Constrained NLP via Gradient Flow Penalty Continuation: Towards Self-Tuning Robust Penalty Schemes

Felipe Scott[a], Raúl Conejeros[b], Vassilios S. Vassiliadis[c,*]

[a] *Green Technology Research Group, Facultad de Ingeniería y Ciencias Aplicadas, Universidad de los Andes, Chile, Mons. Álvaro del Portillo 12455, Las Condes, Santiago, 7620001, Chile*
[b] *School of Biochemical Engineering, Pontificia Universidad Católica de Valparaíso, Av. Brasil 2085, Valparaíso, Chile*
[c] *Department of Chemical Engineering and Biotechnology, University of Cambridge, Cambridge CB2 3RA, UK*

Abstract

This work presents a new numerical solution approach to nonlinear constrained optimization problems based on a gradient flow reformulation. The proposed solution schemes use self-tuning penalty parameters where the updating of the penalty parameter is directly embedded in the system of ODEs used in the reformulation, and its growth rate is linked to the violation of the constraints and variable bounds. The convergence properties of these schemes are analyzed, and it is shown that they converge to a local minimum asymptotically. Numerical experiments using a set of test problems, ranging from a few to several hundred variables, show that the proposed schemes are robust and converge to feasible points and local minima. Moreover, results suggest that the GF formulations were able to find the optimal solution to problems where conventional NLP solvers fail, and in less integration steps and time compared to a previously reported GF formulation.

Keywords: gradient flow, nonlinear programming problem, convergence analysis

1. Introduction

Optimization problems arise in many areas of chemical engineering practice, from component and systems design [1] to operation and control [2]. Due to an increasing concern of legislators and the general public in environmental sustainability, optimization has been recently used to aid in the design of supply chains and products considering their life cycle [3], and as a tool for the design of new sustainable energy conversion systems [4, 5]. Applications encompass formulations ranging from linear programming problems (LP) to mixed-integer non-linear programming problems (MINLP) and dynamic optimization problems (optimal control problems, OCP). A common feature of many of these classes of problems, is that at a certain point one or several non-linear constrained programming problems (NLP) need to be solved. The solution of large-scale NLP problems was made possible by breakthroughs in non-linear programming during the previous decades. In particular, the development of modern barrier methods [6, 7, 8], sequential quadratic programming [9] and reduced gradient methods [10], led to implementations (solvers) such as IPOPT [8] , SNOPT

*Corresponding author
Email address: vsv20@cam.ac.uk (Vassilios S. Vassiliadis)

[9] and CONOPT [10] that can be used in user-friendly modeling and optimization environments such as GAMS [11], AMPL [12] and AIMMS [13].

Most algorithms used to compute a local optimum of constrained NLP problems rely on Taylor series expansions truncated after the linear or quadratic term; according to this, constraints are linearized and large steps towards the local minimum are allowed. For this reason, in highly nonlinear problems intermediate iterations might prove infeasible and frequent failures to converge to a local optimum may arise. Alternatively to the Taylor expansion based methods, Gradient Flow (GF) methods have been proposed for the solution of unconstrained and constrained nonlinear programming problems. In its most simple version, the solution of an unconstrained problem $\min_x f(x)$ can be obtained by solving the following set of coupled ordinary differential equations (ODEs):

$$\frac{dx}{dt} = -\nabla_x f(x); \quad x(0) = x_0 \tag{1}$$

where $x \in \mathbb{R}^n$, $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^1$. This approach creates a smooth trajectory that might offer an advantage for highly nonlinear problems compared with the conventional optimization techniques which take finite steps along line-search directions. For the latter, finding a suitable step-size can be difficult when the optimization function is non-quadratic and has large third derivatives, resulting in a slow progress towards the solution due to the smalls steps required [14].

