Annals of Mathematical Sciences and Applications
Volume 9, Number 1, 237–279, 2024

Parameter estimation and control by penalized multiple shooting*

L. Héctor Juárez†, Jorge López‡, and Jessica T. Rojas†

Multiple shooting methods have been studied and applied to parameter estimation and optimal control problems, governed by ODEs during the past decades, and also to problems governed by PDEs less frequently. In this work we explore numerically a multiple shooting method via augmented Lagrangian and penalized models, to estimate parameter values of systems modelled by ODEs. Unlike most authors, who prefer algorithms like Gauss-Newton or SQP to solve the associated optimization problems, we apply the BFGS algorithm with inexact line-search, and a variant of a dual ascent method, along with the adjoint equation method to compute derivatives or gradients. The proposed method, with the mentioned ingredients, is simple and efficient. It estimates accurately parameters of the Lorentz equations in chaotic regime. The same multiple shooting approach can also be applied to optimal control problems, particularly to simultaneously control the transition between equilibrium states and the stabilization around an unstable equilibria of a model that describes the dynamics of a Josephson Junction Array, a quantum interference device used in superconductivity.

AMS 2000 subject classifications: Primary 65K10, 65L09, 90C26, 93-08; secondary 34A55, 34H05, 49M15, 49M37.
Keywords and phrases: Multiple shooting, parameter estimation, optimal control, augmented Lagrangian, quasi-Newton method.

*The penalized approach is obtained from augmented Lagrangian.
†The author is supported by UAM-I and Conacyt-México.
‡The author is supported by UJAT.
†The author is supported by UAM-I and Conacyt-México.

1. Introduction

Systems of ordinary and partial differential equations are an important tool to model the physical state of a real phenomenon that arise in many areas of engineering and applied sciences. Predicting the future behaviour or allowing control of those processes, the design of optimal controls and its practical realization, requires accurate or acceptable solutions and also finding or improving parameter values of the physical model.

Nowadays, multiple shooting, introduced decades ago, is a well-known method. For instance, some early references are [31, 37, 42, 18]. In particular, for parameter estimation it was improved and mathematically analysed in [4, 5]. Since then it has been extensively studied and applied, usually by means of fitting observed noisy data, or experimental measurements, to systems modelled by ODEs [38, 6, 12, 2], and also for solving optimal control problems, [16, 17, 27, 43], and more recently to problems modelled by PDEs, [13, 29, 30, 40]. Two of the most attractive features of multiple shooting methods are its improved stability and convergence, which are obtained in

part from the partition of the objective function along with the subdivision of the time–interval, allowing an efficient minimization of the cost function without getting lost too easily in local minima [4, 5, 38], and also its potential for parallel-in-time implementation [21]. Of course, these advantages can be used for the design of faster and more robust optimization procedures. In fact, the multiple shooting approach has shown to be very effective for parameter estimation and optimal control problems, when gradient based nonlinear programming solvers (NLP) are employed to solve the associated constrained optimization models. For instance, generalized Gauss–Newton methods (perhaps the most used) [4, 6, 2], sequential quadratic programming (SQP) [19, 40, 30] among the Newton or quasi–Newton methods. These NLP solvers need the sensitivity equations (derivatives of the state variables with respect to the parameters), which are useful in finding the gradient of the objective and/or constraint functions. So, the computational cost of these shooting methods depends not only on the efficiency of ODE and sensitivity solvers [4, 2] but also on the solution of middle or large scale linear algebraic problems arising at each iteration that appear with these methods, mainly in real applications.

Here, we want to explore a multiple shooting strategy via an augmented Lagrangian (AL) formulation where the penalized terms are those associated to the equality constraints of the shooting parameters. The gradients of the Lagrangian and of objective functions are computed with the adjoint equation method, allowing the possibility of using gradient descent quasi–Newton methods, like the BFGS algorithm, which do not require the solution of linear systems at each iteration. This methodology is implemented in an unified manner to solve both parameter estimation and optimal control problems. Our first numerical studies about optimal control for an ODE system [24, 32], and about parameter identification for systems of ODE [26], occurred recently, after previous experience with the numerical solution of inverse and optimal control problems modelled by PDE [28, 14], where we applied well known methods and algorithms extensively documented in the celebrated Glowinski's 'red book' [23] and elsewhere. Although all these techniques are well known in the PDE context, for instance [45], we feel that their combination with shooting methods has not been explored exhaustively for ODE systems, perhaps because there are already well established optimization methods, like the generalized Gauss–Newton and SQP, and special techniques to compute derivatives. Recently we were surprised to know that the adjoint equation method together with multiple shooting is seldom used to estimate parameters of ODE systems, according to the authors in [2]. Another motivation is the resurgence of augmented Lagrangian methods in

fields such as total variation denoising and compressed sensing, as well as their somewhat strong connection to ADMM (alternating direction methods of multipliers) [7, 22]. On the other hand, the adjoint equation method has certainly been used successfully to solve optimal control PDE problems, see [25] and some references already mentioned above. Lately, we have found the application of the AL approach to PDE-optimal control problems, [29], showing that this methodology has good properties and it is attractive for potential applications to PDE-based optimization, not only for optimal control, but also for the solution of problems that arise into the context of data driven physical phenomena, like data assimilation, among others.

The proposed methodology in this article is tested with two problems: parameter estimation of the Lorentz equations (fitting an ODE to chaotic data, see [3]) and control and stabilization of a Josephson junction array (JJA). In this last problem, previously studied in recent collaborations with Roland Glowinski [24, 32], we will show that, with multiple shooting, it is possible (and much easier) to control the transition between equilibrium states and stabilize the system around an unstable equilibrium, both actions in a simultaneous way. The organization of the article is the following. In Section 2 we start with some basic notions and continue with the introduction of the multiple shooting and augmented Lagrangian approach for parameter estimation of ODEs. In Section 3 we apply the numerical methodology to parameter estimation of the Lorenz equation, fitting the ODE to chaotic synthetic data. In Section 4 we adapt the proposed methodology to control the transition between equilibria and simultaneous stabilization around an unstable equilibria of a JJA. In Section 5 we present numerical results with piecewise constant or piecewise linear controls, more amenable for practical applications. Finally we give some conclusions and perspectives in Section 6.

2. Multiple shooting for parameter estimation from noisy data

Let $\mathbf{x}(t) \in \mathbb{R}^d$ be a state variable at time $t \in I := [t_0, t_f]$ of a continuous time ordinary differential equation (ODE) satisfying the following initial value problem:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}(t), \boldsymbol{\theta}), \qquad \mathbf{x}(t_0) = \mathbf{s}_0, \quad t_0 < t \leq t_f, \tag{2.1}$$

where $\mathbf{f} : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^{np} \mapsto \mathbb{R}^d$ may not depend on t, but for sure it depends on the state $\mathbf{x}(t)$ and on the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{np}$. When convenient we use the short notation $\mathbf{x}(t) = \mathbf{x}(t; \mathbf{s}_0, \boldsymbol{\theta})$ for simplicity.

In many practical problems not all components of the state $\mathbf{x}(t)$ are observable, and it may be decomposed into observable variables, $\overline{\mathbf{x}}$, and non–observable variables, $\underline{\mathbf{x}}$, which can be regarded as orthogonal projections of $\mathbf{x}$ over the coordinates of these observable and non-observable variables. We assume that we are given experimental noisy measurements at times $t_0 \leq t_1 < \ldots < t_m \leq t_f$:

$$\overline{\mathbf{x}}_i = \overline{\mathbf{x}}(t_i; \mathbf{s}_0, \boldsymbol{\theta}) + \overline{\boldsymbol{\epsilon}}_i, \quad \overline{\boldsymbol{\epsilon}}_i \sim \mathcal{N}(\mathbf{0}, \overline{\Sigma}_i), \quad \overline{\Sigma}_i = diag(\overline{\boldsymbol{\sigma}}_i^2). \tag{2.2}$$

A common model to estimate the unknown parameter $\boldsymbol{\theta}$ and initial conditions $\mathbf{s}_0$, from the given measurements, relays on the minimization of a least squares objective function. Before introducing the multiple shooting approach, in the next subsection we first introduce the objective function and basic notions to compute its gradient with the adjoint equation method.

2.1. A basic non linear optimization model

The parameter estimation of (2.1) consists of identifying $\mathbf{s}_0$ and $\boldsymbol{\theta}$ by solving the optimization problem:

$$\min \; J(\mathbf{s}_0, \boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{m} \left\| \frac{\overline{\mathbf{x}}(t_i; \mathbf{s}_0, \boldsymbol{\theta}) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i} \right\|_{\mathbb{R}^{no}}^2, \tag{2.3}$$

subject to the constraint that the state variable $\mathbf{x}(t) = \mathbf{x}(t, \mathbf{s}_0, \boldsymbol{\theta})$ satisfies the ODE (2.1).

Remark 2.1. *Observe that the quotients in (2.3) are computed componentwise. Formally*

$$||(\overline{\mathbf{x}}(t_i; \mathbf{s}_0, \boldsymbol{\theta}) - \overline{\mathbf{x}}_i)/(\overline{\boldsymbol{\sigma}}_i)||_{\mathbb{R}^{no}}^2 = (\overline{\mathbf{x}}(t_i; \mathbf{s}_0, \boldsymbol{\theta}) - \overline{\mathbf{x}}_i)^T \, \overline{W}_i \, (\overline{\mathbf{x}}(t_i; \mathbf{s}_0, \boldsymbol{\theta}) - \overline{\mathbf{x}}_i),$$

where $\overline{W}_i = \overline{\Sigma}_i^{-1}$ *is the precision matrix.*

Variational approach. We want to employ gradient descent methods or quasi-Newton algorithms, where a critical task is the efficient calculation of the gradient of the objective function. Some options are available to compute these quantities as shown in [10, 11, 41], and references therein. Here, we are interested on an approach based on variational calculus, so we apply the following formal perturbation analysis.

Let $\mathcal{U} = \mathbb{R}^{d+np}$ and $\mathbf{p} = (\mathbf{s}_0, \boldsymbol{\theta})^T \in \mathcal{U}$. Differentiating J at $\mathbf{p}$ with respect to a perturbation $\delta\mathbf{p} = (\delta\mathbf{s}_0, \delta\boldsymbol{\theta})^T$ yields

$$\delta J(\mathbf{p}) = J(\mathbf{p} + \delta\mathbf{p}) - J(\mathbf{p}) = \langle DJ(\mathbf{p}), \delta\mathbf{p} \rangle_{\mathcal{U}} = \nabla J(\mathbf{p}) \cdot \delta\mathbf{p}, \tag{2.4}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{U}}$ denotes the inner product in $\mathcal{U}$ and the dot denotes the usual scalar product, and

(2.5)
$$\nabla J(\mathbf{p}) \cdot \delta\mathbf{p} = \sum_{i=1}^{m} \frac{\overline{\mathbf{x}}(t_i;\mathbf{p}) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^2} \cdot \frac{\partial \overline{\mathbf{x}}(t_i;\mathbf{p})}{\partial \mathbf{p}} \delta\mathbf{p} = \sum_{i=1}^{m} \left(\frac{\partial \overline{\mathbf{x}}(t_i;\mathbf{p})}{\partial \mathbf{p}} \right)^T \frac{\overline{\mathbf{x}}(t_i;\mathbf{p}) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^2} \cdot \delta\mathbf{p}.$$

Then, $DJ(\mathbf{p})$ and $\nabla J(\mathbf{p})$ are related to the Jacobian matrix $\partial \overline{\mathbf{x}}(t_i;\mathbf{p})/\partial \mathbf{p}$. This matrix contains the partial derivatives associated to the sub–Jacobians $\partial \overline{\mathbf{x}}(t_i)/\partial \mathbf{s}_0$ and $\partial \overline{\mathbf{x}}(t_i)/\partial \boldsymbol{\theta}$. These partial derivatives are known in the engineering community as the *sensitivities*, and they may be computed directly. Computing these sensitivities is the most delicate task, and it may be expensive. The most basic method to compute these derivatives is the finite difference method, see [41]. Other choice is the forward variational approach where, besides the solution of the state equation, we must solve two forward matricial dynamical systems. Here, we concentrate on the adjoint equation method, which consists in solving only the state equation and the so called adjoint equation, without computing explicitly the sensitivities. To do this, we rewrite (2.5) by lumping together the sensitivities,

$$\delta\mathbf{x}(t) = \frac{\partial \mathbf{x}(t)}{\partial \mathbf{p}} \delta\mathbf{p} = \frac{\partial \mathbf{x}(t)}{\partial \mathbf{s}_0} \delta\mathbf{s}_0 + \frac{\partial \mathbf{x}(t)}{\partial \boldsymbol{\theta}} \delta\boldsymbol{\theta},$$

obtaining

(2.6)
$$\nabla J(\mathbf{p}) \cdot \delta\mathbf{p} = \sum_{i=1}^{m} \frac{\overline{\mathbf{x}}(t_i;\mathbf{p}) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^2} \cdot \delta\overline{\mathbf{x}}(t_i) = \int_{t_0}^{t_f} \sum_{i=1}^{m} \frac{\overline{\mathbf{x}}(t;\mathbf{p}) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^2} \delta_D(t - t_i) \cdot \delta\overline{\mathbf{x}}(t)\, dt,$$

where δ_D is the Dirac measure centered at zero.

Lagrangian, adjoint equation and the gradient. The Lagrangian for the constrained optimization problem (2.3) is

$$\mathcal{L}(\mathbf{p}, \mathbf{y}) = J(\mathbf{p}) + \int_{t_0}^{t_f} \mathbf{y}(t) \cdot \{\, \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}) - \dot{\mathbf{x}}(t) \,\}\, dt, \tag{2.7}$$

where $\dot{\mathbf{x}} = dx/dt$, and $\mathbf{y} \in (L^2(I))^d$ is the Lagrange multiplier. If $\mathbf{x} \in (H^1(I))^d$ solves the state equation (2.1) then $\nabla_{\mathbf{p}} \mathcal{L}(\mathbf{p}, \mathbf{y}) = \nabla J(\mathbf{p})$ for all $\mathbf{y}$. So, if we perturb θ, the state $\mathbf{x}$ and its deviation $\delta\mathbf{x}$ satisfy

$$\int_{t_0}^{t_f} \mathbf{y}(t) \cdot \delta\dot{\mathbf{x}}(t)\, dt = \int_{t_0}^{t_f} \mathbf{y}(t) \cdot \{\, \mathbf{f}_{\mathbf{x}}(\mathbf{x}(t), \boldsymbol{\theta})\, \delta\mathbf{x}(t) + \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t), \boldsymbol{\theta})\, \delta\boldsymbol{\theta} \,\}\, dt.$$

Doing integration by parts, and after some algebraic operations, we get

$$\mathbf{y}(t)\cdot\delta\mathbf{x}(t)\big|_{t_0}^{t_f}-\int_{t_0}^{t_f}\left\{\dot{\mathbf{y}}+\mathbf{f}_{\mathbf{x}}^{T}\,\mathbf{y}\right\}(t)\cdot\delta\mathbf{x}(t)\,dt=\int_{t_0}^{t_f}\left\{\mathbf{f}_{\boldsymbol{\theta}}^{T}\,\mathbf{y}\right\}(t)\,dt\cdot\delta\boldsymbol{\theta}. \tag{2.8}$$

Now, we choose $\mathbf{y}(t)$ as the solution of the following, backward in time, *adjoint differential equation*:

$$\begin{cases}-\dot{\mathbf{y}}(t)=\mathbf{f}_{\mathbf{x}}(\mathbf{x}(t),\boldsymbol{\theta})^{T}\,\mathbf{y}(t)+\displaystyle\sum_{i=1}^{m}\frac{\overline{\mathbf{x}}(t_i;\mathbf{p})-\overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^{2}}\,\boldsymbol{\delta}_{D}(t-t_i),\\ \mathbf{y}(t_f)=\mathbf{0}.\end{cases} \tag{2.9}$$

Then (2.8) becomes

$$\sum_{i=1}^{m}\frac{\overline{\mathbf{x}}(t_i;\mathbf{p})-\overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^{2}}\cdot\delta\overline{\mathbf{x}}(t_i)\quad=\quad\mathbf{y}(t_0)\cdot\delta\mathbf{s}_0+\int_{t_0}^{t_f}\left\{\mathbf{f}_{\boldsymbol{\theta}}^{T}\,\mathbf{y}\right\}(t)\,dt\cdot\delta\boldsymbol{\theta}, \tag{2.10}$$

where we have used that $\delta\mathbf{x}(t_0)=\delta\mathbf{s}_0$. The gradient is obtained from equations (2.6) and (2.10):

$$\nabla_{\mathbf{p}}\mathcal{L}(\mathbf{p},\mathbf{y})=\nabla J(\mathbf{p})=\begin{bmatrix}\nabla_{\mathbf{s}_0}J(\mathbf{p})\\ \nabla_{\boldsymbol{\theta}}J(\mathbf{p})\end{bmatrix}=\begin{bmatrix}\mathbf{y}(t_0)\\ \displaystyle\int_{t_0}^{t_f}\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}(t),\boldsymbol{\theta})^{T}\,\mathbf{y}(t)\,dt\end{bmatrix}, \tag{2.11}$$

where $\mathbf{x}(t)$ solves the state equation (2.1) and $\mathbf{y}(t)$ solves the adjoint equation (2.9).

