlled by the numerical scheme accuracy.
Compared to the FD approach, this does not depend on an arbitrary setting of ε.

In the end, the FD approach and the TLM approach are feasible for small dimension cases only that is for $m = O(1)$. Moreover, if possible, it is preferable to compute the gradient using the TLM compared to the FD.

10.6 Cost gradient computation: the adjoint method

The adjoint equations are a mathematical trick enabling the gradient computation by solving one (1) extra system only. This has to be compared to the $O(m)$ resolutions if using the FD-based approach or the TLM-based approach.

10.6.1 Deriving the gradient expression without the term $w^{\delta u}$

Recall that: $\forall \delta u \in U$,

$$j'(u) \cdot \delta u \;=\; \frac{\partial J}{\partial u}(u; y^u) \cdot \delta u + \frac{\partial J}{\partial y}(u; y^u) \cdot w^{\delta u} \tag{10.23}$$

Recall the TLM:

$$< \frac{\partial A}{\partial y}(u; y^u).w^{\delta u}, z >_{V' \times V} = < \frac{\partial F}{\partial u}(u).\delta u, z >_{V' \times V} - < \frac{\partial A}{\partial u}(u; y^u).\delta u, z >_{V' \times V} \quad \forall z \in V \tag{10.24}$$

with $w^{\delta u}$ defined by (10.18).

Recall the relation for any linear operator L: $< Ly, z >_{V' \times V} = < L^* z, y >_{V' \times V}$.

By adding the two equations above, we obtain: $\forall \delta u \in U$,

$$\begin{aligned} j'(u) \cdot \delta u \;=\; & \frac{\partial J}{\partial u}(u; y^u) \cdot \delta u - < \left(\frac{\partial A}{\partial u}(u; y^u) - \frac{\partial F}{\partial u}(u) \right) \cdot \delta u, z >_{V' \times V} \\ & + < \left(\frac{\partial J}{\partial y}(u; y^u) - (\frac{\partial A}{\partial y})^*(u; y^u) \cdot z \right), w^{\delta u} >_{V' \times V} \quad \forall z \in V \end{aligned} \tag{10.25}$$

where $(\partial_y A)^*$ is the adjoint operator of the linearized direct model operator $\partial_y A$.

The goal is here to make vanish the term in $w^{\delta u}$ in the expression of $j'(u) \cdot \delta u$ above.

Then we define an "adjoint field" p^u such that it satisfies:

$$< \left(\frac{\partial A}{\partial y}(u; y^u) \right)^* \cdot p^u, w >_{V' \times V} \;=\; < \frac{\partial J}{\partial y}(u, y^u), w >_{V' \times V} \quad \forall w \in V, \tag{10.26}$$

We then reach our goal: an expression of $j'(u)$ *independent of* $w^{\delta u}$.

$$j'(u)\cdot\delta u \;=\; \frac{\partial J}{\partial u}(u;y^u)\cdot\delta u - <\left(\frac{\partial A}{\partial u}(u;y^u)-\frac{\partial F}{\partial u}(u)\right)\cdot\delta u, p^u>_{V'\times V}$$

We denote indifferently $j'(u)\cdot\delta u \equiv < j'(u),\delta u >_{U'\times U}$ with $< .,. >_{U'\times U}$ the duality product in U (U Banach space).

We obtain the expected expression: $\forall\delta u\in U$,

$$< j'(u),\delta u >_{U'\times U} \;=\; < \frac{\partial J}{\partial u}(u;y^u),\delta u >_{U'\times U} - <\left(\frac{\partial A}{\partial u}(u;y^u)-\frac{\partial F}{\partial u}(u)\right)^*\cdot p^u,\delta u >_{U'\times U} \tag{10.27}$$

Therefore the explicit expression of $j'(u)$ in $U'=\mathcal{L}(U,\mathbb{R})$,

$$j'(u) \;=\; \frac{\partial J}{\partial u}(u;y^u) \;-\; \left(\frac{\partial A}{\partial u}(u;y^u)-\frac{\partial F}{\partial u}(u)\right)^*\cdot p^u \quad \text{in } U' \tag{10.28}$$

This expression of $j'(u)$ independent of $w^{\delta u}$ relies on the so-called adjoint equation (10.26).

10.6.2 The general key result

Theorem 10.7. *Let us consider the direct model (10.1) and the cost function $j(u)$ defined by (10.2)-(10.3). It is assumed that:*
i) the state equation (10.6) is well-posed,
ii) the TLM (10.22) is well-posed,
iii) the operators $A(u;y)$, $F(u)$, see (10.6), are C^1 with respect to u.
Then, given a C^1 objective function $J(u;y)$ and the cost function $j(u)$ defined by (10.3), the cost function $j(c)$ is of class C^1.
Moreover, the expression of the differential $j'(u)$ reads: $\forall\delta u\in U$,

$$j'(u)\cdot\delta u \;=\; \frac{\partial J}{\partial u}(u;y^u)\cdot\delta u - \left(\frac{\partial a}{\partial u}(u;y^u,p^u)\cdot\delta u - \frac{\partial b}{\partial u}(u;p^u)\cdot\delta u\right) \tag{10.29}$$

with $\partial_u J(u;y^u) = \alpha_{reg}J'_{reg}(u)$ if considering the particular decomposition (10.2) of $J(u;y)$.

y^u is the unique solution of the state equation (10.6),
and p^u is solution of the adjoint equation:

$$\begin{cases} \text{Given } u \text{ and } y^u \text{ the unique solution of (10.6),} \\ \text{find } p \in V \text{ satisfying:} \\ \dfrac{\partial a}{\partial y}(u; y^u, p) \cdot z = \dfrac{\partial J}{\partial y}(u, y^u) \cdot z \qquad \forall z \in V \end{cases} \tag{10.30}$$

Its solution p^u (the adjoint state) exists and is unique.
One has $\partial_y J(u; y^u) = J'_{obs}(y)$ if considering the particular decomposition (10.2) of $J(u; y)$.

Proof.
Under assumptions i)-iii), the implicit function theorem applies and the differentiability of the state with respect to u follows: the operator $\mathcal{M} : u \in U \mapsto y^u \in V$ is C^1. Therefore $j(u)$ is of class C^1 too.
We have: $j'(u) \in \mathcal{L}(U; \mathbb{R})$. As already written above:

$$< j'(u), \delta u >_{U' \times U} \;=\; < \frac{\partial J}{\partial u}(u; y^u), \delta u >_{U' \times U} + < \frac{\partial J}{\partial y}(u; y^u), w^{\delta u} >_{V' \times V} \quad \forall \delta u \in U \tag{10.31}$$

(see Lemma (10.17)) with $< \cdot, \cdot >_{U' \times U}$, $< \cdot, \cdot >_{V' \times V}$ the corresponding duality products.
For sake of simplicity, we denote: $j'(u) \cdot \delta u \equiv < j'(u), \delta u >_{U' \times U}$.
By this equation with the TLM in weak form, see (10.24), we obtain:

$$\begin{aligned} < j'(u), \delta u >_{U' \times U} \;=\; & < \frac{\partial J}{\partial u}(u; y^u), \delta u >_{U' \times U} \\ & - < \left(\frac{\partial A}{\partial u}(u; y^u) - \frac{\partial F}{\partial u}(u) \right) \cdot \delta u, z >_{V' \times V} \\ & + < \left(\frac{\partial J}{\partial y}(u; y^u) - \left(\frac{\partial A}{\partial y} \right)^* (u; y^u) \cdot z \right), w^{\delta u} >_{V' \times V} \quad \forall \delta u \in U \; \forall z \end{aligned}$$

where $(\partial_y A)^*$ is the adjoint operator of the linearized direct model operator $\partial_y A$.

The linearized problem is well-posed therefore the operator $\partial_y A(u; y^u)$ is an isomorphism from V into V', and its adjoint operator $(\partial_y A)^*(u; y^u)$ is an isomorphism from V into V' too, see e.g. [10].
As a consequence, the adjoint equation (10.26) is well-posed too.
$p^u \in V$ is defined as its unique solution. We obtain the final expression (10.29) of $j'(u)$. □

Advantages and drawbacks of the adjoint-based expression

$\oplus$: The expression of $j'(u) \cdot \delta u$ does not depend on $w^{\delta u}$ anymore: the expression of $j'(u)$ is explicit with respect to the direction δu.
Thus, after discretization if solving the direct model plus the adjoint model then *all components of the gradient* follow i.e. the complete gradient vector.
In other words, for m large, the adjoint-based approach enables to obtain the m gradient components by one (1) extra system to solve only.