Another interesting feature of GF methods is the possibility of using state-of-the-art integration software to find the solution of optimization problems. This approximation for the solution of unconstrained problems can be traced to the work of Botsaris [15]. In the following decades, efforts were made to reach a competitive level in terms of computational time and iterations compared to conventional methods, with a summary found in Brown and Bartholomew-Biggs [14]. The application of GF methods was further extended by introducing new formulations that were able to cope with constrained nonlinear problems [16, 17, 18, 19]. The constraints of the NLP problem ($h(x)$) are incorporated to the objective function ($f(x)$) with a penalty scheme in order for GF methods to be employed, with one of the major issues being the updating of the penalty parameters utilized. For an optimization problem with equality constraints only, Tanabe [20] proposed the following Gradient Flow formulation:

$$\frac{dx}{dt} = -Q(x)\nabla_x f(x); \quad x(0) = x_0 \tag{2}$$

$$Q(x) = [I - J^T(x)(J(x)J^T(x))^{-1}J(x)]$$

where $\nabla_x f(x)$ and $J(x)$ represents the gradient of the objective function with respect to the optimization variables and the Jacobian matrix, respectively. Equation 2 is a direct generalization of the gradient projection method proposed by Rosen [21] to a differential form, which is based on projecting the search direction given by $\nabla_x f(x)$ into the subspace tangent to the active constraints. The method proposed by Tanabe [20] was further modified by Yamashita [22] and Evtushenko and Zhadan [17] to yield

$$\frac{dx}{dt} = -sQ(x)\nabla_x f(x) + -J(x)(J(x)J^T(x))^{-1}h(x); \quad x(0) = x_0 \tag{3}$$

with s a positive constant. Following the work of Evtushenko and Zhadan [17], Wang et al. [23] proposed an approach to include inequality constraints and bounds where a pseudo-inverse of the square matrix $J(x)J^T(x)$ acts as a penalizer (equation 4), with this approach requiring a non-singular Jacobian.

$$\frac{dx}{dt} = -[\nabla f(x) + J^T(x)(\tau h(x) - (J(x)J^T(x))^{-1}J(x)\nabla f(x))]; \quad x(0) = x_0 \tag{4}$$

In their work, they avoid the use of slacks to account for variable bounds by using the state-space transformation technique proposed by Evtushenko and Zhadan [17], otherwise the use of quadratic slacks would result in singular Jacobians. As proved by the above mentioned authors, their GF formulations have the property that once the equality constraints are satisfied, the trajectory of the solution will stay on the manifold determined by the constraints. However, as analyzed by Brown and Bartholomew-Biggs [16], the ODE system that allows following a path with those characteristics needs to be solved quite accurately. This and the fact that inverses of large matrices need to be calculated, produce a heavy numerical overhead. Moreover, in the formulations represented by equations 2 to 4, authors do not present an approach to select the values of the parameters required in their formulations (such as τ in equation 4). In practice, the value of these parameters needs to be adjusted to each problem. Finally, Schropp and Singer [24] compare SQP methods and GF methods for the solution of nonlinear problems from a theoretical point of view and using two case studies. They concluded that SQP methods are efficient tools whereas the ODE approach may be more reliable, with the ODE approach being more efficient for problems with only a small number of highly nonlinear constraints. Moreover, they propose an approach combining differential and algebraic equations that, according to the authors, combines efficiency and reliability.

In this work, a self-tuning penalty scheme is presented for the solution of constrained NLPs. The approach does not require the calculation of an inverse (or pseudo-inverse) of the Jacobian matrix, and the penalty parameters updating is directly embedded into the system of ODEs. The performance of the GF formulations presented in this work are compared to the formulation presented by Wang et al. [23] and also against several state-of-the-art NLP solvers.