Having this efficient way to compute the gradient, we may use descent gradient methods to solve the optimization problem. In [26] a detailed comparison between the conjugate gradient (CG) and BFGS algorithms is presented, for parameter identification of the SEIRD epidemiological model [39], using the secant method for line search. There, it is shown that similar numerical results are obtained with both methods, under the same tolerance to achieve a given accuracy, but the BFGS algorithm is more efficient and robust. The main drawback of the proposed methodology is that it is sensitive to time location of the data and noise level, and also to the initial guesses for the initial conditions, so the CG and BFGS algorithms not always converge, or converge to a local optimum. The first problem (sensitivity to data and their location) is because the adjoint equation may have discontinuous (or high gradient) solutions, since data arises as instantaneous pulses in (2.9). The lack of stability on the estimation of initial conditions

is related to the sensitivity $||\mathbf{x}(t;\mathbf{s}_0) - \mathbf{x}(t;\mathbf{s}_0 + \delta\mathbf{s}_0)|| \leq e^{L(t-t_0)}||\delta\mathbf{s}_0||$, where L is a Lipschitz constant of the flow $\mathbf{f}$ in (2.1) with respect to $\mathbf{x}$. Of course, the convergence properties of the optimization algorithms improve when the initial conditions are known.

2.2. Multiple shooting and augmented Lagrangian

The inherent instability and difficulty to find the initial conditions, as well as the effect of data in long time intervals, may be relaxed incorporating the technique of multiple shooting, e.g. [2, 12]. The idea about multiple shooting is dividing the time interval $[t_0, t_f]$ into ns smaller subintervals $I_j = [\tau_j, \tau_{j+1}]$, $j = 1, \ldots, ns$, and splitting the cost function over each subinterval. We guess an unknown 'initial condition' $\mathbf{s}_j$ and a 'target' $\mathbf{s}_{j+1}$ in each subinterval (see Figure 1).

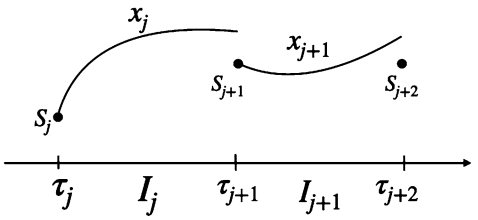

Figure 1: Shooting times and nodes for two typical subintervals of $[t_0, t_f]$.

With this approach, we introduce the shooting times $\tau_1, \tau_2, \ldots, \tau_{ns+1}$, and the associated shooting vector parameters $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{ns+1}$, obtaining a new parameter vector $\mathbf{p} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{ns+1}, \boldsymbol{\theta})^T$ in $\mathbb{R}^{d\times(ns+1)+np}$. For simplicity, we assume that each shooting subinterval I_j contains at least one experimental measurement t_i, and those t_i not necessarily match the shooting times τ_j. The new optimization model is:

$$\min_{\mathbf{p}} \; J(\mathbf{p}) := \sum_{j=1}^{ns} L_j(\mathbf{p}_j) \quad \text{with} \quad L_j(\mathbf{p}_j) = \frac{1}{2} \sum_{t_i \in I_j} \left\| \frac{\overline{\mathbf{x}_j}(t_i) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i} \right\|^2_{\mathbb{R}^{no}}, \tag{2.12}$$

subject to the following shooting constraints for each $\mathbf{x}_j(t)$, $j = 1, \ldots, ns$:

$$\begin{cases} \dot{\mathbf{x}}_j(t) = \mathbf{f}(\mathbf{x}_j(t), \boldsymbol{\theta}), & \tau_j < t \leq \tau_{j+1}, \\ \mathbf{x}_j(\tau_j) = \mathbf{s}_j, \end{cases} \tag{2.13}$$

$$\mathbf{x}_j(\tau_{j+1}) = \mathbf{s}_{j+1}. \tag{2.14}$$

The first constraint ensures that $\mathbf{x}_j$ satisfies the state equation in the local time-interval I_j, while the second one ensures continuity (or matching) conditions between solutions at consecutive intervals (or interfaces).

For the above constrained optimization problems let $\mathbf{y}_j \in (H^1(I_j))^d$ and $\boldsymbol{\lambda}_j \in \mathbb{R}^d$, the Lagrange multipliers associated to the constraints (2.13) and (2.14), respectively, for $j = 1, \ldots, ns$. With the notation $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{ns})^T$, $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{ns})^T$, $\mathbf{k} = (\mathbf{k}_1, \ldots, \mathbf{k}_{ns})^T$, the associated augmented Lagrangian is

$$\begin{aligned}(2.15)\quad \mathcal{L}_{\mathbf{k}}(\mathbf{p}, \mathbf{y}, \boldsymbol{\lambda}) = &\sum_{j=1}^{ns} L_j(\mathbf{p}_j) + \sum_{j=1}^{ns} \int_{I_j} \mathbf{y}_j \cdot \{\mathbf{f}(\mathbf{x}_j(t), \boldsymbol{\theta}) - \dot{\mathbf{x}}_j(t)\}\, dt \\ &+ \sum_{j=1}^{ns} \left\{ \boldsymbol{\lambda}_j \cdot [\mathbf{x}(\tau_{j+1}) - \mathbf{s}_{j+1}] + \frac{k_j}{2} \|\mathbf{x}(\tau_{j+1}) - \mathbf{s}_{j+1}\|^2_{\mathbb{R}^d} \right\}.\end{aligned}$$

The penalization parameters k_j are positive scalars. Doing a perturbation analysis, like in the previous section, yields

$$\begin{aligned}(2.16)\quad \nabla_{\mathbf{p}} \mathcal{L}_{\mathbf{k}} \cdot \delta\mathbf{p} = &\sum_{j=1}^{ns} \nabla L_j(\mathbf{p}_j) \cdot \delta\mathbf{p}_j + \sum_{j=1}^{ns} \int_{I_j} \mathbf{y}_j \cdot [\delta\mathbf{f}(\mathbf{x}_j, \boldsymbol{\theta}) - \delta\dot{\mathbf{x}}_j]\, dt \\ &+ \sum_{j=1}^{ns} \{\boldsymbol{\lambda}_j + k_j [\mathbf{x}_j(\tau_{j+1}) - \mathbf{s}_{j+1}]\} \cdot (\delta\mathbf{x}_j(\tau_{j+1}) - \delta\mathbf{s}_{j+1}),\end{aligned}$$

where

$$\nabla L_j(\mathbf{p}_j) \cdot \delta\mathbf{p}_j = \sum_{t_i \in I_j} \frac{\overline{\mathbf{x}}_j(t_i) - \overline{\mathbf{x}}_i}{\overline{\boldsymbol{\sigma}}_i^2} \cdot \delta\overline{\mathbf{x}}_j(t_i),$$

with $\delta\overline{\mathbf{x}}_j(t_i) = \frac{\partial \overline{\mathbf{x}}_j(t_i)}{\partial \mathbf{s}_j} \delta\mathbf{s}_j + \frac{\partial \overline{\mathbf{x}}(t_i)}{\partial \mathbf{s}_{j+1}} \delta\mathbf{s}_{j+1} + \frac{\partial \overline{\mathbf{x}}_j(t_i}{\partial \boldsymbol{\theta}} \delta\boldsymbol{\theta}$.

2.3. Adjoint equations and gradient of the objective function

Those terms in (2.16) associated to $\delta\mathbf{x}_j(\tau_{j+1})$ are incorporated as final conditions of the adjoint equations. Therefore, after doing the appropriate algebraic manipulation, for $j = 1, \ldots, ns$ we get the following set of adjoint equations in each subinterval I_j:

(2.17)

$$\begin{cases} -\dot{\mathbf{y}}_j(t) = \mathbf{f}_{\mathbf{x}}(\mathbf{x}_j(t), \boldsymbol{\theta})^T \mathbf{y}_j(t) + \displaystyle\sum_{t_i \in I_j} \frac{\overline{\mathbf{x}}_j(t_i) - \overline{\mathbf{x}}_i)}{\overline{\boldsymbol{\sigma}}_i^2} \delta_{D}(t - t_i), \quad \tau_{j+1} > t \geq \tau_j, \\ \mathbf{y}_j(\tau_{j+1}) = \boldsymbol{\lambda}_j + k_j [\mathbf{x}_j(\tau_{j+1}) - \mathbf{s}_{j+1}]. \end{cases}$$

Likewise, the last terms in (2.16), but now associated to $\delta \mathbf{s}_{j+1}$, are included directly into the gradient. Therefore, assuming that $\mathbf{x}_j(t)$ satisfies the state equation (2.13) in I_j, and $\mathbf{y}_j(t)$ satisfies the corresponding adjoint equation (2.17), we obtain the gradient:

$$\nabla_{\mathbf{p}} \mathcal{L}_{\mathbf{k}} = \nabla J(\mathbf{p}) = \begin{bmatrix} \mathbf{y}_1(\tau_1) \\ \mathbf{y}_2(\tau_2) - \mathbf{r}_1 \\ \vdots \\ \mathbf{y}_{j+1}(\tau_{j+1}) - \mathbf{r}_j \\ \vdots \\ \mathbf{y}_{ns}(\tau_{ns}) - \mathbf{r}_{ns-1} \\ \\ -\mathbf{r}_{ns} \\ \\ \displaystyle\sum_{j=1}^{ns} \int_{\tau_j}^{\tau_{j+1}} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_j(t), \boldsymbol{\theta})^T \, \mathbf{y}_j(t) \, dt \end{bmatrix} \tag{2.18}$$

where $\mathbf{r}_j = \boldsymbol{\lambda}_j + k_j \left[\mathbf{x}_j(\tau_{j+1}) - \mathbf{s}_{j+1} \right]$ for $j = 1, \ldots, ns$.

2.4. Optimization algorithms

The state and adjoint variables, used in the gradient calculation, are computed independently in each time window I_j, allowing for an easy parallel implementation. Setting $\boldsymbol{\lambda}_j = \mathbf{0}$ for $j = 1, \ldots, ns$, the BFGS algorithm (Algorithm 1) may be applied directly to the resultant pure penalized model with constant penalization parameters. The most critical step in Algorithm 1 is the solution of the one dimensional optimization problem at step 2, which is not a trivial step and requires careful treatment. There are several options, the most common are line search methods and trust region methods. Some classic references are [15, 35], or the more recent one [44], while some publications, e,g, [1, 36], show that this topic is still under development. Given that we have an efficient way to compute the derivative $\varphi'(\rho) = \nabla J(\mathbf{p}^\ell + \rho \, \mathbf{d}^\ell) \cdot \mathbf{d}^\ell$, we chose the secant method for line search, whose iteration formula is

$$\rho_{k+1} = \rho_k - \frac{\varphi'(\rho_k)(\rho_k - \rho_{k-1})}{\varphi'(\rho_k) - \varphi'(\rho_{k-1})} = \frac{\rho_{k-1}\varphi'(\rho_k) - \rho_k \varphi'(\rho_{k-1})}{\varphi'(\rho_k) - \varphi'(\rho_{k-1})}, \quad k = 1, 2, \ldots \tag{2.19}$$

requiring two initial values ρ_0 and ρ_1. For $\ell \geq 1$, we set $\rho_0 = 0$ and $\rho_1 = \mathbf{g}^\ell \cdot (\mathbf{p}^\ell - \mathbf{p}^{\ell-1}) / \mathbf{g}^\ell \cdot \mathbf{d}^\ell$. For $\ell = 0$, we also choose $\rho_0 = 0$, but this time $\rho_1 = -\varepsilon \, \mathbf{g}^0 \cdot$

Algorithm 1 BFGS algorithm

INPUT: $\mathbf{p}^0 = (\mathbf{s}_1^0, \ldots, \mathbf{s}_{ns+1}^0, \boldsymbol{\theta}^0)^T$ and $0 < \epsilon < 1$ (tolerance to stop iterations).
OUTPUT: optimal value $\mathbf{p}^* = (\mathbf{s}_1^*, \ldots, \mathbf{s}_{ns+1}^*, \boldsymbol{\theta}^*)^T$).

Initialization

1. Hessian $H^0 = I$, gradient $\mathbf{g}^0 = \nabla J(\mathbf{p}^0)$, direction $\mathbf{d}^0 = -\mathbf{g}^0$.

Descent

For $\ell \geq 0$, given $\mathbf{p}^\ell$, $\mathbf{g}^\ell$, $\mathbf{d}^\ell$, H^ℓ find $\mathbf{p}^{\ell+1}$, $\mathbf{g}^{\ell+1}$, $\mathbf{d}^{\ell+1}$, $H^{\ell+1}$, doing the following:

2. Find $\rho_\ell = \arg\min_{\rho \geq 0} \varphi(\rho) = J(\mathbf{p}^\ell + \rho\, \mathbf{d}^\ell)$.
3. Update $\mathbf{p}^{\ell+1} = \mathbf{p}^\ell + \rho_\ell\, \mathbf{d}^\ell$ and $\mathbf{g}^{\ell+1} = \nabla J(\mathbf{p}^{\ell+1})$.

Convergence test and new direction

if $\|\mathbf{g}^{\ell+1}\| \leq \epsilon \|\mathbf{g}^0\|$ **then**

4. Set $\mathbf{p}^* = \mathbf{p}^{\ell+1}$, stop and exit.

else

5. Update $H^{\ell+1} = \left(I - \dfrac{\mathbf{v}\mathbf{u}^T}{\mathbf{u}^T\mathbf{v}}\right) H^\ell \left(I - \dfrac{\mathbf{u}\mathbf{v}^T}{\mathbf{u}^T\mathbf{v}}\right) + \dfrac{\mathbf{v}\mathbf{v}^T}{\mathbf{u}^T\mathbf{v}}$ with $\begin{array}{l}\mathbf{u} = \mathbf{p}^{\ell+1} - \mathbf{p}^\ell \\ \mathbf{v} = \mathbf{g}^{\ell+1} - \mathbf{g}^\ell\end{array}$.
6. Update $\mathbf{d}^{\ell+1} = -H^{\ell+1}\mathbf{g}^{\ell+1}$.
7. Do $\ell = \ell + 1$ and go back to 2.

end if

$\mathbf{p}^0/\mathbf{g}^0 \cdot \mathbf{d}^0$, with $\varepsilon < 1$, has shown appropriate in practice. These elections take into account a proper scaling (see step 3). We want to emphasize that in practice inexact line search is applied and, in our numerical experiments, we found that it is enough to do at most three secant iterations.