Let us recall that after discretization (in the finite dimension space U_h), we have:

$$\nabla j(u) \in \mathbb{R}^m, \ \ < \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u \text{ for all } \delta u \in U_h \subset \mathbb{R}^m$$

$\ominus$: The adjoint model has to be implemented (intrusive approach).
This important drawback may be done by automatic differentiation. This option is more or less complex depending on the direct code complexity and the programming langage.

Remarks

- *By construction, the adjoint model is linear*, whatever if the direct model is linear or not. Recall the adjoint is the adjoint operator of the linearized direct operator.

- Let us point out that excepted for few particular cases (e.g. if the operator A is self-adjoint, $A^* = A$, of course), the adjoint model has in the general case no physical meaning.

- If the direct operator is *self-adjoint*, in other words if $a(u, v)$ is bilinear symmetric, then the adjoint operator equals the direct operator (but the RHS).
  Indeed, in such a case, we have:

$$\partial_y a(u; y^u, p).z \underbrace{=}_{linear} a(u; z, p) \underbrace{=}_{symmetric} a(u; p, z) \tag{10.32}$$

Only the source term (RHS) and the boundary conditions differ from the state equation.
Then the differential operator, hence the numerical method and numerical solver, are the same.

Case of non-homogeneous Dirichlet boundary conditions

$$\left\{ \begin{array}{l} \text{Find } y \in V_t = V_0 \oplus y_d \text{ such that :} \\ a(u; y, z) = b(u; z) \text{ for all } z \in V_0 \end{array} \right.$$

A typical example for a second order linear elliptic equation is : $V = H^1(\Omega)$, $V_0 = \{z \in V,\ z = 0 \text{ on } \Gamma_d\}$ and $V_t = V_0 \oplus y_d = \{z \in V,\ z = y_d \text{ on } \Gamma_d\}$.

Then the question is : What the non-homogeneous Dirichlet boundary conditions becomes when defining the TLM hence the adjoint model ?

The answer is : *the non-homogeneous Dirichlet condition in the direct model becomes the corresponding homogeneous condition in the linear tangent and adjoint models.*

Let us show this statement in the linear case.
The direct model, if linear in y, re-reads as follows:

$$\left\{ \begin{array}{l} \text{Find } y_0 \in V_0 \text{ such that :} \\ a(u; y_0, z) = b(u; z) - a(u; \tilde{y}_d, z) = \tilde{b}(u; z) \text{ for all } z \in V_0 \end{array} \right.$$

with $y_0 = y - \tilde{y}_d$, $\tilde{y}_d$ being a raising from y_d on Γ_d onto the whole domain Ω.

Following the proof of Theorem 10.7, it is easy to notice that the corresponding boundary condition in the TLM is homogeneous, hence the same for the adjoint model.

The optimality system

$$\left\{\begin{array}{l} a(u;y^u,z) \;=\; b(u;z) \quad \forall z \in V \\ \partial_y a(u;y^u,p)\cdot z \;=\; J'_{obs}(y^u)\cdot z \quad \forall z \in V \\ j'(u)\cdot \delta u \;=\; 0 \quad \forall \delta u \in U \\ \quad \text{with } j'(u)\cdot \delta u \;=\; \alpha_{reg} J'_{reg}(u)\cdot \delta u - (\partial_u a(u;y^u,p^u) - \partial_u b(u;p^u))\cdot \delta u \end{array}\right. \tag{10.33}$$

The optimality system (10.33) is nothing else than the stationary point conditions (10.10) of the Lagrangian. The adjoint state p is the lagrangian multiplier associated to the "-model constraint".

Exercises

Exercice 10.8. *Write the adjoint of a few classical second order and first order operators, in the case of*

- *Dirichlet BC all over the boundary,*
- *mixed B.C.*

See details of the enunciation in the supplementary material.

Exercice 10.9. *Write the optimality system which characterizes the solution u of the optimal control problem of your practical. Detail both the weak and the classical forms.*

10.7 The VDA algorithm (3D-var)

10.7.1 Gauss-Newton vs Quasi-Newton

Assume that the gradient expression $\nabla j(u_h)$ is available for any $u_h \in U_h$ by performing a direct model solver and an adjoint model solver.
A natural approach (and efficient) approach consists to use the Newton-Raphson algorithm to solve the first order optimality condition (Euler's equation) $\nabla j(u) = 0$.

This numerical approach is very efficient since second order (when converging). However, it requires the computation of the Hessian H_j of j which is often highly complex or too CPU time consuming to compute...
That is why in many cases, first order descent algorithms based on the gradient information only are preferred.
In practice, we use Quasi-Newton methods like the BFGS algorithm, [**?**].

10.7.2 The 3D-Var algorithm

Given a *first guess* u_0, we seek $(u^m)_m$ that decreases the cost function using a Quasi-Newton method e.g. the L-BFGS algorithm.
The algorithm is as follows, see Fig. (10.3).

Given the current control value u,

1) compute the cost function $j(u)$ from the direct model output y^u,
2) compute the gradient $\nabla j(u)$ from the adjoint model output p^u and the direct model output y^u,
3) given u, $j(u)$ and $\nabla j(u)$, compute the new iterate u^{new} as $u^{new} = u + \alpha d(\nabla j(u))$ where $d \in \mathbb{R}^m$ is the descent direction and $\alpha \in \mathbb{R}_*^+$ the step in the linear search.
   The descent algorithm simply ensures that:

$$j(u^{new}) < j(u)$$

*) Iterate until convergence.

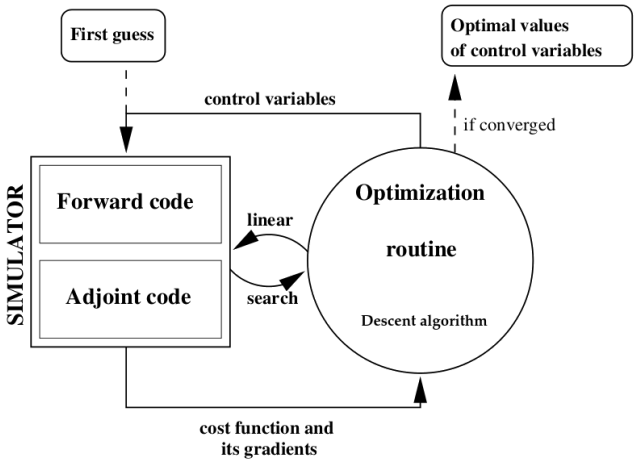


Figure 10.3: VDA algorithm: optimal control of the PDE system (identification process). This provides the so-called 3D-Var algorithm. (4D-var in its unsteady version, see next chapter).

10.8 The fundamental equations at a glance

10.8.1 General continuous formalism

The considered general **non-linear stationary PDE model** reads:

$$\left\{ \begin{array}{l} \text{Given } u(x), \text{ find } y(x) \text{ such that:} \\ A(u(x); y(x)) = F(u(x)) \text{ in } \Omega \\ \text{with Boundary Conditions on } \partial\Omega \end{array} \right. \tag{10.34}$$

In weak form:

$$u \in V_t : \; a(u; y^u, z) \; = \; b(u, z) \quad \forall z \in V_0 \tag{10.35}$$

The direct model (= the state equation) is supposed to be well-posed.

The parameter-to-state operator ("model operator") $\mathcal{M}(u)$ is defined as: $\mathcal{M}(u) = y^u$.
This operator $\mathcal{M}(\cdot)$, which is a-priori non-linear, is supposed to continuous (well-posed direct model).

The cost function $j(u)$ is defined from the observation function $J(u; y)$ as follows:

$$j(u) = J(u\,; y^u) \tag{10.36}$$

where $y^u(x)$ denotes the unique solution of the direct model, given $u(x)$.

The observation function $J(u; y)$ is classically decomposed as follows:

$$J(u\,; y) \; = \; J_{obs}(y) \; + \alpha_{reg} \; J_{reg}(u) \tag{10.37}$$

with the data misfit term:

$$J_{obs}(y) \; = \; \frac{1}{2} \Big\| \, Z(y) - z_d \Big\|^2_{R^{-1}} \tag{10.38}$$

The observation operator $Z(\cdot)$ may be linear or not.
The regularization term is here defined from quadratic terms (in u or higher-order terms).
Classical expressions are: $J_{reg}(u) \; = \; \frac{1}{2} \Big\| \, u - u_b \Big\|^2_{B^{-1}}$ or $\frac{1}{2} \Big\| \, D^p u \Big\|^2_0$ with $p = 1$ or 2.