2. New formulations using the Gradient Flow approach for solving NLP problems

2.1. Problem definition

The minimization of the following standard constrained NLP is considered:

$$\begin{aligned} &\min_{x_s} f(x_s) \\ \text{subject to:} \quad & \\ &h_s(x_s) = 0 \\ &g(x_s) \leq 0 \\ &x_s^L \leq x_s \leq x_s^U \end{aligned} \tag{5}$$

where $x_s \in \mathbb{R}^{n_s}$, $f(x_s) : \mathbb{R}^{n_s} \mapsto \mathbb{R}^1$, $h_s(x_s) : \mathbb{R}^{n_s} \mapsto \mathbb{R}^{m_1}$ and $g(x_s) : \mathbb{R}^{n_s} \mapsto \mathbb{R}^{m_2}$. The subscript s stands for standard, as this problem will be converted to a penalized version were the subscripts will be dropped to simplify the notation. This problem is converted to a penalty function minimization, using a quadratic penalty scheme and standard transformations. Inequality constraints are converted to equalities via the use of squared slack variables as follows. First, inequality constraints are converted to equality constraints using the following transformation:

$$g(x_s) + w^2 = 0 \tag{6}$$

where $w \in \mathbb{R}^{m_2}$.

Variables bounds are transformed to equalities, by using the following equations:

$$x_s + \left(s^U\right)^2 = x_s^U \tag{7}$$

$$x_s - \left(s^L\right)^2 = x_s^L \tag{8}$$

where s^L, $s^U \in \mathbb{R}^{n_s}$. The equality constraints defined by equations 6 to 8 and the original constraints, $h_s(x)$, will be appended in the vector $h(x) : \mathbb{R}^n \mapsto \mathbb{R}^m$ with $n = 3n_s + m_2$, $m = 2n_s + m_1 + m_2$ and $x = \{x_s, s^U, s^L, w\}$. Using this new defined set of constraints and variables, the original problem posed in equation 5 can be redefined as the following (higher dimensionality) optimization problem:

$$\min_x f(x) \tag{9}$$

$$\text{subject to:}$$

$$h(x) = 0$$

with the Lagrangian of the problem defined by:

$$L(x, \mu) = f(x) + \lambda^T h(x) \tag{10}$$

A pair of points (x^*, λ^*) is said to be a stationary point of equation 10 if the following first order necessary conditions (Karush-Kuhn-Tucker conditions, KKT) are satisfied:

$$\nabla_x L(x^*, \lambda^*) = \nabla_x f(x^*) + J^T(x^*)\lambda^* = 0 \tag{11}$$

$$\nabla_\mu L(x^*, \lambda^*) = h(x^*) = 0 \tag{12}$$

where $\nabla_x f(x^*)$ is the gradient vector of the objective function ($n \times 1$ rows and columns), $h(x)$ the vector of equality constraints ($m \times 1$), λ the vector of Lagrange multipliers ($m \times 1$) and $J(x)$ the Jacobian matrix ($m \times n$). Furthermore, the second order sufficient conditions require that for some feasible point x^* and a Lagrange multiplier vector λ^*, if

1. linear independence constraint qualification (LICQ) holds at x^*, and
2. for any vector d satisfying $J(x^*)d = 0$ holds that: $d^T \nabla_{xx} L(x^*, \lambda^*) d > 0$,

then $x^*$is a strict local solution of 9 . In the following, it will be assumed that problem 9 show this properties.

Using a penalty function, the problem defined by equation 9 can be stated as:

$$\min_x P_{PP}(x; M) = f(x) + \frac{1}{2} M h(x)^T h(x) \tag{13}$$

where M is a positive penalty parameter. This is a classical approach used to solve the original NLP problem that is notorious for yielding badly conditioned unconstrained problems for conventional NLP solvers as M is increased [25].

In the following section, a gradient flow method with a novel self-tuning scheme for updating the value of the penalty parameter is presented.

2.2. A gradient flow formulation for constrained NLP problems

Considering the approach for the solution of unconstrained NLP problems represented by equation 1, the unconstrained formulation of the originally constrained NLP (equation 9) can be written as the following set of coupled ODEs:

$$\begin{aligned} \frac{dx}{dt} &= -\nabla_x P_{PP} \\ \frac{dx}{dt} &= -\left[\nabla_x f(x) + M\ J(x)^T h(x)\right]; \quad x(0) = x_0 \end{aligned} \tag{14}$$

where $0 \leq t \leq +\infty$.