Remark 2.2. *We have not employed limited memory quasi-Newton implementations, like L-BFGS [33, 34], and their variants, mainly because the number of variables of the objective functions for the examples in this paper is small. The most expensive numerical experiment in terms of CPU time, from the examples presented bellow, takes very few seconds, and it is enough to consider the above basic BFGS algorithm. However, for large scale problems, like those arising in parameter estimation and/or control of problems modelled by PDEs, it is necessary, actually mandatory, to implement memory saving techniques. Some advantages of limited memory quasi-Newton methods are that: the amount of storage required by the algorithms, and thus the cost of the iteration, can be controlled by the user, these methods are simple to program, they require substantially fewer function evaluations, and they do not require additional information than the one required by the basic quasi-Newton methods, see [34, 9] and references therein.*

If we want to estimate the Lagrange multipliers $\boldsymbol{\lambda}_j$, and update the penalization parameters k_j, we can apply Algorithm 2 (bellow) which is a

variant of the dual ascent and method of multipliers adapted from [7], see also [35], [29] and references therein. In this algorithm we use the notation $\mathbf{g}^\ell = \nabla_{\mathbf{p}}\mathcal{L}_{\mathbf{k}}(\mathbf{p}^\ell, \mathbf{y}^\ell, \boldsymbol{\lambda}^\ell, \mathbf{k}^\ell)$, and $\mathbf{p}^\ell, \boldsymbol{\lambda}^\ell, \mathbf{k}^\ell \quad \rightsquigarrow \quad \mathbf{y}^\ell$ to indicate that given $\mathbf{p}^\ell = (\mathbf{s}_1^\ell, \ldots, \mathbf{s}_{ns+1}^\ell, \boldsymbol{\theta}^\ell)$, we obtain $\mathbf{x}_j^\ell$ solving (2.13), and then we obtain $\mathbf{y}_j^\ell$ solving (2.17) for each $j = 1, \ldots, ns$. Since the parameters k_j penalize the term associated to the continuity constraints (2.14), we propose the following updating formula for each $j = 1, \ldots, ns$:

$$k_j^\ell = \max\left\{k_j^{\ell-1}, \ \frac{\|\mathbf{g}_{j+1}^\ell\|}{\|\mathbf{x}_j^\ell(\tau_{j+1}) - \mathbf{s}_{j+1}^\ell\|}\right\} \quad \text{with} \quad 0 < k_j^0 \leq 1, \tag{2.20}$$

where $\mathbf{g}_{j+1}^\ell$ is the $(j+1)$–th vector component of the gradient (2.18) evaluated at iteration ℓ. With this updating formula it is easy to see that k_j is incremented only when $\mathbf{g}_{j+1}^\ell$ does not decrease proportionally to the term $k_j^{\ell-1}\,[\mathbf{x}_j^\ell(\tau_{j+1}) - \mathbf{s}_{j+1}^\ell]$.

Concerning the convergence tolerances ϵ_ℓ at step 5 of Algorithm 2, we employ the following variant of the one employed in [29], and justified in [20]:

$$\epsilon_\ell = \max\left\{\epsilon, \ \frac{a_\ell}{\|\mathbf{k}^\ell\|^b}\right\} \quad \text{with} \quad a_\ell = \frac{\|\mathbf{g}^\ell\|}{\|\mathbf{g}^0\|} \quad \text{and} \quad 1 \leq b < 2. \tag{2.21}$$

Observe that $0 < a_\ell < 1$ and $a_\ell \to 0$ at convergence, so that $\epsilon_\ell\|\mathbf{k}^\ell\| \to 0$ as well, i.e. $\epsilon_\ell = o(1/\|\mathbf{k}^\ell\|)$, as recommended in [20, 29]. In numerical calculations we fix $b = 1.05$.

From now on we will call Algorithm 2 simply as AL. Likewise, we will call BFGS to Algorithm 1 when it is applied to the pure penalized model with penalty parameters k_j fixed (equivalently when step 2 in Algorithm 2 is solved once with $\boldsymbol{\lambda}^0 = \mathbf{0}$ and $\mathbf{k}^0$ fixed).

Remark 2.3. *A complete study include a theoretical or formal argumentation about convergence to feasible points of these algorithms. This is an open pending issue, which we will consider later. We may look at local in time properties by linearisation around* $\mathbf{s}_j$ *and* $\boldsymbol{\theta}$ *on each shooting interval* I_j*, hoping to get some convexity information, in these coordinate directions, of the quadratic-linear optimization model. Since the role of* k_j *is enforcing continuity conditions on the shooting parameters* $\mathbf{s}_j$*, and these penalty parameters represent the relative importance between the cost function and the difference* $||\mathbf{x}_j(\tau_{j+1} - \mathbf{s}_{j+1})||$ *at the end of* I_j *in a penalized model (i.e. with* $\boldsymbol{\lambda}_j = 0$*), then reducing the gradient in the direction of* $\boldsymbol{\theta}$ *may be more*

Algorithm 2 AUGMENTED LAGRANGIAN (AL) algorithm

INPUT: $\widehat{\mathbf{p}} = (\widehat{\mathbf{s}}_1, \widehat{\mathbf{s}}_2, \ldots, \widehat{\mathbf{s}}_{ns+1}, \widehat{\boldsymbol{\theta}})^T$ and $0 < \epsilon < 1$ (tolerance to stop iterations).
OUTPUT: Optimal parameter value $\mathbf{p}^* = (\mathbf{s}_1^*, \mathbf{s}_2^*, \ldots, \mathbf{s}_{ns+1}^*, \boldsymbol{\theta}^*)^T$.

Initialization

1. $\mathbf{p}^0 = \widehat{\mathbf{p}}$, $\boldsymbol{\lambda}^0 = \mathbf{0} \;\rightsquigarrow\; \mathbf{y}^0$, $\mathbf{g}^0 := \nabla_{\mathbf{p}} \mathcal{L}_{\mathbf{k}}(\mathbf{p}^0, \mathbf{y}^0, \boldsymbol{\lambda}^0)$ with $0 < \|\mathbf{k}^0\| \leq \mathcal{O}(1)$.

Descent

For $\ell \geq 1$, given $\mathbf{p}^{\ell-1}$, $\boldsymbol{\lambda}^{\ell-1}$, $\mathbf{k}^{\ell-1} \;\rightsquigarrow\; \mathbf{y}^{\ell-1}$, find $\mathbf{p}^\ell$, $\boldsymbol{\lambda}^\ell$, $\mathbf{k}^\ell \;\rightsquigarrow\; \mathbf{y}^\ell$, doing the following:

2. Obtain $\mathbf{p}^\ell = \arg\min_{\mathbf{p}} \mathcal{L}_{\mathbf{k}^{\ell-1}}(\mathbf{p}, \mathbf{y}^{\ell-1}, \boldsymbol{\lambda}^{\ell-1})$. (BFGS algorithm)
3. Update $\boldsymbol{\lambda}_j^\ell = \boldsymbol{\lambda}_j^{\ell-1} + k_j^{\ell-1} [\mathbf{x}_j^\ell(\tau_{j+1}) - \mathbf{s}_{j+1}^\ell] \;\rightsquigarrow\; \mathbf{y}_j^\ell, \quad j = 1, \ldots, ns.$
4. Update $k_j^\ell \geq k_j^{\ell-1}, \quad j = 1, \ldots, ns.$ (see eq. 2.20)
5. Update convergence tolerance $\epsilon_\ell \leq \epsilon_{\ell-1}$. (see eq. 2.21)

Convergence test

if $\epsilon_\ell < \epsilon$ **then**

6. Set $\mathbf{p}^* = \mathbf{p}^\ell$, stop and exit.

else

7. Update for next iteration: $\ell = \ell + 1$, and go back to step 2.

end if

significant overall, and setting k_j with constant values give meaning results, according to the numerical experiments. But, we must understand more the role of $\boldsymbol{\theta}$ and its relation with the penalized terms.

Finally, we want to emphasize that whereas solutions remain time-continuous in the single shooting approach, in multiple shooting, they are allowed to be discontinuous during the optimization process, and matching conditions between windows only need to be satisfied upon convergence.

3. Numerical examples for parameter estimation

To validate the fitting model and the proposed optimization algorithms we consider the problems of estimating the parameters of the Lorenz equations in the chaotic regime. In our opinion, this problem is a good benchmark to test the multiple shooting approach for parameter estimation in chaotic ODEs in general, assuming that we have enough information (data) to reflect all relevant frequencies occurring, see [3]. The Lorenz equations are a simplified model of thermal convection in a box, given by

$$\frac{dx}{dt} = \sigma\,(y - x), \tag{3.1}$$

$$\frac{dy}{dt} = x\,(\rho - z) - y, \tag{3.2}$$

$$\frac{dz}{dt} = x\,y - \beta\,z, \tag{3.3}$$

where the state variables are the rate of convective overturning $x(t)$, the horizontal temperature difference $y(t)$, and departure from vertical temperature gradient $z(t)$. The parameters are: the Prandtl number σ, the Rayleigh number ρ and an aspect ratio β. For $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, Lorenz (1963) suggested that trajectories in a bounded region converge to an attractor that is a fractal, with dimension about 2.06, as estimated by Liapunov exponents. Figure (2) shows the numerical solution obtained with the standard fourth-order Runge-Kutta method (RK4) in the interval $[t_0, t_f] = [0, 30]$ with time step $h = 0.01$, and initial conditions $\mathbf{x}(t_0) = (x(t_0), y(t_0), z(t_0))^T = (14, 45, 11)^T$.

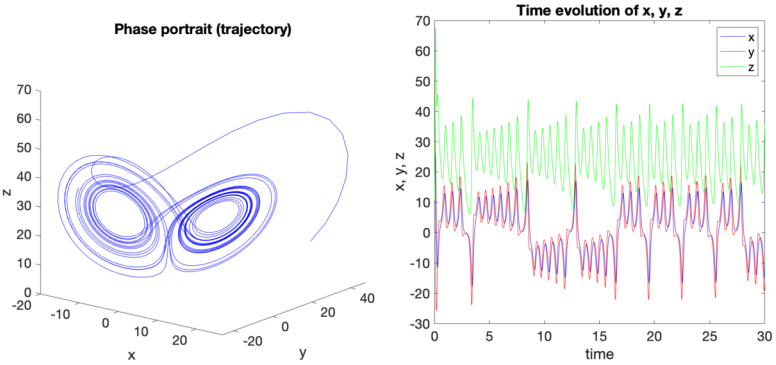

Figure 2: Phase portrait of the solution and time-evolution of x, y, z.

We take this solution as our reference true solution. Synthetic data is generated adding white noise to this solution, using the Matlab random Gaussian generator with zero mean and standard deviations $\boldsymbol{\sigma}_i = noise\_level \times \mathbf{x}(t_i))$, where $\mathbf{x}(t_i) = (x(t_i), y(t_i), z(t_i))^T$, at a series of instant times t_i in the short interval $[t_0, t_f] = [0, 3]$. The proposed model and the numerical algorithm are tested for several noise levels, for instance $0.1, 0.2, 0.3, 0.5$ (i.e. adding 10%, 20%, 30% and 50% noise to the true solution). Then, we hope to recover the initial conditions and an approximation to the exact parameter vector $\boldsymbol{\theta} = (\sigma, \rho, \beta)^T = (10, 28, 10/3)^T$. All numerical calculation were done using the Matlab environment with our own computational programs, using a MacBook Pro (2019) with an Intel core i5 processor, five kernels and 16 GB of RAM. Standard fourth order Runge Kutta (RK4) solvers are employed for the solution of the state and adjoint equations at each iteration.

Example 3.1. *We begin considering perturbed data at 20% in the time interval* $[0,3]$ *with Gaussian noise (noise_level* $= 0.2$*), generated with Matlab from the 'exact solution' with the given initial conditions (see Figure 2). So, we obtain the time-set* $t_i = i\,h$, $i = 0,1,\ldots,300$ *with* $h = 0.01$ *and synthetic data* $\{\overline{\mathbf{x}}_i\}_{i=1}^m = \{(x_i,y_i,z_i)^T\}_{i=1}^m$ *with* $m = 301$*. We divide the time interval into* $ns = 150$ *subintervals for best results. The initial guesses to start iterations, in either Algorithm 1 or 2, are* $\boldsymbol{\theta}^0 = (5,15,5)^T$ *and* $\mathbf{s}_j^0$ *equal to the values of the synthetic perturbed data for all* j*. We choose* $k_j^0 = 1$ *for* $1 \le j \le ns$*.*

Table 1 summarizes the numerical results obtained with BFGS and AL algorithms for different combinations of the observable state variables. The following notation is used: Obs* indicate which variables are observable in each experiment, ALG* = AL or BFGS, ϵ = tolerance to stop AL or BFGS, Iters* = number of iterations to achieve convergence to the given tolerance, one number for the BFGS and two for AL (the first one being the number of AL iterations ℓ, and the second one is the total cumulative number of BFGS iterations). Last two columns show the computed value of the parameters and their relative difference with respect to the exact value, respectively.

Table 1: Numerical results obtained with BFGS and AL algorithms for different observable state variables. Initial guess is $\theta^0 = (5,15,5)^T$, *noise_level* $=$ 0.2, and $ns = 150$

Obs*	ALG*	ϵ	Iters*	Computed $\sigma^*, \rho^*, \beta^*$	Relative difference: σ, ρ, β
x	BFGS	10^{-3}	57	10.0636, 27.5455, 2.6777	0.0064, 0.0162, 0.0041
x	AL	10^{-5}	2,58	10.0876, 27.6121, 2.6775	0.0088, 0.0139, 0.0041
y	BFGS	1.4×10^{-3}	70	10.1831, 27.8341, 2.7648	0.0183, 0.0059, 0.0368
y	AL	10^{-4}	3,78	10.0312, 27.9439, 2.7536	0.0031, 0.0020, 0.0326
z	BFGS	10^{-3}	51	10.0384, 27.5645, 2.6796	0.0038, 0.0156, 0.0048
z	AL	10^{-4}	3,57	10.0301, 27.5405, 2.6464	0.0030, 0.0164, 0.0076
x, y	BFGS	10^{-3}	63	10.0986, 27.7424, 2.7462	0.0099, 0.0092, 0.0298
x, y	BFGS	10^{-4}	199	9.8304, 27.5918, 2.7434	0.0170, 0.0146, 0.0288
x, y	AL	10^{-5}	4, 88	10.1105, 27.8548, 2.7420	0.0110, 0.0052, 0.0282
x, z	BFGS	10^{-3}	57	10.0715, 27.5560, 2.6739	0.0072, 0.0159, 0.0027
x, z	BFGS	10^{-4}	157	10.2219, 27.6649, 2.6759	0.0222, 0.0120, 0.0035
x, z	BFGS	10^{-5}	416	10.2102, 27.7936, 2.6579	0.0210, 0.0074, 0.0033
x, z	AL	10^{-6}	4,336	10.0075, 27.8580, 2.6664	0.0008, 0.0051, 0.0001
y, z	BFGS	1.4×10^{-3}	70	10.1951, 27.8560, 2.7588	0.0195, 0.0051, 0.0346
y, z	AL	10^{-4}	4,80	9.9428, 27.9816, 2.7494	0.0057, 0.0007, 0.0310
x, y, z	BFGS	10^{-3}	63	10.1043, 27.7523, 2.7427	0.0104, 0.0088, 0.0285
x, y, z	BFGS	10^{-4}	185	10.0166, 28.0205, 2.7022	0.0017, 0.0007, 0.0133
x, y, z	BFGS	10^{-5}	229	10.0023, 28.0336, 2.7011	0.0002, 0.0012, 0.0129
x, y, z	BFGS	10^{-6}	445	9.9972, 28.0433, 2.6996	0.0003, 0.0015, 0.0123
x, y, z	AL	10^{-6}	5,280	9.9638, 27.9544, 2.6776	0.0036, 0.0016, 0.0041

This table shows good results when one, two or three state variables are observable, however when only one state variable is observable the BFGS algorithm admits tolerances down to $\epsilon = 10^{-3}$ to get convergent results. When two state variables are observable, each combination gives good results, but the combinations x, y and y, z are less flexible to smaller tolerances ϵ for BFGS. Overall, the best results are obtained when two or three variables are observable. AL yields better parameter estimation in most of the cases, however the penalty-BFGS algorithm is very competitive. Figure 3 shows the log of the relative gradient $||\mathbf{g}^{\ell}||/||\mathbf{g}^{0}||$ against iterations number when z, (x, z) and (x, y, z) are observable, respectively. The norm of the gradient decreases slower with AL (blue lines) than with BFGS (green lines) as iterations progress, but AL converges faster (to the given tolerance) to the optimal parameters $\boldsymbol{\theta} = (\sigma, \rho, \theta)$ with two and three state observable variables, showing that updating the penalty parameters and Lagrange multipliers helps to improve convergence in the direction of the vector coordinate $\boldsymbol{\theta}$.