The inverse problem is formulated as the following **optimization problem**:

$$\left\{ \begin{array}{l} \text{Minimize } j(u) \text{ in } U_{ad} \text{ under the "model constraint"} \\ \text{ since } j(u) = J(u; y^u) \text{ with } \ y^u = \mathcal{M}(u). \end{array} \right. \tag{10.39}$$

The adjoint model reads:

$$\left\{ \begin{array}{l} \text{Given } u(x), \text{given } y^u(x), \text{ find } p(x) \text{ such that:} \\ (\partial_y A)^*(u(x); y^u)(x) \cdot p(x) \ = \ J'_{obs}(y^u(x)) \text{ in } \Omega \\ \text{with the } \underline{\text{adjoint B.C.}} \text{ on } \partial\Omega \end{array} \right. \tag{10.40}$$

In weak form. By defining $a^*((u, y^u); p, z) \equiv \partial_y a(u; y^u, p) \cdot z$, the adjoint equation reads

$$p \in V_0 : \ a^*((u, y^u); p, z) \ = \ J'_{obs}(y^u) \cdot z \quad \forall z \in V_0 \tag{10.41}$$

The resulting gradient expression.
The relationship between the gradient $\nabla j(u)$ and the differential $j'(u)$ is:

$$< \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u \text{ for all } \delta u \in U_h \subset \mathbb{R}^m \tag{10.42}$$

The differential of j reads: for all $\delta u \in U$,

$$j'(u) \cdot \delta u \;=\; \alpha_{reg} J'_{reg}(u) \cdot \delta u - (\partial_u A(u; y^u) \cdot \delta u - F'(u) \cdot \delta u) \cdot p^u \tag{10.43}$$

In weak form.

$$j'(u) \cdot \delta u \;=\; \alpha_{reg} J'_{reg}(u) \cdot \delta u - \partial_u a(u; y^u, p^u) \cdot \delta u + \partial_u b(u; p^u) \cdot \delta u \tag{10.44}$$

In the case of a composite gradient i.e. c containing different components, $c = (u_1, u_2)$, we have:

$$j'(u) \equiv (\frac{\partial j}{\partial u_1}(u), \frac{\partial j}{\partial u_2}(u))^T \text{ and } j'(u) \cdot \delta u = \frac{\partial j}{\partial u_1}(u) \cdot \delta u_1 + \frac{\partial j}{\partial u_2}(u) \cdot \delta u_2$$

10.8.2 Discrete formalism

Recall that the general continuous formalism above, based on the weak form of the equations, enables to rigorously derive the adjoint equations and the gradient expression, including in the non-linear case, even if the boundary conditions are non trivial. The discrete formalism below is easier to handle but it may be incomplete in presence of complex boundary conditions.

The direct model reads:

$$\left\{ \begin{array}{l} \text{Given } u \in \mathbb{R}^m, \text{ find } y \in \mathbb{R}^n \text{ such that:} \\ A(u;y) = F(u) \end{array} \right. \tag{10.45}$$

$A(u;y)$ represents a system of n (non-linear) equations at n unknowns $(y_1, \ldots, y_n)$. One has: $A : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$; $A(u;y) \in \mathbb{R}^n$. $F : \mathbb{R}^m \to \mathbb{R}^n$; $F(u) \in \mathbb{R}^n$.

Let us recall the adjoint property (transpose in the real case) of a $n \times n$-matrix M (therefore a linear operator from $\mathbb{R}^n$ onto $\mathbb{R}^n$): for y and z in $\mathbb{R}^n$, $(M^T y, z)_{\mathbb{R}^n} = (y, Mz)_{\mathbb{R}^n}$.

Given these conventions, one has the adjoint model which reads:

$$\left\{ \begin{array}{l} \text{Given } u \in \mathbb{R}^m, \text{given } y^u \in \mathbb{R}^n, \text{ find } p \in \mathbb{R}^n \text{ such that:} \\ (D_y A(u;y^u))^T p \;=\; J'_{obs}(y^u) \\ \text{(modulo the Dirichlet boundary conditions)} \end{array} \right. \tag{10.46}$$

This is a linear system of n equations at n unknowns $(p_1, \ldots, p_n)$.

The resulting gradient $\nabla j(u) \in \mathbb{R}^m$ reads:

$$\nabla j(u) \;=\; \alpha_{reg} \nabla J_{reg}(u) - \left(D_u A(u;y^u)\right)^T p^u + \left(D_u F(u)\right)^T p^u \tag{10.47}$$

In the case of few components e.g. $u = (u_1, u_2)\mathbb{R}^{m_1} \times \mathbb{R}^{m_2}$, then one simply has:

$$\nabla_u j(u) = \left(\nabla_{u_1} j(u), \nabla_{u_2} j(u)\right)^T$$

10.9 Practical aspects

10.9.1 Validate your codes: computed gradients

Validating a computational code is a mandatory step before performing simulations. Below are described methods how to validate the adjoint code and the cost fucntion gradient.

- *Validation of the adjoint code.* It can be verified that the code actually computes the adjoint of the tangent linear code by computing the scalar product property. This supposes however to have developed the tangent linear code too.
- *Validation of the gradient.* The adjoint-based gradient can be compared to finite differences values. This is the so-called the gradient test. This test should be done for any computational code computing a model output gradient.

The scalar product test

This test aims at checking if the adjoint code is actually the adjoint of the Tangent Linear code. This test supposes to have developed both the adjoint code and the linear tangent code.

The test aims at numerically verifying the definition of an adjoint operator. Let M be a linear operator defined from $\mathcal{U}$ to $\mathcal{Y}$, we have:

$$< Mu, y >_{\mathcal{Y}} = < u, M^* y >_{\mathcal{U}}$$

Let u_0 be a given parameter value.

- Given an arbitrary perturbation $du \in \mathcal{U}$, the Tangent Linear code output is computed:
$$dy = \left(\frac{\partial \mathcal{M}}{\partial u}(u_0) \right) \cdot du$$
- Given an arbitrary perturbation $dy^* \in \mathcal{Y}$, the adjoint code output is computed:
$$du^* = \left(\frac{\partial \mathcal{M}}{\partial u}(u_0) \right)^* \cdot dy^*$$
- The two following scalar products are computed:
$$sp_y = \left\langle dy^*, dy \right\rangle_{\mathcal{Y}} \text{ and } sp_u = \left\langle du^*, du \right\rangle_{\mathcal{U}}$$
- The validation relies on the relation: $sp_y = sp_u$.

Figure 10.4 (b) shows a typical example of the scalar product test.

```
####################################
##    TEST DU PRODUIT SCALAIRE    ##
####################################

Appel du code linaire tangent...
Appel du code adjoint...
<Xd,Xb> =   -682.277083033688
<Yd,Yb> =   -682.277082428555
relative error :  -8.869324203484993E-010
```

Figure 10.4: Adjoint code validation: scalar product test

The gradient test

The objective of this test aims at verifying that the gradient obtained from the adjoint corresponds to the partial derivatives of the cost function.
If first using an adjoint code, this test must be done before any further computations based on the adjoint-based gradient.
This test requires to perform a dozen of times the direct code and one time the adjoint code. The Tangent Linear code is here not required.

Let u_0 be a given parameter value. The Taylor expansion of the cost function j at u_0 for a small perturbation $\alpha\,\delta u$ ($\alpha \in \mathbb{R}^+$) reads:

$$j(u_0 + \alpha\,\delta u) \;=\; j(u_0) + \alpha\,\frac{\partial j}{\partial u}(u_0)\cdot\delta u + o\left(\alpha\|\delta u\|\right)\;. \tag{10.48}$$

It follows the uncentered finite difference approximation (order 1) and the centered finite difference approximation (order 2):

$$\frac{j(u_0 + \alpha\,\delta u) - j(u_0 - \alpha\,\delta u)}{2\alpha} \;= \frac{\partial j}{\partial u}(u_0)\cdot\delta u + O\left(\alpha^2\|\delta u\|^2\right)\;. \tag{10.49}$$

Then, we set either

$$I_\alpha \;=\; \frac{j(u_0 + \alpha\,\delta u) - j(u_0 - \alpha\,\delta u)}{2\alpha\;\frac{\partial j}{\partial u}(u_0)\cdot\delta u} \tag{10.50}$$

or

$$I_\alpha \;=\; \frac{j(u_0 + \alpha\,\delta u) - j(u_0)}{\alpha\;\frac{\partial j}{\partial u}(u_0)\cdot\delta u} \tag{10.51}$$

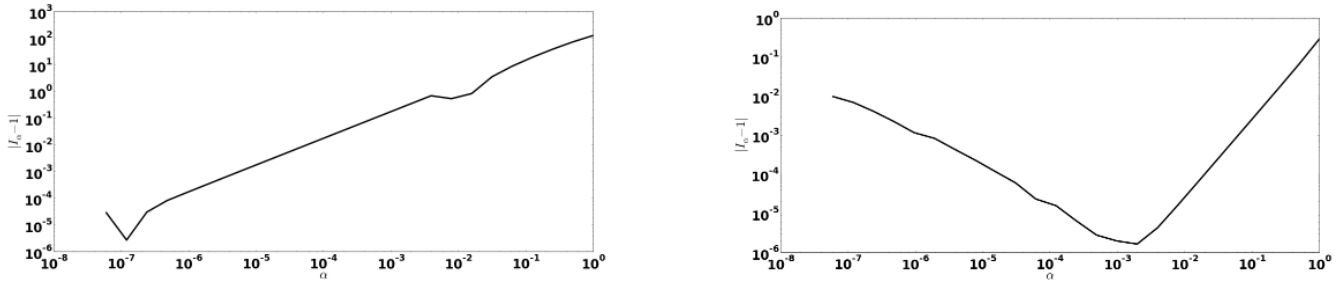
According to the Taylor expansions above, we have: $\lim\limits_{\alpha\to 0} I_\alpha = 1$.