In order for this scheme to be successful, the updating of the penalty parameter needs to be embedded in a coupled way within the GF scheme and the value of the penalty parameter can be linked to the constraint norm. Thus, considering a $\kappa > 0$ the value of the penalty can be formulated as

$$\frac{dM}{dt} = \kappa \left\|h(x)\right\|_2; \quad M(0) = M_0 \tag{15}$$

On the other hand, by considering a simple updating scheme according to the original penalty-multiplier approach (Hestenes method) [26], following the minimization at any iteration (k), if $\left\|h^{(k)}\right\| \geq \gamma \left\|h^{(k-1)}\right\|$ with $0 < \gamma < 1$, e.g. $\gamma = 0.25$, then the penalty parameter is increased by $M^{(k+1)} = \alpha \cdot M^{(k)}$. To derive a continuous variant, suitable for embedding in a GF methodology, the following algebraic steps are considered:

$$M^{(k+1)} - M^{(k)} = \kappa \cdot M^{(k)} - M^{(k)} \tag{16}$$

$$\Delta M^{(k)} = (\kappa - 1) \cdot M^{(k)} \tag{17}$$

with $(\kappa - 1) > 0$ and by renaming $(\kappa - 1) \to \alpha > 1$ in the limit it can be obtained that

$$\frac{dM}{dt} = \alpha M; \quad M(0) = M_0 \tag{18}$$

The scheme should allow the possibility of not increasing the penalty parameter if sufficient progress towards feasibility is made, so that the formulations in equations (15) and (18) may be combined as

$$\frac{dM}{dt} = \alpha M \left\|h(x)\right\|_2; \quad M(0) = M_0 \tag{19}$$

Since the value of M is updated within the integration of the ODE system, and embeds a type of exponential growth for M when the norm of the constraints is large, the above scheme will limit the necessary value for α. Combining the penalty parameter differential equation and the set of coupled ODEs representing the original GF approach, the following novel scheme is obtained (scheme PP)

$$\frac{dx}{dt} = -\left[\nabla_x f(x) + MJ(x)^T h(x)\right]; \quad x(0) = x_0 \tag{20a}$$

$$\frac{dM}{dt} = \alpha M^\beta \left\|h(x)\right\|_2; \quad M(0) = M_0 \tag{20b}$$

with $0 \leq t \leq +\infty$. The parameter β can take any positive value, or be set to zero, however some considerations are required. Since, the term M^β was included to act as an acceleration parameter of the trajectory of $M(t)$, it is convenient to analyze the behavior of the following equation

$$\frac{dM_p}{dt} = M_p^\beta; \quad M_p(0) = M_{p0} \tag{21}$$

for different values of β, where M_p represents a simplified version of the trajectory of the penalty parameter M. Clearly, this equation is a first order separable ODE. Its solution depends on the value of β according to

$$M_p(t) = \begin{cases} (1-\beta)^{1/(1-\beta)} \left[\frac{M_{p0}^{1-\beta}}{1-\beta} + t\right]^{1/(1-\beta)} & 0 \leq \beta < 1 \\ e^t & \beta = 1 \\ (\beta-1)^{1/(1-\beta)} \left[\frac{(\beta-1)M_{p0}^{\beta-1}}{1-(\beta-1)M_{p0}^{\beta-1}t}\right]^{1/(\beta-1)} & \beta \geq 1 \end{cases} \tag{22}$$

When $\beta \geq 1$, the values of M_{p0} are restricted to be lower than $[(\beta-1)t_f]^{(1-\beta)}$, where t_f is the final integration time, to avoid a zero value in the denominator of the equation defining $M_p(t)$. Figure 1 shows the trajectories of $M_p(t)$ for different values of β and a final integration time of 10 units. For $\beta = \{0, 0.5, 1\}$, M_{p0} was set to 1.0 while for values of β equal to 2 and 3, the value of M_{p0} was taken as $0.99 \cdot [(\beta-1)t_f]^{(1-\beta)}$. The trajectories for $\beta = \{2, 3\}$ show smaller values when compared to β equal to zero or greater. Since the purpose of this function is to increase the rate of growth of the penalty parameter (M), only β values of 0 and 1 will be considered for further analysis as they represent two extreme cases.