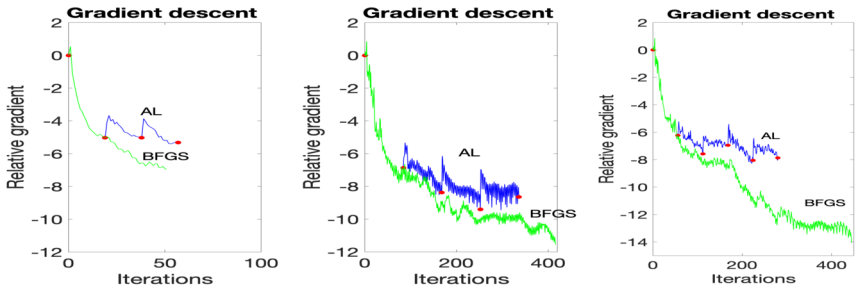

Figure 3: Gradient reduction for BFGS and AL iterations: z is observable (left), x, z are observables (middle), and x, y, z are observables (right). Red dots represent AL iterations.

Figure 4 shows how $(\sigma^{\ell}, \rho^{\ell}, \beta^{\ell})$ converge to the optimal $(\sigma^*, \rho^*, \beta^*)$. The true parameter value (σ, ρ, β) is included in each plot, represented with a circular black dot at the right end of each curve. Convergence improve when two or three variables are observable, although the difference between each case is not very significant, except for the number of iterations. To better illustrate convergence, we also include plots of $\log|\sigma - \sigma^*|$ versus iterations. This is done only for this parameter in each case to save space. Figure 5 shows the time-history of the state variables, obtained with the computed shooting parameters $\mathbf{s}_j^*$ and the computed parameters $\boldsymbol{\theta}^* = (\sigma^*, \rho^*, \beta^*)^T$ when x, z are observable (solid-colored segments), along with the noisy data in the experimental time-window $[0, 3]$ (dots), and the exact solution (continuous

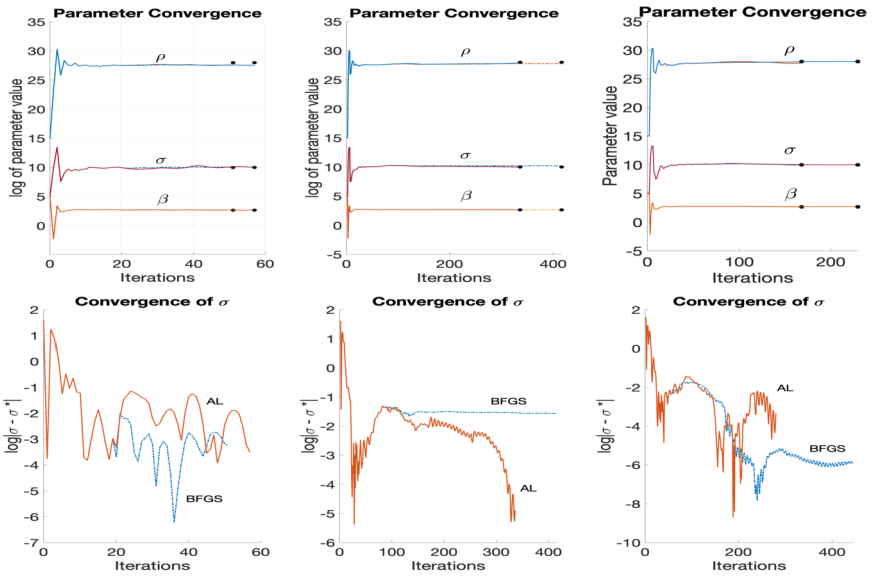

Figure 4: Parameters convergence when z is observable (left), x, z are observables (middle), and x, y, z are observables (right). The convergent curves are shorter for AL when x, z and x, y, z are observables. The bottom plots show the convergence of parameter σ in log scale for each case in the top.

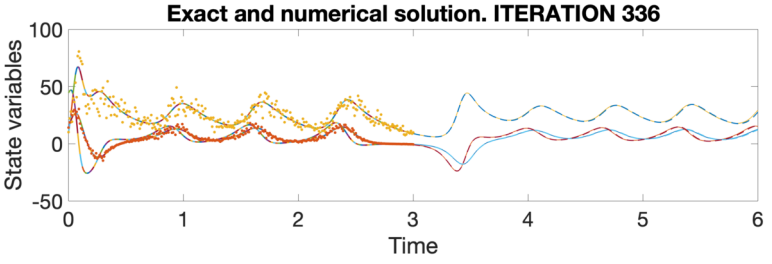

Figure 5: Time history of (x, y, z) obtained with the computed parameters. Case with AL, (x, z) observable variables, $noise\_level = 0.2$ and $ns = 150$.

line). For best illustration, we show the forecast in the interval $[3, 6]$ (dashed line), with initial condition $\mathbf{x}|_{t=3} = \mathbf{s}^*_{ns+1}$, and the corresponding computed parameters $\boldsymbol{\theta}^*$. The relative component-wise difference between the exact value $\mathbf{x}(t = 3)$ and $\mathbf{s}^*_{ns+1}$ is $(0.0083, 0.0054, 0.0013)$. Good agreement with the 'exact solution' (continuous line) is observed. Similar plots are obtained with the other cases.

Example 3.2. *In this example we investigate the performance of the methodology with respect to noise level. Table 2 summarizes the numerical results obtained for the following perturbations:* $10\%, 20\%, 30\%, 50\%$.

Table 2: Numerical results for different values of *noise_level*. We set $\theta^0 = (5, 15, 5)^T$ and $ns = 150$ in the same experimental window, for all cases

Noise	Obs*	ALG*	ϵ	Iters*	$\sigma^*, \rho^*, \beta^*$	Relative diff: σ, ρ, β
10%	z	BFGS	10^{-3}	63	9.7883, 27.8798, 2.6969	0.0212, 0.0043, 0.0113
10%	z	AL	10^{-4}	4,124	10.0318, 28.0167, 2.6655	0.0032, 0.0006, 0.0004
20%	z	BFGS	10^{-3}	51	10.0384, 27.5645, 2.6796	0.0038, 0.0156, 0.0048
30%	z	BFGS	2×10^{-3}	60	10.5232, 28.1838, 2.6024	0.0523, 0.0066, 0.0241
30%	z	AL	2×10^{-3}	4,62	9.9495, 27.8286, 2.6905	0.0051, 0.0061, 0.0089
50%	z	BFGS	12×10^{-3}	91	9.0340, 30.5445, 6.0347	0.0966, 0.0909, 1.2630
50%	z	AL	10^{-3}	5,90	9.9714, 28.2055, 2.9063	0.0029, 0.0073, 0.0899
10%	x, z	BFGS	10^{-4}	100	10.0941, 28.2135, 2.6882	0.0094, 0.0076, 0.0081
10%	x, z	AL	10^{-5}	3,99	10.0442, 28.0458, 2.6941	0.0044, 0.0016, 0.0103
20%	x, z	BFGS	10^{-5}	416	10.2102, 27.7936, 2.6579	0.0210, 0.0074, 0.0033
30%	x, z	BFGS	10^{-3}	431	10.0689, 28.0454, 2.6333	0.0069, 0.0016, 0.0125
30%	x, z	AL	10^{-5}	5,295	9.9971, 27.9732, 2.6419	0.0003, 0.0010, 0.0093
50%	x, z	BFGS	10^{-3}	160	9.3661, 27.8869, 2.8947	0.0634, 0.0040, 0.0855
50%	x, z	AL	10^{-5}	5,130	9.7488, 27.2330, 2.8287	0.0251, 0.0274, 0.0608
10%	x, y, z	BFGS	10^{-5}	164	10.0270, 28.1979, 2.6586	0.0027, 0.0071, 0.0030
10%	x, y, z	AL	10^{-6}	5,160	10.0129, 28.0234, 2.6733	0.0013, 0.0008, 0.0025
20%	x, y, z	BFGS	10^{-5}	229	10.0023, 28.0336, 2.7011	0.0002, 0.0012, 0.0129
30%	x, y, z	BFGS	10^{-4}	298	10.0713, 27.9585, 2.6790	0.0071, 0.0015, 0.0046
30%	x, y, z	AL	10^{-5}	5,271	9.9894, 28.2285, 2.6547	0.0011, 0.0082, 0.0045
50%	x, y, z	BFGS	10^{-4}	384	10.4567, 27.9382, 2.7624	0.0457, 0.0022, 0.0359
50%	x, y, z	AL	10^{-4}	8,278	10.0458, 27.0393, 2.6548	0.0046, 0.0343, 0.0045

For 10% perturbation, a visible improvement (with respect to the 20% case) is obtained when two or three state variables are observable, and a good parameter estimation is still obtained for 30% perturbation. But, for a perturbation of 50%, precision is lost, although an acceptable estimate of the parameters is still obtained. However, in this case convergence slows down and larger values of ϵ are necessary to stop the iterations. Figure 6 shows the reduction of the gradient as the iterations progress, when the observable variables are x, z. Figure 7 illustrates the convergence curves of the parameters with respect to iterations for the same case. Like in Figure 4 we include plots of $\log|\sigma - \sigma^*|$ versus iterations.

In this case, unlike Example 1, we must choose smaller initial values in (2.20) for high noisy cases. This is because a high noise can imply a greater deviation $||\mathbf{x}_j^\ell(\tau_{j+1}) - \mathbf{s}_{j+1}^\ell||$ and, consequently, a uncontrolled increase in k_j^ℓ, if the initial estimate $\mathbf{k}^0 = \{k_j^0\}_{j=1}^{ns}$ is not chosen carefully. In particular, the

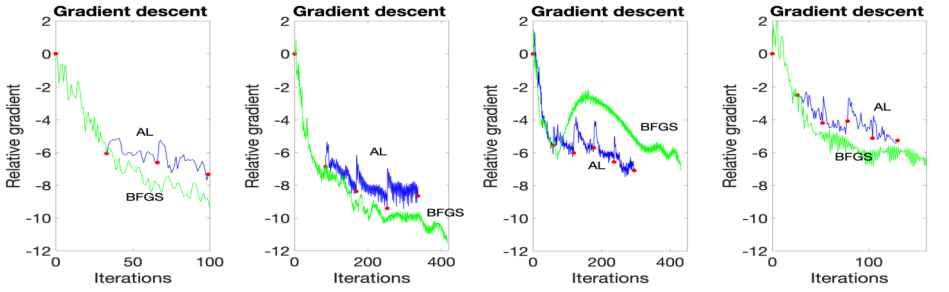

Figure 6: Gradient descent with respect to iterations for different levels of noise in the perturbed data. From left to right, 10%, 20%, 30%, 50%. Case when x, z are observable.

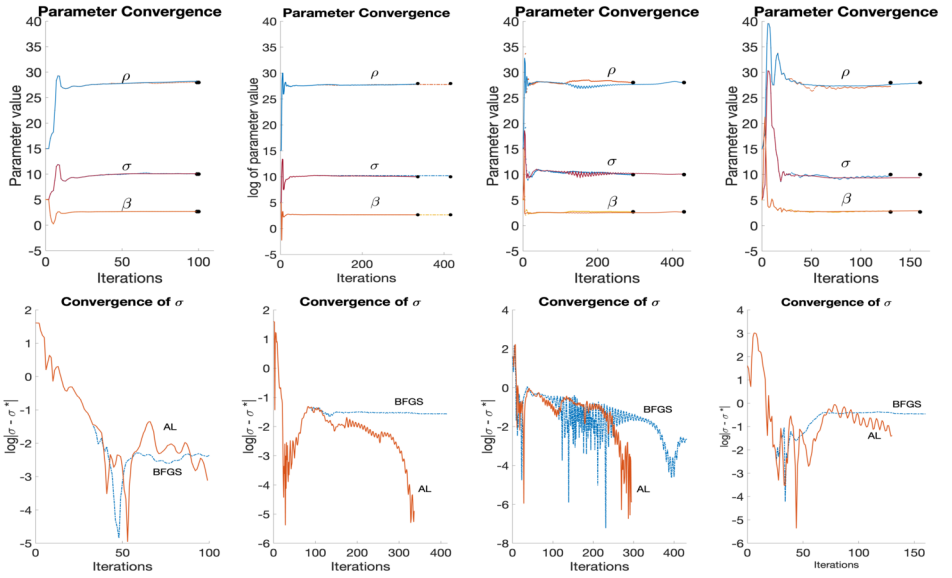

Figure 7: Parameters convergence for 10%, 20%, 30%, 50% noise in the perturbed data (left to right), when x, z are observable. The bottom plots show convergence of parameter σ in log scale for each case in the top. AL convergent curves are shorter than those for BFGS, except for the first case.

following values were chosen for 30% and 50% perturbations, respectively:

$$k_j^0 = \begin{cases} 1.0 \text{ for } z \text{ observable,} \\ 0.3 \text{ for } (x,z) \text{ observable,} \\ 0.025 \text{ for } (x,y,z) \text{ observable,} \end{cases} \qquad k_j^0 = \begin{cases} 10^{-5} \text{ for } z \text{ observable,} \\ 0.1 \text{ for } (x,z) \text{ observable,} \\ 0.05 \text{ for } (x,y,z) \text{ observable.} \end{cases}$$

The influence of high noisy data is observed in Figure 7, especially the oscillations that arise on the convergence curves with AL for 30% and 50% cases, as well as the higher picks at the beginning of iterations for 30% and 50% perturbations.

4. Application to control transition between equilibria, and stabilization around an unstable equilibrium

We discuss the application of multiple shooting and optimization models, based on augmented Lagrangian and quasi-Newton methods, to the numerical simulation of control processes and stabilization of a Josephson Junction Array (JJA). This problem was studied previously in [24, 32], where a time partitioned method and a conjugate gradient algorithm are applied to a linearised model of the problem. We show that the multiple shooting approach address this difficult problem in a much easier and effective way.

A Josephson junction is a quantum interference device that consists of two superconductors coupled by a weak link that may be, for example, an insulator or a ferromagnetic material. For more details about the description, properties, applications and importance of these devices, we refer the reader to [8, 24] and references therein. Here, we are interested in the following dimensionless first order non-linear equations, modelling the dynamics of an array (circuit) of three Josephson junctions inductively coupled:

$$\Gamma \frac{d\boldsymbol{\phi}(t)}{dt} + K\boldsymbol{\phi}(t) + \sin(\boldsymbol{\phi}(t)) = \mathbf{i}_c + \mathbf{v}(t) \quad \text{in} \quad (t_0, t_f], \tag{4.1}$$

$$\boldsymbol{\phi}(t_0) = \boldsymbol{\phi}_0. \tag{4.2}$$

where $\boldsymbol{\phi}(t)$ and $\mathbf{v}(t)$ are 3D vector functions of $t \in [t_0, t_f]$, and

$$\Gamma = \begin{bmatrix} \gamma_1 & 0 & 0 \\ 0 & \gamma_2 & 0 \\ 0 & 0 & \gamma_3 \end{bmatrix}, \quad K = \begin{bmatrix} \kappa_1 & -\kappa_1 & 0 \\ -\kappa_1 & \kappa_1 + \kappa_2 & -\kappa_2 \\ 0 & -\kappa_2 & \kappa_2 \end{bmatrix},$$

$\sin(\boldsymbol{\phi}(t)) = (\sin(\phi_1(t)), \sin(\phi_2(t)), \sin(\phi_3(t)))^T$. In (4.1), $\mathbf{i}_c = (1, 0.8, -1)^T$ is a critical direct current vector and $\mathbf{v}(t)$ is a vector current pulse (our control function), $\phi_i(t)$ and $\gamma_i\, d\phi_i/dt$ are the phase difference and the voltage across each junction, respectively. We will consider the parameter values employed in [24, 32]: $\gamma_1 = 0.7$, $\gamma_2 = 1.1$, $\gamma_3 = 0.7$, $\kappa_1 = \kappa_2 = 0.1$. For these parameters the uncontrolled system, i.e. $\mathbf{v}(t) = \mathbf{0}$, has several steady state solutions, which may be stable or unstable, consisting of constant phase

triplets $\boldsymbol{\phi}(t) = \boldsymbol{\theta} := (\theta_1, \theta_2, \theta_3)^T$. Some approximations to the stable and unstable steady states are given in Table 3.

Table 3: Some stable and unstable equilibria of system (4.1)

Stable equilibria	Unstable equilibria
1.2517, 0.7458, −0.9752	6.8539, 2.2566, −2.6015
7.4205, 6.4954, −0.3237	13.2016, 9.1355, −0.0786
12.9821, 7.0148, −0.2746	13.5568, 11.9196, −3.7436
13.6159, 12.2878, 0.2094	

In this part of the work, we continue with the investigation of numerical methods for the control of transitions between steady states, particularly transitions form an stable steady state to an unstable steady state [24], and also about the stabilization around the unstable one [32]. Here, we go beyond the methodology applied in those publications, where a standard approach, closely related to the methodology discussed in [23] for systems modelled by partial differential equations, is applied. The main ingredient to control transitions in [24] is the application of the conjugate gradient algorithm to a quadratic cost function model. Concerning the stabilization of phase junctions around an unstable equilibrium studied in [32], the idea was to compute the control of a linear perturbed system of the state equation in short-time intervals with conjugate gradient algorithms. We have found that such a time-partitioning method is quite cumbersome to implement computationally. Also, the method is designed to obtain general optimal controls in Hilbert L^2 spaces, and they may not be appropriate for practical applications.