The gradient test consists to check this property as follows.

- Given an arbitrary parameter value u_0, compute $\frac{\partial j}{\partial u}(u_0)$ using the adjoint code.
- Using the direct code, compute $j(u_0)$.
- For $n = 0, \ldots, N$:
  - Compute $\alpha_n = 2^{-n}$;
  - Using the direct code, compute $j(u_0 + \alpha_n \, \delta u)$;
  - Compute I_{α_n} ;
- Verify if $\lim_{\alpha \to 0} I_{\alpha_n} = 1$ or not.

Figure 10.5 shows two results of the gradient test: at order 2 and at order 1. $|I_\alpha - 1|$ is plotted vs α in logarithmic scale.
The convergence is good until $\alpha > 10^{-7}$.
However, observe the difference of accuracy between the 1st order and 2nd order approximation.

In the present exemple, the truncation errors errors appear for α smaller than $\approx 10^{-7}$ at order 1 ($\approx 10^{-3}$ at order 2). (To show this statement, one add a fix term in the Taylor expansion and notice that it is divided by the perturbation therefore increasing).



(a) (b)

Figure 10.5: The adjoint code validation. Gradient test at order 1 (a), at order 2 (b).

Exercice 10.10. *Perform the gradient test for your practical problem.*
Is your gradient computation valid ?

10.9.2 Twin experiments

When addressing a real-world problem with a DA approach, the first mandatory step is to analyze *twin experiments* with increasing complexity. The principle of twin experiments is as follows.

- First, a dataset (the observations z^{obs}) is generated by applying the direct model to the input parameter u (this value will be referred to as the "true" value, denoted by u_t). The obtained observations are perfect in the sense that they are free from model errors. Next, noise (e.g., Gaussian noise with a realistic amplitude) is added to these perfectly synthetic data.
- Second, the optimal control process is performed starting from an initial guess value u_b that differs from the "true" value u_t (the one used to generate the synthetic data).

As a consequence, since the true solution u_t corresponding to z^{obs} is known, thorough investigations can be conducted.

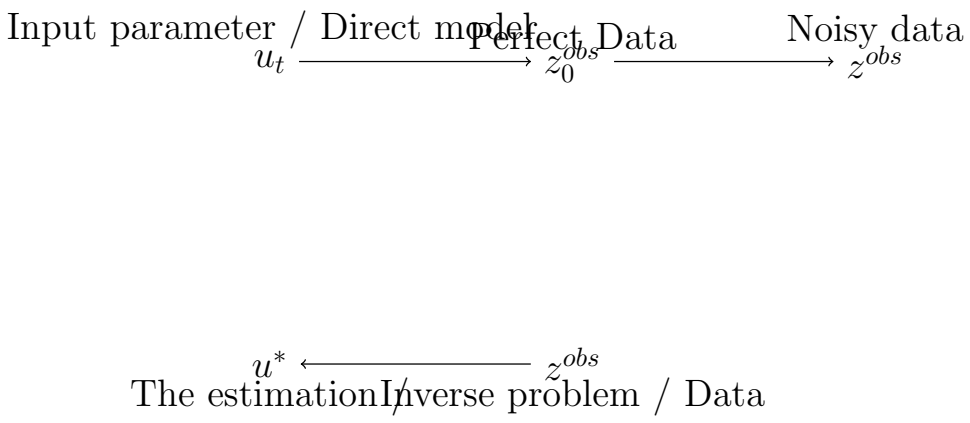


Figure 10.6: Twin experiments concept.

Chapter 11

VDA for Time-Dependent PDEs

The outline of this chapter is as follows[1].

Contents

[1]Recall that the sections indicated with a * are "to go further sections". They can be skipped in a first reading or if the reader is not particularly interested in deeper mathematical basis, mathematical proofs.

11.1 The inverse formulation

11.1.1 The general direct model

$$(\mathcal{D})\begin{cases} \text{Given the I.C. } y_0(x), \text{ given the space-time control } u(x,t), \\ \text{ find the state } y(x,t) \text{ satisfying:} \\ \partial_t\, y(x,t) \;+\; A(u(x,t)\,;y(x,t)) \;=\; F(u(x,t)) \qquad \text{in } \Omega\times]0,T[\\ y(x,0) \;=\; y_0(x) \text{ in } \Omega \\ \text{with Boundary Conditions for all } t \end{cases} \tag{11.1}$$

Examples of direct models As a scalar parabolic equation example, the reader may guess to the heat equation (scalar linear parabolic equation) or to the non-linear case:

$$A(u;y) = -div(\lambda(u_1;y)\nabla y) + w\cdot\nabla y + cy \text{ and } F(u) = u_2$$

with $u = (u_1, u_2)$.

In real-world problems, the I.C. is often uncertain. The I.C. can even be the most important "parameter" to be identified/estimated e.g. in atmosphere dynamic problems for weather prediction. Then, we consider the control variable enriched with the I.C. as:

$$\boxed{c(x,t) = (y_0(x), u(x,t))} \tag{11.2}$$

We assume that the direct model is well posed in the following sense.

Assumption 11.1. *Given $c(x,t) \in \mathcal{C}$ and $T > 0$, it exists a unique function $y(x,t)$, $y \in W_V(0,T)$, solution of Problem $(\mathcal{D})$. Furthermore, this unique solution y depends continuously on $c(x,t)$.*

Assumptions of differentiability* The parameter-to-state operator (the model operator too) reads here as:

$$\boxed{\mathcal{M} : \mathcal{C} \to \ W_V(0,T) : c(x,t) = (y_0(x), u(x,t)) \mapsto y(c(x,t); x, t)} \tag{11.3}$$

The direct model is well-posed implies that $\mathcal{M}(y)$ is continuous, for all t.

To consider cost functions $j(c)$ of class C^1 (continuously differentiable), one needs to assume that the state y is differentiable with respect to the control parameter c that is

Assumption 11.2. *The model operator $\mathcal{M}(c)$ is continuously differentiable for all $t \in]0, T[$.*

Under the assumption above, one can formally write:

$$y(c + \delta c; t) \ = \ y(c; t) + \frac{dy}{dc}(c; t) \cdot \delta c + o_0(\|\delta c\|_c) \tag{11.4}$$

In some cases, this differentiability property is not satisfied at all control value c.
Indeed, in the case of a non-linear hyperbolic system such as the Euler (or Shallow Water) equations for example, a shock can appear when making varies continuously e.g. a physical model parameter or a boundary condition value. In this case, the operator $\mathcal{M}(c)$ is even not continuous, therefore cannot be differentiable.

11.1.2 Cost function terms: data misfit and regularizations

The objective function J is decomposed as in the stationary case:

$$\boxed{J(c\,;y) \;=\; J_{obs}(y) \;+\; \alpha_{reg} J_{reg}(c)} \tag{11.5}$$

Then, the cost function j to be minimized is defined from J as usual:

$$\boxed{j(c(x,t)) = J(c(x,t)\,;y^c(x,t)))} \tag{11.6}$$

where $y^c(x,t)$ is the (unique) solution of the direct model $(\mathcal{D})$, given $c(x,t)$.

Data misfit term In the unsteady case, the misfit term J_{obs} is naturally integrated in time as follows:

$$J_{obs}(y(x,t)) \;=\; \int_0^T \Big\| \, Z(y(x,t)) - z^{obs}(x,t) \Big\|^2_{R^{-1}} \, dt \tag{11.7}$$

with $Z(\cdot)$ the observation operator.

In practice, observations are often local ("point-wise"), moreover very sparse. Then, in the case the observations are provided as time-series, the misfit observation term $J_{obs}(y)$ reads as:

$$\boxed{J_{obs}(y) \;=\; \sum_{n=1}^{N} \Big\| \, Z(y(t_n^{obs})) - z_n^{obs} \Big\|^2_{R^{-1}}} \tag{11.8}$$

where z_n^{obs} is the measurement at instant $t_n^{obs},\ n = 1, \ldots, N$.