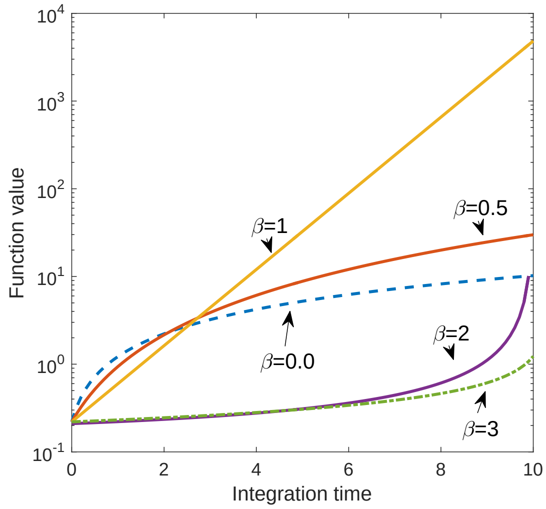

Figure 1: Trajectories of the acceleration parameter M_p as a function of time for different values of β.

2.2.1. Incorporating the Hestenes method to a gradient flow approach

The Hestenes multiplier scheme considers an Augmented Lagrangian for the NLP problem in equation 9 as follows [26]

$$\min_x P_{PM}(x; M, \lambda) = f(x) + \tilde{\lambda}^T h(x) + M\frac{1}{2}h(x)^T h(x) \tag{23}$$

Following minimization for a given value of $M = M^{(k)}$, the minimizer $x^*(M^{(k)})$ is obtained. The Lagrange multipliers are updated by

$$\Delta\tilde{\lambda}^{(k)} = M^{(k)} h(x^*(M^k)) \tag{24}$$

For this scheme it is known that there is a lower value of $\underline{\text{M}}$ for which if $M \geq \underline{M}$ it converges to the true solution of the NLP (x^*) [27]. Considering that the updating of the multipliers can be embedded also as an ODE in a GF scheme, the following formulation results (scheme PM)

$$\frac{dx}{dt} = -\left[\nabla_x f(x) + J(x)^T \left(\tilde{\lambda} + M\, h(x)\right)\right]; \quad x(0) = x_0 \tag{25a}$$

$$\frac{d\tilde{\lambda}}{dt} = M\, h(x); \quad \tilde{\lambda}(0) = \tilde{\lambda}_0 \tag{25b}$$

$$\frac{dM}{dt} = \alpha M^\beta \left\|h(x)\right\|_2; \quad M(0) = M_0 \tag{25c}$$

with $0 \leq t \leq +\infty$.

2.2.2. Proofs of convergence and stability analysis

In the following section, it will be proven that if x^* is a stationary point of the NLP defined by 9, then x^* is an equilibrium point of the PP scheme and vice versa (theorems 1 to 4). Theorems

5 and 6 show that the PM scheme converges to a stationary point of the original NLP problem, and that a solution of the aforementioned problem is a stationary point of the PM scheme. Finally, theorems 7 and 8 establish that the equilibrium points are asymptotically stable and theorems 9 and 10 show that the penalty functions used in both schemes define a strictly decreasing trajectory.