Here, we will introduce an optimization model based on the multiple shooting strategy and augmented Lagrangian, which allow a more robust and efficient algorithm for the numerical solution of both problems, control of transitions and stabilization, simultaneously. Also, we will show that it is possible to apply the methodology to get controls of particular shape, like piecewise constant, piecewise linear or any other of finite dimensional type, which are more appropriate for practical applications. We found that the augmented Lagrangian, with the adjoint equation approach, allows the application of the standard BFGS algorithm in a simple manner. We consider that this approach is an alternative to more elaborated quasi-Newton algorithms, like Gauss-Newton and SQP algorithms, which are popular in engineering applications for non-linear programming when the multiple shooting approach is employed. Furthermore, we will show that this methodology allows to control and stabilize with any combination of three or two controls,

and even with only one control, which is the best option for practical applications.

4.1. Control of transitions and stabilization based on multiple shooting

We first consider the time-shooting nodes $\tau_1, \tau_2, \ldots, \tau_{ns+1}$, and their associated shooting vector parameters $\mathbf{s}_1$, $\mathbf{s}_2$, ..., $\mathbf{s}_{ns+1} \in \mathbb{R}^3$. As before, we take $\tau_1 = t_0$ and $\tau_{ns+1} = t_f$. We define control functions $\mathbf{v}_j(t)$ on each time-subinterval $I_j = [\tau_j, \tau_{j+1}]$, $j = 1, \ldots, ns$. These control functions are approximated by a linear combination of a given finite set of base functions, i.e. $\mathbf{v}_j(t) = \sum_{k=1}^{n} q_j^k(t)\, \mathbf{c}_j^k$. The simplest control functions are discontinuous piecewise constant functions and continuous piecewise linear functions. More precisely, for each $j = 1, \ldots, ns$, and $t \in I_j$, we may define either

$$\mathbf{v}_j(t) = \mathbf{c}_j, \quad \text{or} \tag{4.3}$$

$$\mathbf{v}_j(t) = \frac{\tau_{j+1} - t}{\tau_{j+1} - \tau_j}\, \mathbf{c}_j + \frac{t - \tau_j}{\tau_{j+1} - \tau_j}\, \mathbf{c}_{j+1}. \tag{4.4}$$

Therefore, this approach allows to generate a finite dimensional control space $\mathcal{U}$ containing vector functions of the given particular type. This control space is a closed subspace of the Hilbert space $(L^2([t_0, t_f])^3$. We want to find the unknown constant vectors $\mathbf{c}_1, \ldots, \mathbf{c}_{ns}$ for piecewise constant functions, or the unknown vectors $\mathbf{c}_1, \ldots, \mathbf{c}_{ns+1}$ for continuous piecewise linear functions. Employing multiple shooting for the solutions of the differential equations, increments the number of unknown parameters. Now, we have $\mathbf{p} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{ns+1}, \mathbf{c}_1, \ldots, \mathbf{c}_{ns})^T$ for piecewise constant functions, or $\mathbf{p} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{ns+1}, \mathbf{c}_1, \ldots, \mathbf{c}_{ns+1})^T$ for continuous piecewise linear functions, with $\mathbf{s}_j$, $\mathbf{c}_j \in \mathbb{R}^3$. For each shooting time subinterval I_j, we define $\mathbf{p}_j = (\mathbf{s}_j, \mathbf{s}_{j+1}, \mathbf{c}_j)^T$ for piecewise constants, or $\mathbf{p}_j = (\mathbf{s}_j, \mathbf{s}_{j+1}, \mathbf{c}_j, \mathbf{c}_{j+1})^T$ for piecewise linear.

Optimization problem. We want to find an optimal control $\mathbf{v}(t) \in \mathcal{U}$ to drive the transition between two admissible phase states in the given time interval $I = [t_0, t_f]$ and stabilize the system simultaneously around the initial and final phase states. Let us denote by ϕ_0 the initial state, and by ϕ_f the final one. They are any two admissible states, including stable steady phase states or unstable steady phase states. We first divide the whole time-interval I in three time-subregions, taking two intermediate shooting-times τ_{n_1} and τ_{n_2}, where the indexes n_1 and n_2 satisfy $1 < n_1 < n_2 < ns$. The

idea is to stabilize the system around ϕ_0 in the first time-subregion $[\tau_1, \tau_{n_1}]$ and, simultaneously allow the transition from ϕ_0 to ϕ_f in the second time-subregion $[\tau_{n_1}, \tau_{n_2}]$ as well as stabilize the system around ϕ_f in the third time-subregion $[\tau_{n_2}, \tau_{ns+1}]$. Then, in order to obtain this desired behaviour we propose the following optimization model, hoping to get an optimal control in a finite dimensional space defined by piecewise (constant or linear) vector functions:

$$\min_{\mathbf{p}} \; J(\mathbf{p}) := \sum_{j=1}^{ns} L_j(\mathbf{p}_j), \tag{4.5}$$

where we define $L_j(\mathbf{p}_j)$, depending on the time-subregion:

$$L_j(\mathbf{p}_j) = \begin{cases} \dfrac{1}{2}\displaystyle\int_{I_j} \|\mathbf{v}_j(t)\|^2 dt + \frac{k_{1j}}{2}\int_{I_j} \|\phi_j(t) - \phi_0\|^2 dt, \text{ if } 1 \le j \le n_1, \\ \dfrac{1}{2}\displaystyle\int_{I_j} \|\mathbf{v}_j(t)\|^2 dt + \frac{k_{1j}}{2}\int_{I_j} \|\phi_j(t) - \mathbf{s}_{j+1}\|^2 dt, \text{ if } n_1 < j < n_2, \\ \dfrac{1}{2}\displaystyle\int_{I_j} \|\mathbf{v}_j(t)\|^2 dt + \frac{k_{1j}}{2}\int_{I_j} \|\phi_j(t) - \phi_f\|^2 dt, \text{ if } n_2 \le j \le ns, \end{cases} \tag{4.6}$$

where the parameters $\mathbf{s}_j$, $\mathbf{s}_{j+1}$, $\mathbf{v}_j$ and the function $\boldsymbol{\phi}_j$ are constrained by the following two conditions:

$$\begin{cases} \Gamma \dfrac{d\boldsymbol{\phi}_j(t)}{dt} + K\boldsymbol{\phi}_j(t) + \sin(\boldsymbol{\phi}_j(t)) = \mathbf{i}_c + \mathbf{v}_j(t) \quad \text{in} \quad I_j, \\ \boldsymbol{\phi}_j(\tau_j) = \mathbf{s}_j, \end{cases} \tag{4.7}$$

$$\boldsymbol{\phi}_j(\tau_{j+1}) = \mathbf{s}_{j+1}. \tag{4.8}$$

The penalization parameters k_{1j} above are positive scalars to be determined.

Remark 4.1. *We may fix* $\mathbf{s}_1 = \phi_1(t_0) \equiv \boldsymbol{\phi}_0$*, but we prefer to let the parameter* $\mathbf{s}_1$ *to be unknown for more freedom and to measure how the methodology stabilizes the system around* $\boldsymbol{\phi}_0$*. So, we must add to* $L_1(\mathbf{p}_1)$ *in (4.6) the penalized term*

$$\frac{k_0}{2}\|\mathbf{s}_1 - \phi_0\|^2, \tag{4.9}$$

with $k_0 > 0$.

4.2. The augmented Lagrangian

The augmented Lagrangian for the constrained optimization problem, associated to each shooting-time subinterval I_j, is

$$\begin{aligned}(4.10)\qquad \mathcal{L}_{j,\mathbf{k}}(\mathbf{p}_j,\mathbf{y}_j,\boldsymbol{\lambda}_j) &= L_j(\mathbf{p}_j)\\ &+\int_{I_j}\mathbf{y}_j\cdot\left\{\mathbf{i}_c+\mathbf{v}_j(t)-\Gamma\,\dot{\boldsymbol{\phi}}_j(t)-K\boldsymbol{\phi}_j(t)-\sin(\boldsymbol{\phi}_j(t))\right\}dt\\ &+\left\{\boldsymbol{\lambda}_j\cdot[\boldsymbol{\phi}_j(\tau_{j+1})-\mathbf{s}_{j+1}]+\frac{k_{2j}}{2}\left\|\boldsymbol{\phi}_j(\tau_{j+1})-\mathbf{s}_{j+1}\right\|^2_{\mathbb{R}^3}\right\},\end{aligned}$$

where again, $\mathbf{y}_j$ and $\boldsymbol{\lambda}_j$ are the corresponding Lagrange multipliers associated to the (4.7) and (4.8), respectivley. The penalization parameters k_{2j} are positive constants. We apply the perturbation analysis, as in the previous section. For instance, for the case $1\le j\le n_1$ (first time-subregion), we obtain

$$\begin{aligned}(4.11)\qquad &\nabla_{\mathbf{p}_j}\mathcal{L}_{j,\mathbf{k}}\cdot\delta\mathbf{p}_j=\int_{I_j}(\mathbf{v}_j+\mathbf{y}_j)\cdot\delta\mathbf{v}_j\,dt+k_{1j}\int_{I_j}(\boldsymbol{\phi}_j-\boldsymbol{\phi}_0)\cdot\delta\boldsymbol{\phi}_j\,dt\\ &+\Gamma\,\mathbf{y}_j(\tau_j)\cdot\delta\mathbf{s}_j-\Gamma\,\mathbf{y}_j(\tau_{j+1})\cdot\delta\mathbf{s}_{j+1}-\int_{I_j}\{-\Gamma\,\dot{\mathbf{y}}_j+(K+C_j)\,\mathbf{y}_j\}\cdot\delta\boldsymbol{\phi}_j\,dt\\ &+\left\{\boldsymbol{\lambda}_j+k_{2j}(\boldsymbol{\phi}_j(\tau_{j+1})-\mathbf{s}_{j+1})\right\}\cdot(\delta\boldsymbol{\phi}_j(\tau_{j+1})-\delta\mathbf{s}_{j+1}),\end{aligned}$$

where C_j is the diagonal matrix with diagonal coefficients, $\cos(\phi_{1j}(t))$, $\cos(\phi_{2j}(t))$, $\cos(\phi_{3j}(t))$.

Remark 4.2. *For $j=1$, according to (4.9), we must add to the right hand side of (4.11) the penalized perturbed term*

$$(4.12)\qquad k_0\,(\mathbf{s}_1-\boldsymbol{\phi}_0)\cdot\delta\mathbf{s}_1.$$

For the other two time-subregions, the similar perturbed expression can be constructed from (4.11), doing the following:
1. For $n_1<j<n_2$, replace $k_{1j}\int_{I_j}(\boldsymbol{\phi}_j-\boldsymbol{\phi}_0)\cdot\delta\boldsymbol{\phi}_j\,dt$ by $k_{1j}\int_{I_j}(\boldsymbol{\phi}_j-\mathbf{s}_{j+1})\cdot\left(\delta\boldsymbol{\phi}_j-\delta\mathbf{s}_{j+1}\right)dt$.
2. For $n_2\le j\le ns$, replace $k_{1j}\int_{I_j}(\boldsymbol{\phi}_j-\boldsymbol{\phi}_0)\cdot\delta\boldsymbol{\phi}_j\,dt$ by $k_{1j}\int_{I_j}(\boldsymbol{\phi}_j-\boldsymbol{\phi}_f)\cdot\delta\boldsymbol{\phi}_j\,dt$.

4.3. The adjoint equation and the gradient

We introduce the following set of *adjoint equations*:

For $j = 1, \ldots, n_1$

$$-\Gamma\,\dot{\mathbf{y}}_j + (K + C_j)\,\mathbf{y}_j = k_{1j}\,(\boldsymbol{\phi}_j - \boldsymbol{\phi}_0) \quad \text{in} \quad I_j, \tag{4.13}$$

$$\Gamma\,\mathbf{y}_j(\tau_{j+1}) = \boldsymbol{\lambda}_j + k_{2j}(\boldsymbol{\phi}_j(\tau_{j+1}) - \mathbf{s}_{j+1}). \tag{4.14}$$

For $j = n_1 + 1, \ldots, n_2 - 1$

$$-\Gamma\,\dot{\mathbf{y}}_j + (K + C_j)\,\mathbf{y}_j = k_{1j}\,(\boldsymbol{\phi}_j - \mathbf{s}_{j+1}) \quad \text{in} \quad I_j, \tag{4.15}$$

$$\Gamma\,\mathbf{y}_j(\tau_{j+1}) = \boldsymbol{\lambda}_j + k_{2j}(\boldsymbol{\phi}_j(\tau_{j+1}) - \mathbf{s}_{j+1}). \tag{4.16}$$

For $j = n_2, \ldots, ns$

$$-\Gamma\,\dot{\mathbf{y}}_j + (K + C_j)\,\mathbf{y}_j = k_{1j}\,(\boldsymbol{\phi}_j - \boldsymbol{\phi}_f) \quad \text{in} \quad I_j, \tag{4.17}$$

$$\Gamma\,\mathbf{y}_j(\tau_{j+1}) = \boldsymbol{\lambda}_j + k_{2j}(\boldsymbol{\phi}_j(\tau_{j+1}) - \mathbf{s}_{j+1}). \tag{4.18}$$

From these equations and (4.11), we obtain the perturbation of the Lagrangian, which implicitly contains the gradient with respect to $\mathbf{s}_j$, $\mathbf{s}_{j+1}$ and $\mathbf{v}_j$:

$$\begin{aligned}\nabla_{\mathbf{p}_j}\mathcal{L}_{j,\mathbf{k}} \cdot \delta\mathbf{p}_j = \Gamma\,\mathbf{y}_j(\tau_j) \cdot \delta\mathbf{s}_j - \left\{\boldsymbol{\lambda}_j + k_{2j}(\boldsymbol{\phi}_j(\tau_{j+1}) - \mathbf{s}_{j+1})\right\} \cdot \delta\mathbf{s}_{j+1} \\ + \int_{I_j} (\mathbf{v}_j + \mathbf{y}_j) \cdot \delta\mathbf{v}_j\,dt,\end{aligned} \tag{4.19}$$

where $\boldsymbol{\phi}_j$ solves the state equation (4.7), and $\mathbf{y}_j$ solves the corresponding adjoint equation.