Regularization terms Recall that $c(x,t) = (y_0(x), u(x,t))$. Then, we naturally consider a regularization term for each control component as:

$$J_{reg}(c(x,t)) = J^0_{reg}(y_0(x)) + J^u_{reg}(u(x,t)) \tag{11.9}$$

If considering a background value y_b for the I.C. to attract the minimization algorithm to this value, we set: $J^0_{reg}(y_0) = \left\| y_0 - y_b \right\|^2_{B^{-1}}$.

The norm B^{-1} may be defined to represent "at best" the inverse of the covariance matrix of the background error matrix B as already discussed, see Section 7.3.

For the control parameter $u(x,t)$, one considers the same regularization terms as in the stationary case (see Section **??**).

‘

11.1.3 The optimization problem

The optimization problem writes similarly to the stationary case:

$$\boxed{\begin{cases} \text{Find } c^*(x,t) = (y_0^*(x), u^*(x,t)) \in H \times \mathcal{U}_T \text{ such that:} \\ j(c^*) = \min\limits_{H \times \mathcal{U}_T} j(c) \end{cases}} \tag{11.10}$$

The control parameter vector c is here a-priori time-dependent. As a consequence, its discrete version is an extremely large vector! Therefore the optimization problem is a large dimension one. One of the consequence is that $\nabla j(c)$ has to be computed by employing the adjoint method.

11.2 Optimality equations in finite dimension (discrete forms)

The formal direct model reads:

$$\boxed{(D_d)\begin{cases}\text{Find } \{y_k\}_{1\leq k\leq N_T}\in\mathbb{R}^n \text{ such that :}\\ y_{k+1}=\mathcal{M}_k(y_k) \quad k=0,1,...,N_T \text{ (the time steps)}\\ y_0\in\mathbb{R}^n \text{ given.}\end{cases}} \tag{11.11}$$

where $\mathcal{M}_k(y_k)$ represents the n *non-linear* equations at instant t_k.
We denote by $M_k = D_y\mathcal{M}_k(y_k)$ the linearized model equations at "point" y_k, $M_k \in \mathbb{R}^{n\times n}$.

For a sake of simplicity again, the control variable c is simply here the I.C.:

$$\boxed{c = y_0 \in \mathbb{R}^n}$$

Moreover, it is supposed that: a) the observations $z^{obs} \in \mathbb{R}^m$ are available at all time step; b) the regularization term relies on the knowledge of a good background value y_b.

Then, the cost function reads:

$$j(y_0) = \frac{1}{2}\sum_{k=1}^{N_T}(\mathcal{Z}(y_k)-z_k^{obs})^T R^{-1}(\mathcal{Z}(y_k)-z_k^{obs})+\alpha_0\frac{1}{2}(y_0-y_b)^T B(y_0-y_b) \tag{11.12}$$

where $\mathcal{Z}:\mathbb{R}^n\to\mathbb{R}^m$ is the observation operator which is a-priori non-linear too.

We denote the linear tangent observation operator taken at y_k as: $Z_k = D_y\mathcal{Z}(y_k)$, $Z_k \in \mathbb{R}^{m\times n}$.

The cost gradient satisfies: for all δy_0 in $\mathbb{R}^n$,

$$< \nabla j(y_0), \delta y_0 >_{\mathbb{R}^n} = \sum_{k=1}^{N_T} < \Delta_k, Z_k \cdot w_k^\delta >_{\mathbb{R}^m} + \alpha_0 < B(y_0 - y_b), \delta y_0 >_{\mathbb{R}^n}$$

with $w_k^\delta = D_{y_0}(y_k) \cdot \delta y_0$, $w_k^\delta \in \mathbb{R}^n$ and the notation $\Delta_k = R^{-1}(\mathcal{Z}(y_k) - z_k^{obs})$.

By differentiating the direct model, we get:

$$D_{y_0}(y_{k+1}) \cdot \delta y_0 = M_k \cdot D_{y_0}(y_k) \cdot \delta y_0 \ , \ k = 1, ..., N_T \tag{11.13}$$

with the I.C. δy_0 given.
The TLM above reads:

$$w_{k+1}^\delta = M_k w_k^\delta = (M_k \dots M_0) \delta y_0 \tag{11.14}$$

By injecting this into the gradient expression, we obtain:

$$< \nabla j(y_0), \delta y_0 >_{\mathbb{R}^n} = \sum_{k=1}^{N_T} < (M_0^T \dots M_{k-1}^T) Z_k^T \Delta_k, \delta y_0 >_{\mathbb{R}^n} + \alpha_0 < B(y_0 - y_b), \delta y_0 >_{\mathbb{R}^n} \tag{11.15}$$

The 1st term $\sum_{k=1}^{N_T} \left(M_0^T \dots M_{k-1}^T\right) Z_k^T \Delta_k$ in the expression above reads:

$$\begin{aligned} & M_0^T Z_1^T \Delta_1 + (M_0^T M_1^T) Z_2^T \Delta_2 + \dots + (M_0^T M_1^T \dots M_{N_T-1}^T) Z_{N_T}^T \Delta_{N_T} \\ & = M_0^T \left[Z_1^T \Delta_1 + M_1^T \left[Z_2^T \Delta_2 + [\dots] M_{N_T-1}^T Z_{N_T}^T \Delta_{N_T} \right] \dots \right] \end{aligned} \tag{11.16}$$

(11.17)

Let us define the sequence p_k defined as follows: for $k = (N_T - 1), \ldots, 0$,

$$p_k = M_k^T p_{k+1} + Z_k^T \Delta_k \tag{11.18}$$

with $p_{N_T} = 0$, and (recall) $\Delta_k = R^{-1}(\mathcal{Z}(y_k) - z_k)$.
We have:

$$\begin{aligned}
p_{N_T-1} &= Z_{N_T-1}^T \Delta_{N_T-1} \\
p_{N_T-2} &= M_{N_T-2}^T \left[Z_{N_T-1}^T \Delta_{N_T-1} \right] + Z_{N_T-2}^T \Delta_{N_T-2} \\
p_{N_T-3} &= M_{N_T-3}^T \left[M_{N_T-2}^T \left[Z_{N_T-1}^T \Delta_{N_T-1} \right] + Z_{N_T-2}^T \Delta_{N_T-2} \right] + Z_{N_T-3}^T \Delta_{N_T-3} \\
&\vdots \\
p_0 &= M_0^T \left[M_1^T \left[\ldots \left[M_{N_T-3}^T \left[M_{N_T-2}^T \left[Z_{N_T-1}^T \Delta_{N_T-1} \right] + Z_{N_T-2}^T \Delta_{N_T-2} \right] + \ldots \right] + \ldots \right] + Z
\end{aligned}$$

This is a Horner type factorization.
It can be shown that the two expressions (11.16) and (11.19) are equal.
ToDo: calcul pas clair... a verifier.... mais le resultat est bien celui-ci (cf demo continue)
Therefore:

$$p_0 = \sum_{k=1}^{N_T} \left(M_0^T \ldots M_{k-1}^T \right) Z_k^T \Delta_k \tag{11.21}$$

In conclusion, by introducing the following sequence (defining the discrete adjoint model):

$$(A_d) \begin{cases} \text{Given the state at each time step } \{y_k\}_{1 \le k \le N_T} \in \mathbb{R}^n, \text{ find } \{p_k\}_{(N_T-1) \ge k \ge 0} \in \mathbb{R}^n \text{ su} \\ p_k = M_k^T p_{k+1} + Z_k^T R^{-1}(\mathcal{Z}(y_k) - z_k) \text{ for } k = N_T, ..., 1 \text{ (the time steps)} \\ p_{N_T} = 0 \end{cases} \tag{11.22}$$

with $Z_k = D_y \mathcal{Z}(y_k)$, $Z_k \in \mathbb{R}^{m \times n}$, the gradient simply reads as, see (11.15):

$$\boxed{\nabla j(y_0) = p_0 + \alpha_0 B(y_0 - y_b)} \tag{11.23}$$

A few remarks

- The adjoint model is retroacting in time. Its initial condition must be given at final time T.
- The gradient with respect to the Initial Condition equals the adjoint state at

initial time (modulo the minus sign and the regularization term).

11.3 The optimality equations in infinite dimension (continuous forms)

11.3.1 The TLM-based gradient

The approach is the same as in the stationary case: we derive the cost function $j(c)$, we obtain the differential expression $j'(c)$ in function of the state derivative $w^{\delta c} = \frac{dy}{dc}(c) \cdot \delta c$. The $w^{\delta c}$ is by construction the solution of the Tangent Linear Model (TLM).