Theorem 1. *If $x^*$is the optimal solution to the unconstrained problem 13 for the penalty parameter M^*,then (x^*, M^*) is the equilibrium point for the system of ODEs defined by equations 20a and 20b with $\beta = 0$.*

Proof. According to the assumptions of the theorem, first order necessary conditions hold. Then,

$$\nabla_x P_{PP}(x^*; M^*) = -\frac{dx^*}{dt} = 0 \tag{26}$$

Furthermore, if x^* is the optimal solution of problem 13, it is also a stationary point of the NLP defined by 9 (see theorem 17.1 in Nocedal and Wright [28]). Thereby, $h(x^*) = 0$ and $\frac{dM}{dt} = \alpha \|h(x^*)\|_2 = 0$. Thus, (x^*, M^*) is the equilibrium point of the dynamic system formed by equations 20a and 20b for $\beta = 0$. □

Theorem 2. *If $x^*$is the optimal solution to the unconstrained problem 13 for the penalty parameter M^* and $M^* h(x^*) = \lambda^*$ is the vector of Lagrange multipliers at a stationary point of the NLP defined by 9, then $x^*$is the equilibrium point of equation 20a and the left hand side of equation 20b with $\beta = 1$ is constant.*

Proof. According to the assumptions of the theorem,

$$\nabla_x P_{PP}(x^*; M^*) = -\frac{dx^*}{dt} = \nabla_x f(x^*) + M^* J(x^*)^T h(x^*) = \nabla_x f(x^*) + J(x^*)^T \lambda^* = 0 \tag{27}$$

since the first order necessary conditions for problems 13 and 9 are satisfied. Now,

$$\lim_{t\to\infty} \frac{dM}{dt} = \lim_{t\to\infty} \alpha \sqrt{Mh(x)^T Mh(x)} = \alpha \lambda^{*T} \lambda^* \tag{28}$$

since according to theorem 17.2 in Nocedal and Wright [28], for a sufficiently large value of M^* the vector of Lagrange multipliers that satisfies the KKT conditions for the NLP defined by 9 can be approximated by $M^* h(x^*) = \lambda^*$. Thereby, x^* is the stationary point of equation 20a and the left hand side of equation 20b takes a constant value asymptotically. □

Theorem 3. *If (x^*, M^*) is the equilibrium point of the ODE system defined by equations 20a and 20b with $\beta = 0$, then $x^*$is an optimal solution of the unconstrained optimization problem defined by 13 and the NLP defined by 9.*

Proof. Since (x^*, M^*) is the equilibrium point of the ODE system 20a and 20b, then the first order necessary conditions of problem 13 are satisfied:

$$\nabla_x P_{PP}(x^*; M^*) = \nabla_x f(x^*) + M^* J(x^*)^T h(x^*) = 0 \tag{29}$$

$$\nabla_M P_{PP}(x^*; M^*) = \frac{1}{2} h(x^*)^T h(x^*) = 0 \tag{30}$$

satisfaction of $\nabla_x P_{PP}(x^*; M^*) = 0$ is clear since $\nabla_x P_{PP}(x^*; M^*) = -\dfrac{dx^*}{dt} = 0$. Satisfaction of equation 30 is ensured by:

$$\frac{dM}{dt} = \alpha \left\| h(x^*) \right\|_2 = \alpha \sqrt{h(x^*)^T h(x^*)} = 0 \tag{31}$$

The second order sufficient optimality condition requires that Hessian matrix of the penalty function defined by 13 to be positive definite:

$$\nabla_{xx} P_{PP}(x^*; M^*) = \nabla_{xx} f(x^*) + \sum_{i=1}^{m} M^* h_i(x^*) \nabla_{xx} h_i(x^*) + M^* J^T(x^*) J(x^*) \tag{32}$$

Replacing the Hessian matrix of problem 13 :

$$\nabla_{xx} P_{PP}(x^*, M^*) = \nabla_{xx} L(x^*, \lambda^*) + M^* J^T(x^*) J(x^*) \tag{33}$$

Now, consider a vector d satisfying $J(x^*)d = 0$, multiplying $\nabla_{xx} P_{PP}(x^*, M^*)$ at both sides:

$$d^T [\nabla_{xx} L(x^*, \lambda^*) + M^* J^T(x^*) J(x^*)] d = d^T \nabla_{xx} L(x^*, \lambda^*) d + d^T M^* J^T(x^*) J(x^*) d > 0 \tag{34}$$

since $M^* J^T(x^*) J(x^*)$ is positive definite and $d^T \nabla_{xx} L(x^*, \lambda^*) d > 0$ by assumption of the properties of problem 9. □