Remark 4.3. *For $j = 1$, we must add to (4.19) the term (4.12), and for $j = n_1 + 1, \ldots, n_2 - 1$ we subtract from (4.19) the term $k_{1j} \int_{I_j} (\boldsymbol{\phi}_j - \mathbf{s}_{j+1}) \cdot \delta\mathbf{s}_{j+1}\,dt$.*

We complete the calculation of the finite-dimensional gradient replacing $\mathbf{v}_j(t)$ by its finite-dimensional approximation. For instance, when the control functions are continuous piecewise linear functions (4.4), the last integral in (4.19) becomes

$$\int_{I_j} (\mathbf{v}_j + \mathbf{y}_j) \cdot \delta\mathbf{v}_j\,dt = \left\{ \int_{I_j} q_j^1(t) \left[q_j^1(t)\,\mathbf{c}_j + q_j^2(t)\,\mathbf{c}_{j+1} + \mathbf{y}_j(t) \right] dt \right\} \cdot \delta\mathbf{c}_j$$

$$+\left\{\int_{I_j} q_j^2(t)\,\left[q_j^1(t)\,\mathbf{c}_j + q_j^2(t)\,\mathbf{c}_{j+1} + \mathbf{y}_j(t)\right]\,dt\right\}\cdot\delta\mathbf{c}_{j+1}. \tag{4.20}$$

Therefore, expression (4.19) contains the contribution to the gradient associated to the parameters $\mathbf{p}_j = (\mathbf{s}_j, \mathbf{s}_{j+1}, \mathbf{c}_j, \mathbf{c}_{j+1})^T$ defined for the shooting-subinterval I_j. The case for piecewise constant functions is even simpler, since $\mathbf{p}_j = (\mathbf{s}_j, \mathbf{s}_{j+1}, \mathbf{c}_j)^T$ and

$$\int_{I_j} (\mathbf{v}_j + \mathbf{y}_j)\cdot\delta\mathbf{v}_j\,dt = \left\{\int_{I_j}\left[\mathbf{c}_j + \mathbf{y}_j(t)\right]\,dt\right\}\cdot\delta\mathbf{c}_j. \tag{4.21}$$

For both cases, the complete gradient defined on the time interval $I = [t_0, t_f]$ is constructed adding together the contribution of each shooting-subinterval I_j, $j = 1, \ldots, ns$. For instance, for piecewise constant controls the gradient is:

$$\nabla_{\mathbf{p}}\mathcal{L}_{\mathbf{k}} = \nabla J(\mathbf{p}) = \begin{bmatrix} \Gamma\,\mathbf{y}_1(\tau_1) & -k_0\,(\boldsymbol{\phi}_0 - \mathbf{s}_1) \\ \Gamma\,\mathbf{y}_2(\tau_2) & -\mathbf{r}_1 \\ & \vdots \\ \Gamma\,\mathbf{y}_{j+1}(\tau_{j+1}) & -\mathbf{r}_j \\ & \vdots \\ \Gamma\,\mathbf{y}_{ns}(\tau_{ns}) & -\mathbf{r}_{ns-1} \\ & -\mathbf{r}_{ns} \\ \displaystyle\int_{\tau_1}^{\tau_2}\left[\mathbf{c}_1 + \mathbf{y}_1(t)\right]dt & \\ & \vdots \\ \displaystyle\int_{\tau_{ns}}^{\tau_{ns+1}}\left[\mathbf{c}_{ns} + \mathbf{y}_{ns}(t)\right]dt & \end{bmatrix} \tag{4.22}$$

where, for $j = 1, \ldots, ns$,

$$\mathbf{r}_j = \boldsymbol{\lambda}_j + k_{2j}\,[\,\boldsymbol{\phi}_j(\tau_{j+1}) - \mathbf{s}_{j+1}\,] + k_{1j}\int_{I_j}\left(\boldsymbol{\phi}_j(t) - \mathbf{s}_{j+1}\right)dt. \tag{4.23}$$

The gradient with piecewise linear controls is similarly obtained, replacing the integral terms with the corresponding expressions given by (4.20) for each $j = 1, \ldots, ns$.

Remark 4.4. *The BFGS Algoritm 1 is applied in the same way that in Section 2 for parameter estimation. One just takes care of how* $\mathbf{p}$ *is defined and the way the gradient is computed. We also apply the AL Algorithm 2 in the same fashion and with the same rule for updating the penalization parameters* k_{2j} *starting with* $k_{2j}^0 = 1$*, but we must add to this algorithm an updating rule for the penalization parameters* k_{1j}*. We propose the following simple formula.*

$$k_{1j}^{\ell} = \begin{cases} \alpha\, k_{2j}^{\ell} & \text{for} \quad j = 1, \ldots, n_1 \quad \text{and} \quad j = n_2, \ldots, ns, \\ k_{2j}^{\ell} & \text{for} \quad j = n_1 + 1, \ldots, n_2 - 1, \end{cases} \tag{4.24}$$

with $\alpha \geq 1$*. That is, we may give more weight to the penalization parameters that enforce stabilization on the first and third time subregion, see (4.6), than the penalization parameters that enforce continuity shooting conditions (4.8). Both parameters come together in the source term and on the final conditions of the adjoint equations (4.13)–(4.18), respectively.*

Remark 4.5. *We have not justified formally that the proposed method converge to a feasible point. However, the following informal argument is pertinent regarding this issue. Since multiple shooting is a time partitioning method, and the length of each time-subintervals* I_j *is small, then a linear model in the neighbour of the shooting parameter* $\mathbf{s}_j$ *is a valid approximation of the state equation in* I_j*. Moreover, if* $\mathbf{v}_j^*$ *is an optimum of the local cost function* $L_j(\mathbf{p}_j)$ *with* $\mathbf{s}_j$ *and* $\mathbf{s}_{j+1}$ *fixed, then from convexity arguments, it may be characterized by the solution of the linear equation* $D_{\mathbf{v}_j} L_j(\mathbf{s}_j, \mathbf{s}_{j+1}, \mathbf{v}_j^*) = \mathbf{0}$ *in the local control subspace, where this derivative is an affine function of the coordinate* $\mathbf{v}_j$*. Therefore this linear equation could be solved by a (quadratic case)-conjugate gradient (or BFGS) algorithm operating in such control subspace. See [32] for more details.*

We will show in the next section that both AL as well as BFGS (for the pure penalized model), yield excellent numerical results for control and stabilization of the JJA system, and that BFGS is cheaper computationally.

5. Numerical examples for control and stabilization

We choose the stable equilibrium $\boldsymbol{\phi}_0 = (1.2517, 0.7458, -0.9752)^T$ as initial state at t_0, and the unstable equilibrium $\boldsymbol{\phi}_f = (13.2016, 9.1355, -0.0786)^T$ as target state. We want to drive the system from $\boldsymbol{\phi}_0$ to $\boldsymbol{\phi}_f$ in a given finite time, and then, after the transition, stabilize the system around this unstable equilibrium over a long time interval. Before obtaining numerical

results, we show the evolution of the uncontrolled system ($\mathbf{v}(t) = \mathbf{0}$) from both equilibria: $\boldsymbol{\phi}_0$ or $\boldsymbol{\phi}_f$. Figure 8 shows the evolution of the system (4.1) in the time interval $0 = t_0 \leq t \leq t_f = 40$, when the initial conditions is $\boldsymbol{\phi}_0$ (continuous lines), together with the evolution of the system when the initial condition is $\boldsymbol{\phi}_f$ (dashed lines). Clearly, starting from the stable equilibrium $\boldsymbol{\phi}_0$, the state variables keep constant in the given time interval, while the unstable equilibrium $\boldsymbol{\phi}_f$ evolves in time to other stable equilibrium: $\boldsymbol{\theta} = (12.9821, 7.0148, -0.2746)^T$, which apparently is the nearest stable equilibrium to the unstable one.

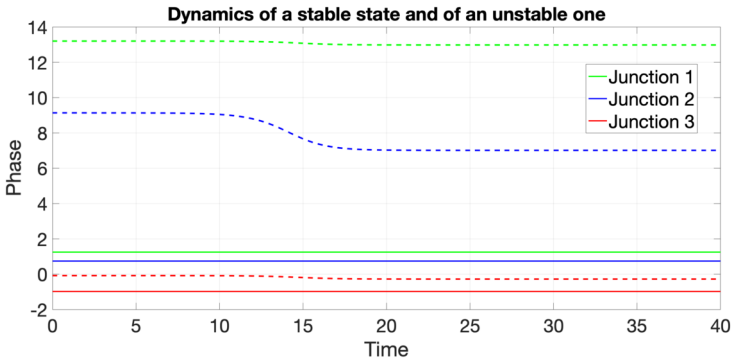

Figure 8: Time evolution of the uncontrolled stable solution (continuous line) and the uncontrolled unstable solution (dashed line) in the time interval $[0, 40]$.

Example 5.1. *We apply the multiple shooting methodology, the optimization model (4.5)–(4.9), to control the system with time subintervals* $[\tau_1, \tau_{n_1}] = [0, 5]$, $[\tau_{n_2}, \tau_{n_2}] = [5, 10]$, *and* $[\tau_{n_2}, \tau_{ns+1}] = [10, 40]$, *for stabilization around* $\boldsymbol{\phi}_0$ *(if necessary), transition, and stabilization around* $\boldsymbol{\phi}_f$, *respectively. These intervals are somewhat arbitrary, and we emphasize that the algorithm yields good similar results with other tested intervals and lengths. Our goal in this example is:*

- *Show that the system can be controlled with only one junction, for instance* ***junction 1****, using continuous PWL and discontinuous PWC control functions.*
- *Compare the performance of the AL algorithm with the plain BFGS algorithm.*
- *Show the role of the parameter* α, *(4.24), in the numerical results, where we choose* $k_0 = k_{2j}^0 = 1$ *for all* j.

The time step for the RK4 solver is $h = 0.01$ (uniform time-mesh). The initial guess for the shooting parameters, to start the iterations, for both AL and BFGS, is obtained with the linear-in-time combination

$$\mathbf{s}_j^0 = \frac{t_f - \tau_j}{t_f - t_0}\,\boldsymbol{\phi}_0 + \frac{\tau_j - t_0}{t_f - t_0}\,\boldsymbol{\phi}_f, \quad j = 1, \ldots, ns + 1,$$

while the initial guess for the control function parameters in (4.3) and (4.4) is set to zero: $\mathbf{c}_j^0 = \mathbf{0} \in \mathbb{R}^3$, $j = 1, \ldots, end$ ($end = ns$ for piecewise constant controls and $end = ns + 1$ for piecewise linear controls).

Table 4 shows different numerical results, controlling via ***junction 1*** and $ns = 100$ shooting subintervals. The following notation is used: ALG* = algorithm (AL or BFGS), CT* = control type (PWL or PWC), ϵ = tolerance iterations, $Iters^*$ = number of iterations to achieve convergence to the given tolerance, one number for the BFGS and two for AL (the number of AL iterations ℓ, and the total cumulative number of BFGS iterations). Last two columns show the component-wise relative difference between $\mathbf{s}_1$ and the initial state $\boldsymbol{\phi}_0$, as well as between $\mathbf{s}_{ns+1}$ and the final state $\boldsymbol{\phi}_f$ (target), multiplied by 10^3. Thus, to recover the true relative differences, the numerical values in last two columns must be multiplied by 10^{-3}. This table shows accurate numerical solutions with any combination. Both PWL and PWC control functions yield very similar numerical results, but PWC are cheaper computationally. AL gives excellent results, however BFGS achieves

Table 4: Numerical results for Example 5.1 with ***junction 1***, AL and BFGS algorithms, $h = 0.01$, $ns = 100$, and different values of α

α	ALG*	CT*	ϵ	Iters*	Rel diff. $(\boldsymbol{\phi}_0, \mathbf{s}_1) \times 10^3$	Rel diff. $(\boldsymbol{\phi}_f, \mathbf{s}_{ns+1}) \times 10^3$
1	AL	PWC	10^{-9}	2,126	0.4727, 2.0350, 0.2700	0.0016, 0.0050, 0.7415
1	AL	PWL	10^{-9}	2,138	0.4980, 2.0436, 0.2702	0.0016, 0.0050, 0.7411
1	BFGS	PWC	10^{-9}	91	0.1187, 0.4224, 0.0679	0.0013, 0.0036, 0.6084
1	BFGS	PWL	10^{-9}	99	0.1251, 0.4246, 0.0680	0.0013, 0.0036, 0.6084
1	BFGS	PWC	10^{-6}	71	0.1188, 0.4223, 0.0680	0.0013, 0.0036, 0.6080
1	BFGS	PWL	10^{-6}	71	0.1253, 0.4252, 0.0676	0.0013, 0.0036, 0.6086
10	AL	PWC	10^{-9}	3,207	0.0068, 0.0110, 0.0133	0.0007, 0.0025, 0.4725
10	AL	PWL	10^{-9}	4,280	0.0026, 0.0082, 0.0101	0.0008, 0.0024, 0.5216
10	BFGS	PWC	10^{-9}	111	0.0055, 0.0068, 0.0085	0.0006, 0.0017, 0.3413
10	BFGS	PWL	10^{-9}	125	0.0053, 0.0068, 0.0085	0.0006, 0.0017, 0.3413
10	BFGS	PWC	10^{-6}	76	0.0050, 0.0065, 0.0087	0.0006, 0.0017, 0.3413
10	BFGS	PWL	10^{-6}	83	0.0054, 0.0069, 0.0085	0.0006, 0.0017, 0.3413
100	BFGS	PWC	10^{-6}	140	0.0013, 0.0042, 0.0060	0.0002, 0.0012, 0.2615
100	BFGS	PWL	10^{-6}	155	0.0010, 0.0045, 0.0060	0.0003, 0.0012, 0.2615
1000	BFGS	PWC	10^{-6}	303	0.0002, 0.0041, 0.0056	0.0000, 0.0012, 0.2513
1000	BFGS	PWL	10^{-6}	343	0.0008, 0.0040, 0.0055	0.0001, 0.0012, 0.2514

the same accuracy with less computational work. Also, this table shows that it is enough to set $\epsilon = 10^{-6}$ with BFGS. Increasing α in (4.24) to update the penalty parameters k_{1j} in the optimization model (4.6) results in a numerical control which fits tighter to the transition time-interval $[t_{n1}, t_{n2}] = [5, 10]$ and, more important, allows a more accurate stabilization in the first and last time-subregions $[t_1, t_{n1}] = [0, 5]$ and $[t_{n2}, t_{ns}] = [10, 40]$.

The next figures provide further insight. Figure 9 shows a comparison of the reduction of the relative gradient obtained with AL and BFGS algorithms with $\epsilon = 10^{-9}$ when $\alpha = 1$, and $\alpha = 10$, respectively. We also include a third subfigure that shows how the relative norms $\|\boldsymbol{\phi}_0 - \mathbf{s}_1\|/\|\boldsymbol{\phi}_0\|$ and $\|\boldsymbol{\phi}_f - \mathbf{s}_{ns+1}\|/\|\boldsymbol{\phi}_f\|$ decrease as α increases.

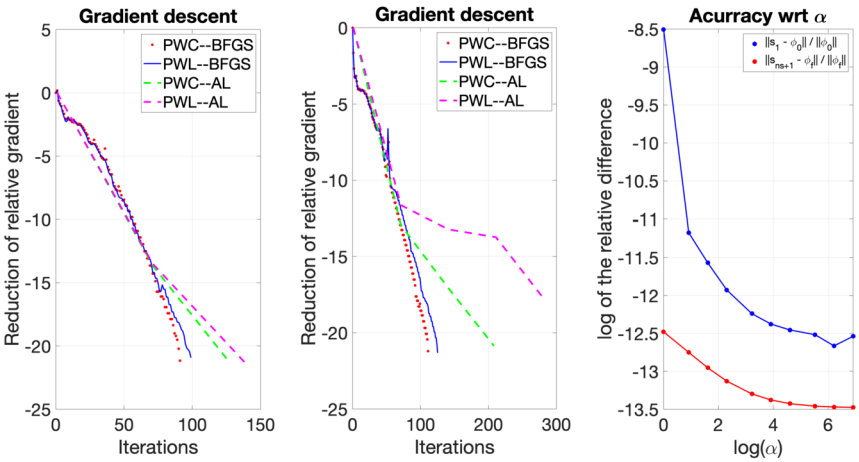

Figure 9: Gradient descent obtained with AL and BFGS algorithms, $\epsilon = 10^{-9}$, $\alpha = 1$ (left) and $\alpha = 10$ (middle). Reduction of relative differences as α increases for BFGS, PWC and $\epsilon = 10^{-6}$ (right).

Figures 10 and 11 show results that contrast cases with $\alpha = 1$ and $\alpha = 10$, respectively. The transition and stabilization, shown in the top-left of both figures, is represented by dots for PWC and continuous lines with PWL, showing an excellent agreement. These figures show how the control changes with respect to α, mainly around the transition time region $[5, 10]$. Finally, Figure 12 shows that the control away form the transition zone is very small, but it is not null. Summing up, the numerical results show that multiple shooting, along with the AL and BFGS optimization algorithms, is a very effective methodology to control the transition between admissible states and stabilize the system around an unstable state over a long time

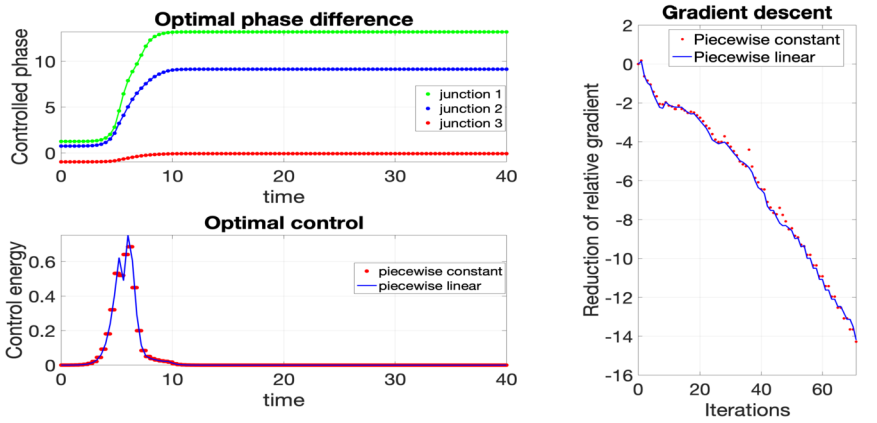

Figure 10: Transition from ϕ_0 to ϕ_f (top left) controlling via *junction 1* (bottom left). Reduction of the gradient with respect to iterations (right). Results obtained with the BFGS algorithm, $\epsilon = 10^{-6}$ and $\alpha = 1$.