Gradient expression depending on the term $w^{\delta c}$

By deriving the cost function (10.3), we get:

$$j'(c) \cdot \delta c \;=\; \partial_c J(c\,; y^c).\delta c + \partial_y J(c\,; y^c).w^{\delta c} \tag{11.24}$$

with $w^{\delta c} = \frac{dy}{dc}(c) \cdot \delta c$ and $c = (y_0, u)$.
If considering the particular form (11.5) (with $\alpha_{reg} = 1$) then:

$$\boxed{j'(c) \cdot \delta c \;=\; J'_{obs}(y^c) \cdot w^{\delta c} + J'_{reg}(c) \cdot \delta c} \tag{11.25}$$

Let us consider the observation term as previously (see Section 11.1.2) with a linear observation operator Z (for a sake of simplicity):

$$J_{obs}(y) \;=\; \frac{1}{2}\int_0^T \Big\| Zy(t) - z^{obs}(t)\Big\|^2_{R^{-1}}\, dt \tag{11.26}$$

For the regularization term, one can naturally consider:

$$J_{reg}(c) = \frac{1}{2}\alpha_{reg,0}\Big\| y_0 - y_b\Big\|^2_{B^{-1}} + \frac{1}{2}\alpha_{reg,u}\Big\| \nabla u\Big\|^2_2 \tag{11.27}$$

with the weights $\alpha_{reg,\square}$ to be determined.
With the definitions above, it follows the expression:

$$j'(c)\cdot\delta c \;=\; \int_0^T \Big\langle\, Z^T R(Z\, y^c(t) - z^{obs}(t)),\; w^{\delta c}(t)\,\Big\rangle\, dt + \big\langle B^{-1}(y_0 - y_b), \delta y_0\big\rangle + \big\langle \nabla u, \nabla(\delta u)\big\rangle \tag{11.28}$$

with $< \cdot, \cdot >$ the corresponding scalar products.

Since the control variables has here two distinct components, $c = (y_0, u)$, then the gradient of $j(c)$ reads:

$$\boxed{\nabla j(c) = (\nabla_{y_0} j(c), \nabla_u j(c))^T} \tag{11.29}$$

And we have the differentiable which satisfies:

$$\boxed{j'(c) \cdot \delta c = \frac{\partial j}{\partial y_0}(c) \cdot \delta y_0 + \frac{\partial j}{\partial u}(c) \cdot \delta u} \tag{11.30}$$

for any perturbation $\delta c = (\delta y_0, \delta u)$.

Recall that we have: $< \nabla j(c), \delta c >_{\mathbb{R}^m} = j'(c) \cdot \delta c$ for all $\delta c \in \mathcal{C}_h \subset \mathbb{R}^m$.

The Tangent Linear Model (TLM)

By deriving the direct model $(\mathcal{D})$ with respect to c (in a direction δc and at "point" $y(c)$), we obtain the TLM.
The TLM solution is the term $w^{\delta c}$. We have:

$$(\mathcal{LT}) \begin{cases} \text{Given } c = (y_0, u) \in H \times \mathcal{U}_T \text{ and } y^c \in W_V(0,T) \text{ solution of the direct problem } (\\ \text{given } \delta c = (\delta y_0, \delta u) \in H \times \mathcal{U}_T, \text{ find } w^{\delta c} \in W_V(0,T) \text{ such that:} \\ \partial_t w^{\delta c}(t) + \frac{\partial A}{\partial y}(u; y) \cdot w^{\delta c}(t) = -\frac{\partial A}{\partial u}(u; y) \cdot \delta u(t) + F'(u) \cdot \delta u(t) \quad \forall t \in]0, T[\\ w^{\delta c}(0) = \delta y_0 \end{cases} \tag{11.31}$$

Remark 11.3. *If the operator A is linear with respect to the state variable y, then we have:*
$\dfrac{\partial A}{\partial y}(u(t), y(t)) \cdot w^{\delta c}(t) = A(u(t), w^{\delta c}(t))$.
In this case, the TLM is the same as the direct model but the I.C. and the RHS.

As a consequence, to compute $w^{\delta c}$ and to obtain the gradient defined by (11.28), one can solve the TLM $(\mathcal{LT})$.
However, the TLM must be solved again to obtain $w^{\delta c}$ for a different perturbation δc. After discretization, if δc_h is large dimensional, the TLM-based approach is not tractable...

11.3.2 The adjoint-based gradient

Theorem 11.4. *Let us consider the direct model (11.1) and the cost function $j(u)$ defined by (11.6)-(11.27).*
It is assumed that:
i) the direct model (11.1) is well-posed,
ii) the TLM (11.31) is well-posed,
iii) the unique solution $y(c)$ is C^1 with respect to c.

Then, the cost function $j(c)$ is C^1 and its gradient components reads $\nabla j(c) = (\partial_{y_0} j(c), \partial_u j(c))^T$ with:

$$\left\{ \begin{array}{rcl} \partial_{y_0} j(c) & = & p^c(0) \ + \ \alpha_{reg,0}(J^0_{reg})'(y_0) \\ \partial_u j(c) & = & -\left[\dfrac{\partial A}{\partial u}(u; y^c) - F'(u)\right]^T p^c(t) + \alpha_{reg,u}(J^u_{reg})'(u) \end{array} \right. \tag{11.32}$$

with y^c the unique solution of the state equation (11.1) and p^c solution of the adjoint model:

$$(\mathcal{A}) \left\{ \begin{array}{l} \text{Given } c(x,t) = (y_0(x), u(x,t)) \in \mathcal{C} \text{ and } y^c(x,y) \in W_V(0,T) \text{ be the unique soluti} \\ \text{find } p(x,t) \in W_V(0,T) \text{ such that:} \\ -\partial_t\, p(x,t) + \left[\dfrac{\partial A}{\partial y}(u; y^c)\right]^T p(x,t) = J'_{obs}(y^c(x,t)) \quad \forall t \in \,]0,T[\\ p(x,T) \ = \ 0 \text{ in } \Omega \end{array} \right. \tag{11.33}$$

The solution p^c of $(\mathcal{A})$ exists and is unique; it is the adjoint state.

Proof.
Under Assumption iii) (deriving from the implicit function theorem, see details in the stationary case), the cost function $j(u)$ is C^1.
First let us recall the direct expression of $j'(c)$, see (11.6)(11.5):

$$j'(c)\cdot\delta c \;=\; J'_{obs}(y^c)\cdot w^{\delta c} \;+\; J'_{reg}(c)\cdot\delta c \tag{11.34}$$

The goal is to obtain an expression of $j'(c)$ independent of $w^{\delta c}$.

The proof follows the same principles than in the stationary case (see the proof of Theorem 10.7). The only difference consists to integrate in time the equations. We write:

$$\int_0^T < (\mathcal{L}\mathcal{T}), p >_{V'\times V} dt \;=\; 0 \quad \forall p\in V$$

By integrating by part in time, we get:

$$\begin{aligned}
&-\int_0^T \left\langle w^{\delta c}(t), \partial_t p(t)\right\rangle_{V'\times V} dt \;+\; \int_0^T \left\langle \tfrac{\partial A}{\partial y}(u;y)\cdot w^{\delta c}(t), p(t)\right\rangle_{V'\times V} dt \\
&+\left\langle p(T), w^{\delta c}(T)\right\rangle_H \;-\; \left\langle p(0), \delta y_0\right\rangle_H \;+\; \int_0^T \left\langle [\tfrac{\partial A}{\partial u}(u;y) - F'(u)]\cdot\delta u(t), p(t)\right\rangle_{V'\times V} dt \;=
\end{aligned}$$

By making appear the adjoint operator of $\frac{\partial A}{\partial y}(u;y)$, it comes:

$$\begin{aligned}
&\int_0^T \Big\langle -\partial_t p(t) + \left[\frac{\partial A}{\partial y}(u;y)\right]^T p(t), w^{\delta c}(t)\Big\rangle_{V'\times V} dt \\
&= \;-\left\langle p(T), w^{\delta c}(T)\right\rangle_H \;+\; \left\langle p(0), \delta y_0\right\rangle_H \;-\; \int_0^T \left\langle [\tfrac{\partial A}{\partial u}(u;y) - F'(u)]\cdot\delta u, p(t)\right\rangle_{V'\times V} dt
\end{aligned} \tag{11.35}$$

The goal is to make vanish the term $w^{\delta c}(t)$ in (11.34). This goal is reached by setting p as the solution of the following equation:

$$-\partial_t p(t) + \left[\frac{\partial A}{\partial y}(u;y^c)\right]^T p(t) = J'_{obs}(y^c) \quad \forall t\in\,]0,T[\tag{11.36}$$

accompanied with the Initial Condition at final time: $p(T)=0$.
These equations constitute the adjoint model ($\mathcal{A}$).

The linearized problem is supposed to be well-posed therefore the operator $\partial_y A(u; y^u)$ is an isomorphism from V into V', and its adjoint operator $(\partial_y A)^T(u; y^u)$ is an isomorphism from V into V' too, see e.g. [10]. As a consequence, the adjoint equation above is well-posed too.

By making appear the adjoint operator of $[\frac{\partial A}{\partial u}(u; y) - F'(u)]$ and by considering the adjoint equation, 11.35 reads:

$$\int_0^T \left\langle J'_{obs}(y^c), w^{\delta c}(t)\right\rangle_{V'\times V} dt = +\left\langle p(0), \delta y_0\right\rangle_H - \int_0^T \left\langle [\frac{\partial A}{\partial u}(u; y) - F'(u)]^T p(t), \delta u(t)\right\rangle_{\mathcal{U}'\times\mathcal{U}} \tag{11.37}$$

Next, by combining (11.34), (11.37) and (11.36), we obtain:

$$j'(c)\cdot\delta c \;=\; J'_{reg}(c)\cdot\delta c + \left\langle p(0), \delta y_0\right\rangle_H - \int_0^T \left\langle [\frac{\partial A}{\partial u}(u; y) - F'(u)]^T p(t), \delta u(t)\right\rangle_{\mathcal{U}'\times\mathcal{U}} dt \tag{11.38}$$

with (recall) $c(x,t) = (y_0(x), u(x,t))$ and $J'_{reg}(c)\cdot\delta c = \alpha_{reg,0}(J^0_{reg})'(y_0)\cdot\delta y^0 + \alpha_{reg,u}(J^u_{reg})'(u)\cdot\delta u$.

The expression of $j'(c)\cdot\delta c$ above does not depend anymore on $w^{\delta c}$: this was the goal.

Moreover, since:

$$j'(c)\cdot\delta c = \frac{\partial j}{\partial y_0}(c)\cdot\delta y_0 + \frac{\partial j}{\partial u}(c)\cdot\delta u \;, \tag{11.39}$$

the gradient expression (11.32) follows by identification. □

A few remarks

In addition to the remarks already made in the finite dimension linear case, see Section 11.2, let us notice that:

- By construction the adjoint model is linear, whatever if the direct model is linear or not (like in the steady-state case).

- If the differential operator $A(u;y)$ is linear and symmetric (in $y(x,t)$), the problem is self-adjoint (see Section 10.6.2 for details): the adjoint model differs from the direct one by the RHS only.

- By considering the expression of J_{reg} and J_{obs} as defined in Section 11.1.2, we obtain:

$$\left\{ \begin{array}{rcl} \partial_{y_0} j(c) & = & p^c(0) \ + \ \alpha_{reg,0} \ B(y_0 - y_b) \\ \partial_u j(c) & = & - \ \left[\dfrac{\partial A}{\partial u}(u;y^c) - F'(u) \right]^T \ p^c(t) \ + \ \alpha_{reg,u} \ (J^u_{reg})'(u) \end{array} \right. \tag{11.40}$$

Moreover, the RHS $J'_{obs}(y^c(t))$ of the adjoint model equals the data misfit term which reads: $Z^T R^{-1}(Zy(t) - z^{obs}(t))$.

Exercice 11.5. *Consider the time-dependent case of your practical (linear case or not).*
a) Write the adjoint equations; both in the weak and the classical forms.
b) Write the gradient expression based on the adjoint method.

11.4 The 4D-Var algorithm

The algorithm

Given a *first guess* $c^0 \in \mathbb{R}^{n+m}$, compute $c^m \in \mathbb{R}^{n+m}$ making diminish the cost function $j(c)$, $j(c) \in \mathbb{R}$. To do so, at each iteration:

1) compute the cost function $j(c)$ by solving the direct model from 0 to T,

2) compute the gradient $\nabla j(c) \in \mathbb{R}^{n+m}$ by solving the adjoint model from T to 0,

3) given the current iteration c^n, the cost function value $j(c^n)$ and the gradient value $\nabla j(c^n)$, compute a new iterationc^{n+1} such that:

$$j(c^{n+1}) < j(c^n)$$

*) Iterate until convergence.

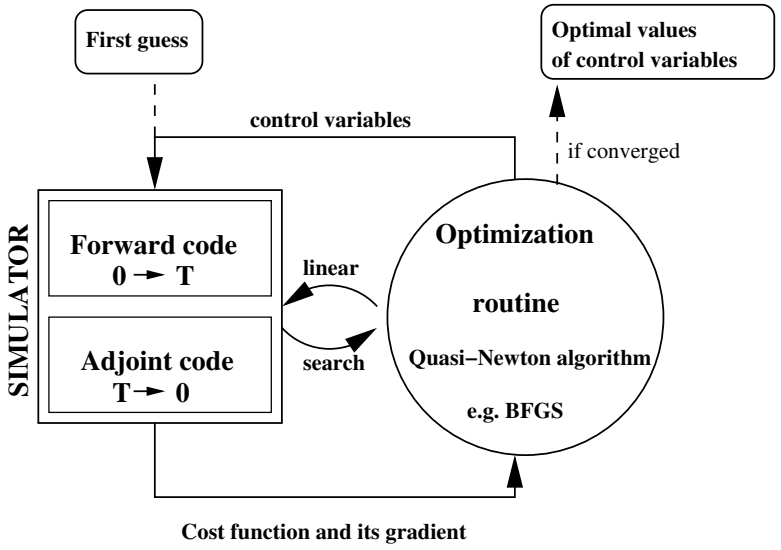


Figure 11.1: Optimal control algorithm for a time-dependent PDE model: the 4D-var algorithm.

A few remarks

Recall that the 4D-Var algorithm can be used for different goals.

A) To estimate the value of uncertain or unknown input parameters (time dependent or not), it is an *identification problem.*

B) To *calibrate* the model in order to perform better predictions; forecasting is the goal.
In this case, the data assimilation proceeds by *"analysis cycles"*.

In this context,

-) the first stage is called the *"analysis step"*.
Observations of present and past time are assimilated to obtain the analysis i.e. the optimal state value (that is the optimal model trajectory).

-) the second stage consists to run the model in time: it is the *"forecasting step"*.

It is expected that if the model fits better the observations in the past, it will be more accurate for future.
Next, the forecast is used in the next analysis cycle etc

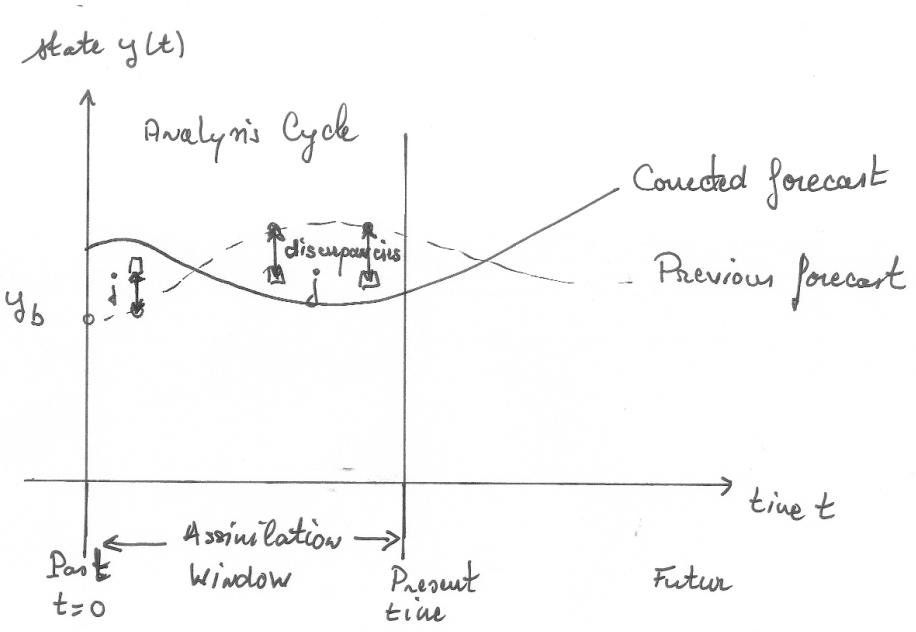


Figure 11.2: The 4D-var algorithm here used to identify the I.C. y_0. All observation within the assimilation time window are assimilated in the global least-square formulation (analysis step). Next, the resulting calibrated model is performed for prediction (forecasting step).