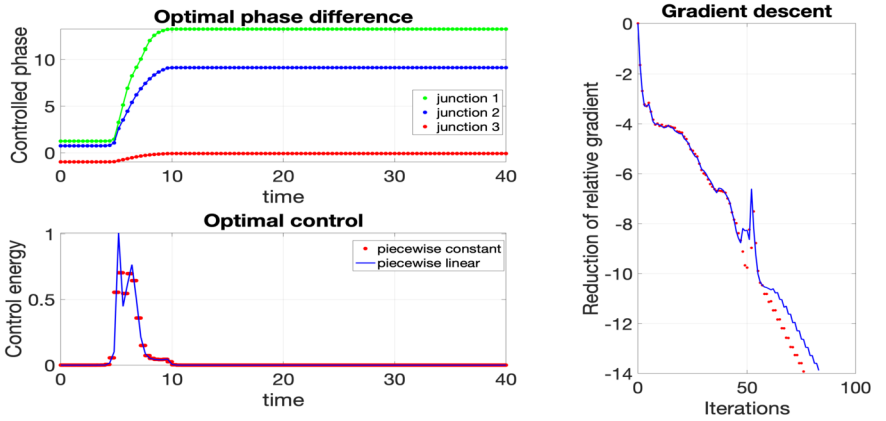

Figure 11: Transition from ϕ_0 to ϕ_f (top left) controlling via *junction 1* (bottom left). Reduction of the gradient with respect to iterations (right). Results obtained with the BFGS algorithm, $\epsilon = 10^{-6}$ and $\alpha = 10$.

interval. This is possible with ***junction 1***, but also with any combination of junctions (as we will show in the following examples). Both PWC and PWL control functions yield very similar numerical results under the same conditions, with PWC slightly cheaper computationally. The AL algorithm gives excellent results, at the expense of more iterations than BFGS (for the simple penalized model). Higher values of α for updating the penaliza-

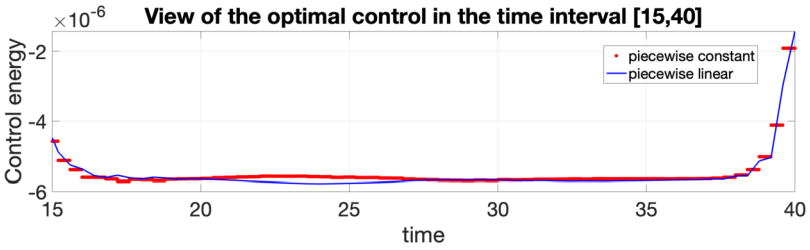

Figure 12: Controls associated to *junction 1* in the time interval $[15, 40]$ for $\alpha = 1$ and BFGS algorithm with stopping parameter $\epsilon = 10^{-6}$.

tion parameters k_{1j} increase the accuracy of the numerical results and yield control functions that fits the transition time interval $[5, 10]$ more tightly, requiring few more iterations. We found in this example that it is enough to solve the pure penalized optimization model with stopping value $\epsilon = 10^{-6}$ for the BFGS algorithm, and $\alpha = 10$, which is an intermediate reasonable value for updating the penalization parameters to enforce stabilization. So, in most of the next examples and cases we will use the penalized model with these parameters fixed.

Example 5.2. *In this example we investigate the behaviour of the numerical algorithm with respect to the number of shooting subintervals ns. Here we employ the BFGS algorithm with stopping tolerance* $\epsilon = 10^{-6}$ *and parameter* $\alpha = 10$. *We control only via* ***junction 1*** *and compute the PWC and PWL control functions with* $ns = 50$, 25 *and* 10 *subintervals.*

These numerical values and results are shown in Table 5. Figures 13, 14 and 15 illustrate the numerical results with $ns = 50$, $ns = 25$ and $ns = 10$ respectively.

These results show that the numerical methodology is able to compute the transition and stabilization with both, PWC and PWL control func-

Table 5: Numerical results of Example 5.2. Control via ***junction 1***, $h = 0.01$, $\alpha = 10$ and different number of shooting subintervals ns

ns	ALG*	CT*	ϵ	Iters*	Rel diff. $(\boldsymbol{\phi}_0, \mathbf{s}_1) \times 10^3$	Rel diff. $(\boldsymbol{\phi}_f, \mathbf{s}_{ns+1}) \times 10^3$
50	BFGS	PWC	10^{-6}	90	0.0103, 0.0174, 0.0186	0.0007, 0.0028, 0.5309
50	BFGS	PWL	10^{-6}	95	0.0070, 0.0183, 0.0188	0.0008, 0.0028, 0.5309
25	BFGS	PWC	10^{-6}	72	0.0896, 0.1513, 0.0558	0.0006, 0.0047, 0.7357
25	BFGS	PWL	10^{-6}	96	4.0795, 0.2650, 0.0584	0.0007, 0.0050, 0.7331
10	BFGS	PWC	10^{-6}	226	122.8312, 15.1424, 2.5375	0.0002, 0.0075, 0.8424
10	BFGS	PWL	10^{-6}	305	83.3319, 22.5437, 3.1771	0.0013, 0.0077, 0.8464

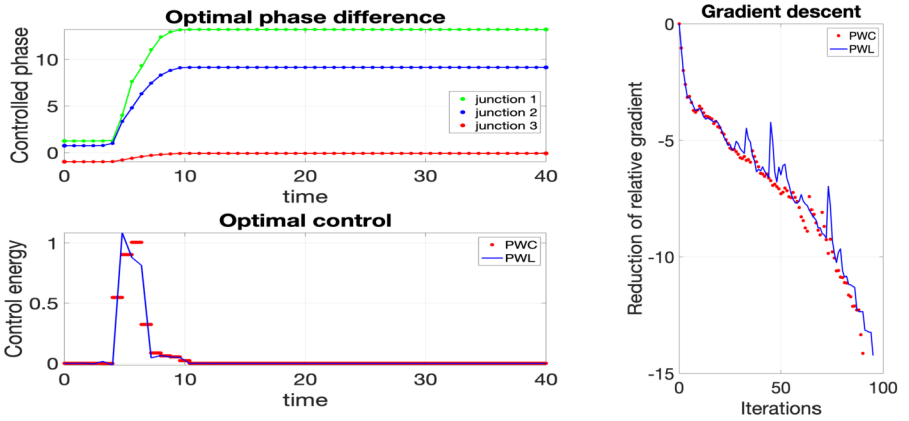

Figure 13: Transition and stabilization via *junction 1*, obtained with $ns = 50$ subintervals (left). Reduction of the gradient (right).

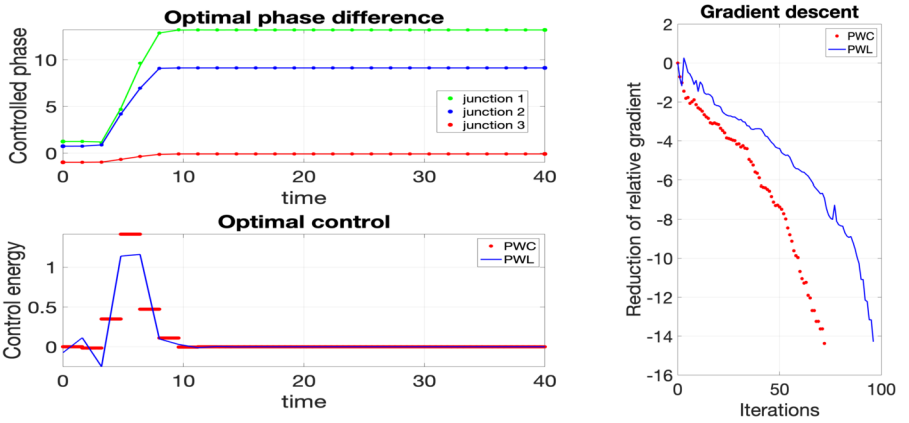

Figure 14: Transition and stabilization via *junction 1*, obtained with $ns = 25$ subintervals (left). Reduction of the gradient (right).

tions, for each case. Table 5 and figures show that the computed control has more difficulty to fit the time subintervals $[0, 5]$, $[5, 10]$ when $ns = 10$, requiring a substantial increment of BFGS iterations for convergence, and obtaining a higher relative difference of $(\mathbf{s}_1, \boldsymbol{\phi})$. However, the accuracy is maintained in the stabilization time interval $[10, 40]$ and the relative difference of $(\mathbf{s}_{ns+1}, \boldsymbol{\phi}_f)$ is much less sensitive to ns. Actually, the lowest value of ns admitted to get convergent results is $ns = 10$ for the given lengths of the first two time-subintervals, $[0, 5]$ and $[5, 10]$.

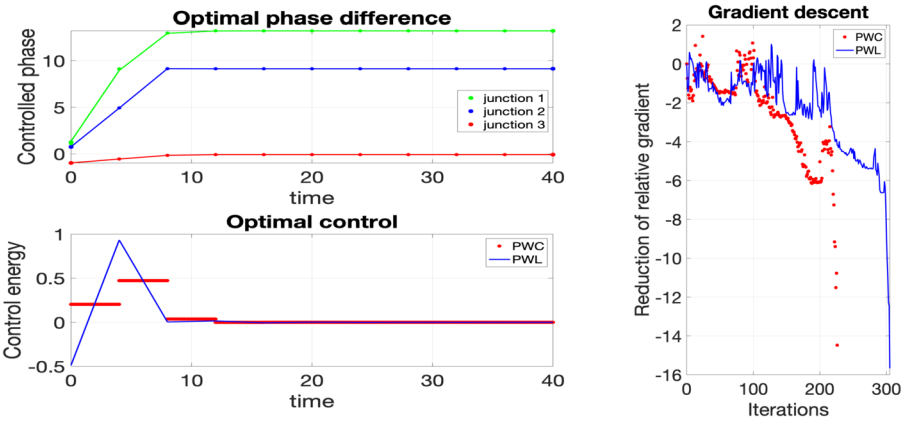

Figure 15: Transition and stabilization via *junction 1*, obtained with $ns = 10$ subintervals (left). Reduction of the gradient (right).

Example 5.3. *In this example we show that the proposed methodology allows controlling with any single junctions available or with combinations of two or three junctions. Again, we consider the BFGS algorithm,* $\epsilon = 10^{-6}$, *and* $ns = 100$. *This time we control with PWC functions, via one junction,* ***junction 1*** *or* ***junction 2*** *or* ***junction 3****, two* ***junctions [2,3]****, and three* ***junctions [1,2,3]*** *acting simultaneously.*

The numerical results for these five cases are shown in Table 6. Figure 16 shows the computed controls in the time subinterval $[0, 15]$ only. The first three cases are plotted together in the left subfigure, while the other two cases are shown separately in each remaining subfigures (center and right). Figure 17 show the corresponding gradient reduction for each case.

Table 6: Numerical results for Example 5.3. Controlling via different combinations of junctions

ALG*	CT*	CJ*	ϵ	Iters*	Rel diff. $(\boldsymbol{\phi}_0, \mathbf{s}_1) \times 10^3$	Rel diff. $(\boldsymbol{\phi}_f, \mathbf{s}_{ns+1}) \times 10^3$
BFGS	PWC	[1]	10^{-6}	76	0.0050, 0.0065, 0.0087	0.0006, 0.0017, 0.3413
BFGS	PWC	[2]	10^{-6}	74	0.0100, 0.0058, 0.0083	0.0008, 0.0015, 0.3412
BFGS	PWC	[3]	10^{-6}	77	0.0103, 0.0074, 0.0051	0.0008, 0.0017, 0.2632
BFGS	PWC	[2,3]	10^{-6}	78	0.0097, 0.0057, 0.0050	0.0008, 0.0015, 0.2630
BFGS	PWC	[1,2,3]	10^{-6}	81	0.0052, 0.0050, 0.0050	0.0006, 0.0015, 0.2630

The agreement between the different cases show that, from the computational point of view, there is no significant difference when controlling via any junction or combination of junctions. Of course, the most convenient

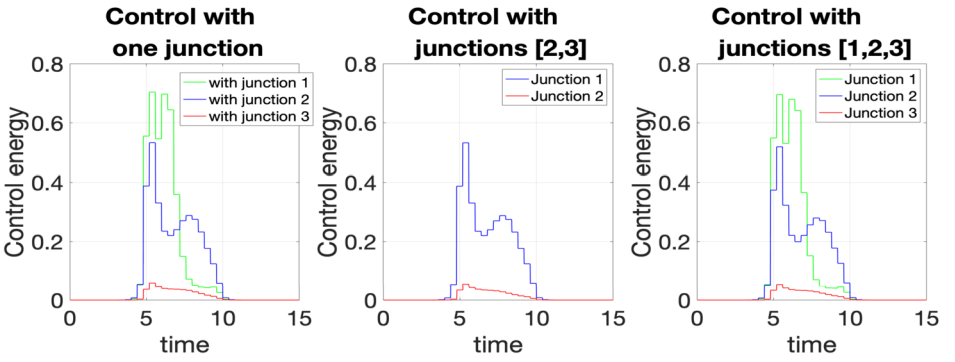

Figure 16: Computed PWC control in the time interval $[0, 15]$: via ***junction 1*** or ***2*** or ***3*** separately (left), via ***junctions [1, 2]*** acting together (center), and via ***junctions [1, 2, 3]*** acting simultaneously (right).

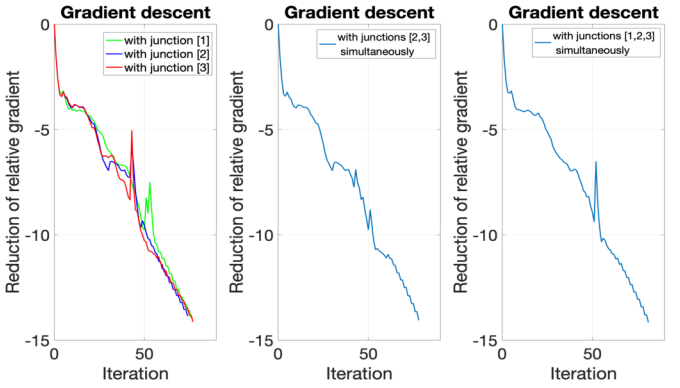

Figure 17: Gradient descent with only ***one junction*** (left), when ***junctions [1,2]*** act together (middle), and when ***junctions [1,2,3]*** act simultaneously (right).

in practical applications would be controlling with one junction only. We obtain similar results with piecewise linear functions, but those results are not included here.

Example 5.4. *In this example we show the dependence of the numerical results with respect to the time step size h. Again, the numerical results are obtained with the plain penalized model and BFGS iterations with stopping parameter $\epsilon = 10^{-6}$, PWC function controls and $\alpha = 10$. In this case we consider $ns = 50$ shooting subintervals and only* ***junction 2*** *for the control*

process. The following five different time steps, to solve the state and adjoint equations, are used: $h = 0.01,\ 0.04,\ 0.1,\ 0.4,\ 0.8$.

The numerical results are shown in Table 7. Figure 18 shows the control for each case (left), and the reduction of the gradient as iterations progress (right). These results are very consistent to each other, showing numerical convergence with respect to h. The most significant difference between each case is the computational time, as shown in Figure 19 (left). The transition between phase states and the stabilization around $\boldsymbol{\phi}_f$ is also very similar for each case (right). Again, multiple shooting, in combination with the penalized optimization model and the BFGS algorithm, yield excellent results, even for coarse time–meshes.