11.5 The fundamental equations at a glance

The **time-dependent (non-linear) PDE model** reads:

$$(\mathcal{D}) \left\{ \begin{array}{l} \text{Given } (y_0(x), u(x,t)), \text{ find } y(x,t) \text{ such that :} \\ \partial_t\, y(x,t) \;+\; A(u(x,t)\,; y(x,t)) \;=\; F(u(x,t)) \qquad \text{in } \Omega\times]0,T[\\ y(x,0) \;=\; y_0(x) \text{ in } \Omega \end{array} \right. \tag{11.41}$$

The direct model is supposed to be well-posed.
The control parameter is: $c(x,t) = (y_0(x), u(x,t))$.
Then, the control-to-state operator (the model operator) $\mathcal{M}(c)$ is defined as: $\mathcal{M}(c) = y^c(x,t)$.
$\mathcal{M}(\cdot)$ is a-priori non linear.

The **objective function** classically reads:

$$J(c\,; y) \;=\; \frac{1}{2}\int_0^T \Big\| \, Z(y(t)) - z^{obs}(t) \Big\|^2_{R^{-1}} \, dt \;+\; \alpha_0 \frac{1}{2} \Big\| \, y_0 - y_b \Big\|^2_{B^{-1}} \;+\; \alpha_u J^u_{reg}(u) \tag{11.42}$$

In discrete form, the observations term $J_{obs}(c\,; y)$ reads: $\frac{1}{2}\sum_n \left(\sum_i Z(y^{obs}(x_i,t_n)) - z^{obs}_{i,n}\right)^2_{R^{-1}}$

The **cost function** is defined as: $j(c) = J(c\,; y^c)$,
where y^c is the (unique) solution of the direct model given c.

The **optimization problem** reads:

$$\left\{ \begin{array}{l} \text{Find } c^*(x,t) = (y_0^*(x), u^*(x,t)) \text{ such that:} \\ j(c^*) = \min\limits_{c \in H \times \mathcal{U}_T} j(c) \end{array} \right. \tag{11.43}$$

The gradient components read: $\nabla j(c) = (\frac{\partial j}{\partial y_0}(c), \frac{\partial j}{\partial u}(c))^T$, with:

$$\left\{ \begin{array}{rcl} \partial_{y_0} j(c(x,t)) & = & p^c(x,0) \; + \; \alpha_0 \; B^{-1}(y_0 - y_b)(x) \\ \partial_u j(c(x,t)) & = & (-[\partial_u A(u; y^c)] + [F'(u)])^T \cdot p^c(x,t) + \; \alpha_u \; (J^u_{reg})'(u(x,t)) \end{array} \right. \tag{11.44}$$

where p^c is the (unique) solution of the **adjoint model**:

$$(\mathcal{A}) \left\{ \begin{array}{l} \text{Given } c = (y_0(x), u(x,t)) \text{ and } y^c(x,t) \text{ the unique solution of the direct problem (} \\ \text{ find } p(x,t) \text{ such that:} \\ -\partial_t\, p(x,t) + \Big[\partial_y A(u(x,t); y^c(x,t))\Big]^T p(x,t) = J'_{obs}(y^c(x,t)) \quad \text{ in } \Omega \times]0,T[\\ p(x,T) \; = \; 0 \text{ in } \Omega \end{array} \right. \tag{11.45}$$

Bibliography

[1] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications*. SIAM, 2016.

[2] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications*. SIAM, 2016.

[3] Richard C Aster, Brian Borchers, and Clifford H Thurber. *Parameter estimation and inverse problems*. Elsevier, 2018.

[4] Combettes P.-L. Bauschke H. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[5] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[6] Leonard David Berkovitz and Negash G Medhin. *Nonlinear optimal control theory*. CRC press, 2012.

[7] Marc Bocquet. *Introduction to the principles and methods of data assimilation in the geosciences*. Lecture notes, Ecole des Ponts ParisTech.

[8] M. Bonavita. Overview of data assimilation methods. ECMWF course online, 2019.

[9] L. Bouttier and P. Courtier. *Data assimilation concepts and methods*. ECMWF Training course. www.ecmwf.int., 1999.

[10] H. Brézis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, 2010.

[11] Edoardo Calvello, Sebastian Reich, and Andrew M Stuart. Ensemble kalman methods: A mean field perspective. *arXiv preprint arXiv:2209.11371*, 2022.

[12] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

[13] Guy Chavent. *Nonlinear least squares for inverse problems: theoretical foundations and step-by-step guide for applications*. Springer Science & Business Media, 2010.

[14] Sibo Cheng, César Quilodrán-Casas, Said Ouala, Alban Farchi, Che Liu, Pierre Tandeo, Ronan Fablet, Didier Lucor, Bertrand Iooss, Julien Brajard, et al. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387, 2023.

[15] P. Courtier and O. Talagrand. Variational assimilation of meteorological observations with the direct adjoint shallow water equations. *Tellus*, 42(A):531–549, 1990.

[16] DassFlow. "data assimilation for free surface flows". Open source computational software: http://www.math.univ-toulouse.fr/DassFlow.

[17] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[18] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[19] Lawrence C. Evans. *An Introduction to Mathematical Optimal Control Theory*. University of California, Berkeley, 2014.

[20] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[21] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[22] David Gilbarg, Neil S Trudinger, David Gilbarg, and NS Trudinger. *Elliptic partial differential equations of second order*, volume 224. Springer, 1977.

[23] Matthew CG Hall and Dan G Cacuci. Physical interpretation of the adjoint functions for sensitivity analysis of atmospheric models. *Journal of the atmospheric sciences*, 40(10):2537–2546, 1983.

[24] Per Christian Hansen. *Discrete inverse problems: insight and algorithms*. SIAM, 2010.

[25] M. Honnorat. Assimilation de données lagrangiennes pour la simulation numérique en hydraulique fluviale. *PhD thesis. Grenoble IT, France*, 2007.

[26] Kayo Ide, Philippe Courtier, Michael Ghil, and Andrew C Lorenc. Unified notation for data assimilation: Operational, sequential and variational (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):181–189, 1997.

[27] Jari Kaipio and Erkki Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006.

[28] Barbara Kaltenbacher, Andreas Neubauer, and Otmar Scherzer. *Iterative regularization methods for nonlinear ill-posed problems*, volume 6. Walter de Gruyter, 2008.

[29] Michel Kern. *Numerical methods for inverse problems*. John Wiley & Sons, 2016.

[30] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Andreas Kirsch. *An introduction to the mathematical theory of inverse problems*, volume 120. Springer Science & Business Media, 2011.

[32] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[33] F.X. Le Dimet and O. Talagrand. Variational algorithms for analysis assimilation of meteorological observations: theoretical aspects. *Tellus A*, 38:97–110, 1986.

[34] J.L. Lions. *Contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles*. Dunod, 1968.

[35] J.L. Lions. *Optimal control of systems governed by partial differential equations*. Springer-Verlag, 1971.

[36] JL Lions and R Dautray. Evolution problems i. mathematical analysis and numerical methods for science and technology, vol. 5, 2000.

[37] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[38] GI Marchuk and VB Zalesny. A numerical technique for geophysical data assimilation problems using pontryagin?s principle and splitting-up method. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 8(4):311–326, 1993.

[39] J. Monnier. Modèles numériques directs et inverses d'écoulements de fluides. *Habilitation à Diriger des Recherches. Institut National Polytechnique de Grenoble (Grenoble IT)*, 2007.

[40] J. Monnier. *Finite Element Methods & Model Reductions*. INSA - University of Toulouse. Open Online Course, 2022.

[41] J. Monnier, K. Larnier, P. Brisset, and F. et al. Couderc. Dassflow (data assimilation for free surface flows): numerical analysis report. Open source computational software: http://www.math.univ-toulouse.fr/DassFlow.

[42] Jennifer L Mueller and Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. SIAM, 2012.

[43] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[44] Yoshikazu Sasaki. An objective analysis based on the variational method. *J. Meteor. Soc. Japan*, 36(3):77–88, 1958.

[45] Yoshikazu Sasaki. Some basic formalisms in numerical variational analysis. *Monthly Weather Review*, 98(12):875–883, 1970.

[46] Olivier Talagrand and Philippe Courtier. Variational assimilation of meteorological observations with the adjoint vorticity equation. i: Theory.

Quarterly Journal of the Royal Meteorological Society, 113(478):1311–1328, 1987.

[47] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

[48] E. Trélat. *Optimal control: theory and applications*. Vuibert, Paris, 2008.

[49] Y. Trémolet. Incremental 4d-var convergence study. *Tellus A*, 59(5):706–718, 2008.

[50] Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[51] Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002.