Table 7: Numerical results for Example 5.4 different values of step-size h. The BFGS algorithm is employed with $\epsilon = 10^{-6}$ and $\alpha = 10$

CT*	CJ*	h	ns	Iters*	Rel diff. $(\boldsymbol{\phi}_0, \mathbf{s}_1) \times 10^3$	Rel diff. $(\boldsymbol{\phi}_f, \mathbf{s}_{ns+1}) \times 10^3$
PWC	[2]	0.01	50	90	0.0223, 0.0105, 0.0175	0.0013, 0.0021, 0.5299
PWC	[2]	0.04	50	90	0.0217, 0.0111, 0.0179	0.0013, 0.0020, 0.5301
PWC	[2]	0.1	50	90	0.0218, 0.0103, 0.0178	0.0013, 0.0020, 0.5302
PWC	[2]	0.4	50	92	0.0214, 0.0100, 0.0173	0.0013, 0.0020, 0.5365
PWC	[2]	0.8	50	94	0.0201, 0.0089, 0.0153	0.0014, 0.0019, 0.5445

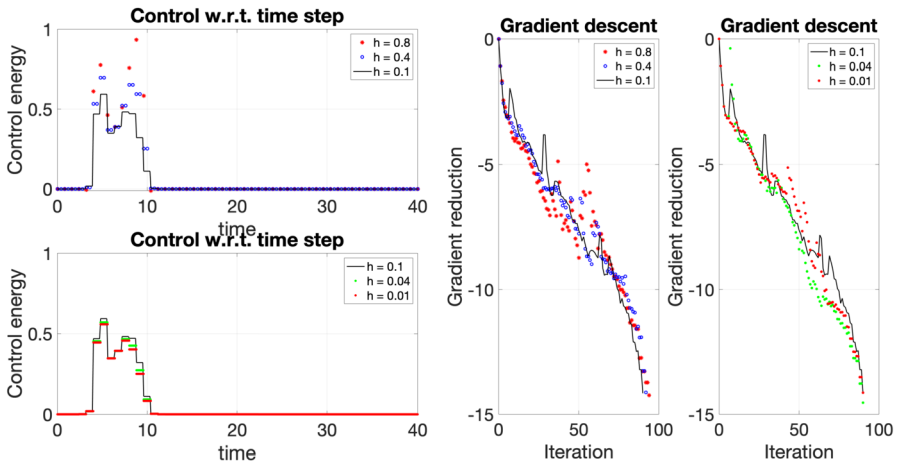

Figure 18: PWC control functions via *junction 2* obtained for the time step sizes $h = 0.8$, 0.4 and 0.1 (top left), and for $h = 0.1$, h = 0.04 and 0.01 (bottom left). Reduction of the gradient for each case (right).

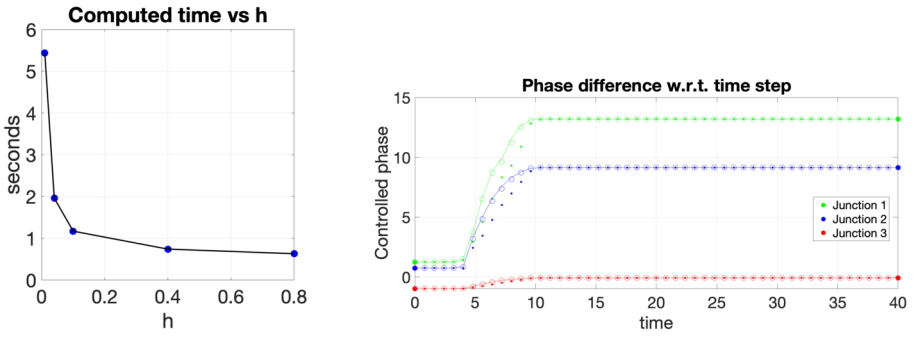

Figure 19: Computed time versus time step size h (left). The right subfigure shows the transition and stabilization with respect time step size obtained with $h = 0.8$ (dots), $h = 0.1$ (circles) and $h = 0.01$ (continuous line).

Summarizing, the proposed methodology has shown to be very effective for simultaneous control transitions between admissible states and stabilization around unstable states. It admits different types of finite dimensional controls, piecewise constant or linear, and we believe that it can be easily extended to other types, like Gaussian pulses, among others. We have shown that the multiple shooting is flexible also to: the number of subintervals, combinations of junctions for control, and different discretization parameters.

6. Conclusions

We have explored a multiple shooting strategy, which is applied to parameter estimation and also to optimal control and stabilization of problems modelled by ODEs. The arising equality constrained optimization problems are solved via an augmented Lagrangian approach. We have presented a detailed perturbation analysis to compute the gradient of the Lagrangian and of objective function, showing that the adjoint equation method incorporates the continuity (equality) constraints as final conditions of the corresponding backward in time adjoint equation, for each shooting subinterval. With this approach, the gradient of the objective function is computed efficiently, and the additional computational cost is related only to the additional memory required to storage the components associated to the shooting parameters $\mathbf{s}_1, \ldots, \mathbf{s}_{ns+1}$. This strategy allows the direct application of either standard augmented Lagrangian or BFGS algorithms. A further computational study, mainly comparing with other NLP solvers, like

Gauss-Newton methods and SQP, is an issue of a future work, which may help to motivate and give support to the idea of adopting the proposed BFGS algorithm for the penalized optimization model. Also, the incorporation of limited memory quasi-Newton optimization methods to large scale problems, like those associated to parameter estimation and optimal control of PDE based models, is a pending issue.

Apparently, the adjoint method for computing the gradient of the objective function has not been preferred for parameter estimation from noisy experimental data, as stated in [2]. Perhaps, the adjoint equation method is not used very much for parameter estimation, because the discrepancy of the data with the true state values arise as source pulses at instantaneous experimental times in the adjoint equations (2.17), requiring careful treatment, since it may lead to discontinuous solutions. However, our numerical results show that, at least for the Lorenz equations, the proposed methodology is able to estimate accurately the parameters in the regime where chaotic solutions arise, with either one, two or three observable state variables.

The same shooting approach via augmented Lagrangian, adapted to the control problem for the JJA dynamical system (4.1)–(4.2), turns out to be a very robust numerical method, which yields accurate solutions under different conditions, with either PWC or PWL controls. The numerical results show that this method can be applied to control the transition between phase states and, at the same time, stabilize the system around unstable equilibria over arbitrary long time-intervals, with any combination of two and three controls and also with only one. A key issue in this problem is the correct formulation of the optimization model. Finally, we want to point out that the methodology presented here can be well adapted for large scale control problems modelled by PDEs, as shown in a recent research work, [29].

Acknowledgements

We want to thank Professor Roland Glowinski (RG) for inviting us to work on the control problem for JJA, which is also of interest to Professor Braiman Yehuda [8]. Unfortunately, RG was no longer able to look at the work and results presented in this paper. We also want to thank the referee for his (her) careful reading and his (her) punctual and very accurate suggestions. Those suggestions helped greatly to improve this work. Finally, we acknowledge the Department of Mathematics at Universidad Autónoma Metroprolitana-Izatapalapa and CONACyT-Mexico for their financial support.

References

[1] I. K. Argyros, M. A. Hernández-Verón, M. J. Rubio, On the Convergence of Secant-Like Methods, *Current Trends in Mathematical Analysis and its Interdisciplinary Applications* (2019), 141–183. MR3970443

[2] O. Aydogmus, Ali Hakan TOR, A Modified Multiple Shooting Algorithm for Parameter Estimation in ODEs Using Adjoint Sensitivity Analysis, *Applied Mathematics and Computation Volume* **390** (2021) 125644. MR4146789

[3] E. Baake, M. Baake, H. G. Bock and K. M. Briggs, Fitting ordinary differential equations to chaotic data, *Phys. Rev. A* **45** (1992), 5524–5529.

[4] H.G. Bock, Numerical treatment of inverse problems in chemical reaction kinetics, in: Modelling of chemical reaction systems, Springer (1981), 102—125.

[5] H. G. Bock, Recent advances in parameter identification for ordinary differential equations, *Progress in Scientific Computing*, 2, eds. P. Deuflhard and E. Hairer, Birkhauser, Boston (1983), 95–121.

[6] H.G. Bock, E. Kostina, J.P. Schlöder, Direct multiple shooting and generalized Gauss-Newton method for parameter estimation problems in ODE models, in: *Multiple Shooting and Time Domain Decomposition Methods*, Springer, 2015, pp. 1–34. MR3676208

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, *Foundations and Trends in Machine Learning* 3, No. 1 (2011), 1–122.

[8] Y. Braiman, B. Neschke, N. Nair, N. Ima, and R. Glowinski. Memory States in Small Arrays of Josephson Junctions, *Physical Rewview E* **94**, 052223 (2016), 1–13.

[9] A. Buckley and A. LeNir, QN-like variable storage conjugate gradients, *Mathematical Programming*, 27 (1983) 155–175. MR0718057

[10] J. Calver and W. Enright, Numerical methods for computing sensitivities for ODEs and DDEs, *Numerical Algorithms* **74** (2017), 1101–1117. MR3626330

[11] Y. Cao, S. Li,L. Petzold, R. Serban, Adjoint sensitivity analysis for differential algebraic equations: the adjoint DAE system and its numerical solution, *SIAM J. Sci. Comput.* **24** (2003) 1076–1089. MR1950525

[12] F. Carbonell, Y. Iturria-Medina, J.C. Jimenez, Multiple Shooting-Local Linearization method for the identification of dynamical systems, *Communications in Nonlinear Science and Numerical Simulation*, **37** C (2016) 292–304. MR3466792

[13] T. Carraro, M. Geiger, Direct and indirect multiple shooting for parabolic optimal control problems, *Contributions in Mathematical and Computational Sciences, Springer International Publishing* (2015) 35–67. MR3676209

[14] J. J. Conde-Mones, L.H. Juárez, J. J. Oliveros, D.A. León-Velasco, Stable numerical solution of the Cauchy problem for the Laplace equation in irregular annular regions, *Journal Numerical Methods for PDE*, 33(6), (2017) 1799–1822. MR3708825

[15] J. E. Dennis and Robert B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs: Prentice–Hall (1983). MR0702023

[16] H. Diedam, S. Sager, Global optimal control with the direct multiple shooting method, *Optim. Control Appl. Methods* **39**(2) (2017) 449–470. MR3796946

[17] M. Diehl, H. Bock, H. Diedam, P.B. Wieber, Fast direct multiple shooting algorithms for optimal robot control, *Lecture Notes in Control and Information Sciences*, Springer Berlin Heidelberg (2006) 65–93. MR2271884

[18] B. van Domselaar and P. Hemker, Nonlinear parameter estimation in initial value problems, *Technical Report NW 18/75, Mathematical Centre Amsterdam* (1975).

[19] P. Drag, K. Styczén, Multiple shooting SQP algorithm for optimal control of DAE systems with inconsistent initial conditions, in: *Recent Advances in Computational Optimization* (2015), Springer, pp. 53–65. MR3381764

[20] D. Fernández, M. Solodov, On the cost of solving augmented lagrangian subproblems, *Mathematical Programming*, 182 (1-2) (2019) 37–55. MR4115643

[21] M.J. Gander, 50 years of time parallel time integration, *Contributions in Mathematical and Computational Sciences*, Springer International Publishing (2015) 69–113. MR3676210

[22] R. Glowinski, On Alternating Direction Methods of Multipliers: A Historical Perspective, *Modeling, Simulation and Optimization for Science and Technology*, (2014) 59–82. MR3330832

[23] R. Glowinski, J. L. Lions, J. He., Exact and Approximate Controllability for Distributed Parameter Systems, *Cambridge University Press*, Cambridge, UK (2008). MR2404764

[24] R. Glowinski, J. López, L.H. Juárez and Y. Braiman, On the controllability of transitions between equilibrium states in small inductively coupled arrays of Josephson junctions: A computational approach, *Journal of Computational Physics* **403**, (2020) 109023. MR4040737

[25] M. Jeon, Parallel optimal control with multiple shooting, constraints aggregation and adjoint methods, *Journal of Applied Mathematics and Computing* **19** (2005) 215–229. MR2162319

[26] L. H. Juárez and J. Rojas. Parameter estimation in ODEs. Modelling and computational issues, *Boletín de la Sociedad Mexicana de Computación Científica y sus Aplicaciones*, Año VIII, **No. 8** (2022) 34–49.

[27] C. Kirches, L. Wirsching, H. Bock, J. Schlöder, Efficient direct multiple shooting for nonlinear model predictive control on long horizons, *J. Process Control* **22** (3) (2012) 540–550.

[28] D.A. León Velasco, R. Glowinski and L. H. Juárez Valencia, On the controllability of diffusion processes on a sphere: a numerical study. *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 22, No. 4, (2016) 1054–1077. MR3570494

[29] L. Fang, S. Vandewalle, J. Meyers, A parallel–in–time multiple shooting algorithm for large–scale PDE–constrained optimal control problems, *Journal of Computational Physics*, Volume **452**, (2022) 110926. MR4361834

[30] L. Fang, S. Vandewalle, J. Meyers, An SQP-based multiple shooting algorithm for large-scale PDE-constrained optimal control problems, *Journal of Computational Physics* **477** (2023) 111927. MR4539478

[31] H. B. Keller, *Numerical Methods for Two-Point Boundary Value Problems*, London: Blaisdell (1968). MR0230476

[32] J. López, L.H. Juárez, R. Glowinski, Stabilizing a Josephson Junction Array Memory around an unstable equilibrium: A control approach using a first order state model, *Boletín de la Sociedad Mexicana de*

Computación Científica y sus Aplicaciones, Año VIII, **No. 8**, (2022) 4–33.

[33] J. Nocedal, Updating quasi-Newton matrices with limited storage, *Mathematics of Computation*, Vol. 35, No. 151 (1980), 773–782. MR0572855

[34] D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Mathematical Programming*, North-Holland 45 (1989), 503–528. MR1038245

[35] J. Nocedal, S. J. Wright, *Numerical Optimization*, New York: Springer (1999). MR1713114

[36] I. F. D. Oliveira, R. H. C. Takahashi, An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality, *ACM Transactions on Mathematical Software* **47**(1) (2020) 5:1–5:24. MR4199502

[37] M. R. Osborne, On shooting methods for boundary value problems, J. Math. Anal Appl, **27** (1969), 417–433. MR0245209

[38] M. Peifer, J. Timmer, Parameter estimation in ordinary differential equations using the method of multiple shooting — a review, *Freiburg Centre for Data Analysis and Modelling*, 1, 79104, Freiburg, Germany (2005).

[39] E.L. Piccolomini and F. Zama, Monitoring Italian COVID-19 spread by an adaptive SEIRD model, *PLoS One* **15**(8) (2020) e0237417.

[40] A. Potschka, Direct multiple shooting for parabolic PDE constrained optimization, *Contributions in Mathematical and Computational Sciences*, Springer International Publishing (2015), 159–181. MR3676213

[41] B. Sengupta, K.J. Friston, W.D. Penny, Efficient gradient computation for dynamical models, *NeuroImage* **98** (2014) 521–527.

[42] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, Springer–Verlag, New York (1980) (The original German version was published by Springer-Verlag Berlin Heidelberg, 1972, 1976). MR0557543

[43] S. Subchan, A direct multiple shooting for the optimal trajectory of missile guidance, 2*008 IEEE International Conference on Control Applications* (2008).

[44] W. Sun and Y.X. Yuan, Optimization Theory and Methods, *Nonlinear Programming*, New York, Springer (2006). MR2232297

[45] F. Tröltzsch, *Optimal Control of Partial Differential Equations: Theory, Methods ans Applications*, Graduate Studies in Mathematics, **vol 112**, AMS, Providence, Rhode Island (2010). MR2583281

L. Héctor Juárez
Universidad Autónoma Metropolitana - Iztapalapa
Av. Ferrocarril San Rafael Atlixco 186
Col. Leyes de Reforma 1 A Sección
Alcaldía Iztapalapa, C.P. 09310
Ciudad de México
México
E-mail address: hect@xanum.uam.mx

Jorge López
Universidad Juárez Autónoma de Tabasco
Av. Universidad s/n, Zona de la Cultura
Col. Magisterial, Vhsa, Centro
Tabasco, Mex. C.P. 86040
México
E-mail address: loppital1@hotmail.com

Jessica T. Rojas
Universidad Autónoma Metropolitana - Iztapalapa
Av. Ferrocarril San Rafael Atlixco 186
Col. Leyes de Reforma 1 A Sección
Alcaldía Iztapalapa, C.P. 09310
Ciudad de México
México
E-mail address: jessrocu@gmail.com

Received July 27, 2023