Theorem 4. *If x^* is the equilibrium point of the ODE system defined by equation 20a and the left hand side of equation 20b with $\beta = 1$ is constant, then x^* is an optimal solution of the unconstrained optimization problem defined by 13 and the NLP defined by 9.*

Proof. Since x^* is the equilibrium point of equation 20a, equation 29 is satisfied and remains to be proven that $\nabla_M P_{PP}(x^*, M)$ is equal to zero to show that the first order necessary conditions for problem 13 are satisfied. To do so, consider that:

$$\frac{dM}{dt} = \alpha M \left\| h(x^*) \right\|_2 = \alpha \sqrt{M h(x^*)^T M h(x^*)} \tag{35}$$

Then $\lim_{t\to\infty} M h(x^*) = \lambda^*$ and since $\lim_{t\to\infty} \dfrac{dM}{dt} = k$, where k is a positive real number, it follows that $\lim_{t\to\infty} h(x^*) = 0$. Thereby $\lim_{t\to\infty} \nabla_M P_{PP}(x; M) = \lim_{t\to\infty} \frac{1}{2} h(x^*)^T h(x^*) = 0$. Note that $\lim_{t\to\infty} \dfrac{dM}{dt} = k$ is required in order for $x^*$to be an equilibrium point of equation 20a and that the value of M used in this theorem is not an equilibrium value (M^*) since the growth rate $\frac{dM}{dt}$ is different than zero.

The satisfaction of the second order optimality conditions for this case is identical to Theorem 1, thus will be omitted. □

Theorem 5. *If $x^*$is the optimal solution to the unconstrained problem 23 for the penalty parameter $M^*$and multiplier $\tilde{\lambda}^*$, then $(x^*, \tilde{\lambda}^*, M^*)$ is the equilibrium point for the system of ODEs defined by equations 25a to 25c.*

Proof. According to the assumptions of the theorem, first order necessary conditions hold. Then,

$$\nabla_x P_{PM}(x^*, \tilde{\lambda}^*, M^*) = -\frac{dx}{dt} = 0 \tag{36}$$

Furthermore, if x^* is the optimal solution of problem 23, it is also a stationary point of the NLP defined by 9 (see theorem 17.5 in Nocedal and Wright [28]). Thereby, $h(x^*) = 0$ and $\dfrac{dM}{dt} = \dfrac{d\tilde{\lambda}}{dt} = 0$. Thus, $(x^*, \tilde{\lambda}^*, M^*)$ is the equilibrium point of the dynamic system formed by by equations 25a to 25c. By virtue of using the Lagrange multiplier method M goes to an equilibrium value M^*, as oppose to theorem 4. The same is true for theorem 6. □

Theorem 6. *If $(x^*, \tilde{\lambda}^*, M^*)$ is the equilibrium point of the ODE system defined by equations 25a to 25c, then $x^*$is an optimal solution of the unconstrained optimization problem defined by 23 and the NLP defined by 9.*

Proof. Since $(x^*, \tilde{\lambda}^*, M^*)$ is the equilibrium point of the ODE system25a to 25c, then the first order necessary conditions of problem 23 are satisfied:

$$\nabla_x P_{PM}(x^*, \tilde{\lambda}^*, M^*) = \nabla_x f(x^*) + J(x^*)^T \left(\tilde{\lambda}^* + M\, h(x^*)\right) = 0 \tag{37}$$

$$\nabla_M P_{PM}(x^*, \tilde{\lambda}^*, M^*) = \frac{1}{2} h(x^*)^T h(x^*) = 0 \tag{38}$$

$$\nabla_\lambda P_{PM}(x^*, \tilde{\lambda}^*, M^*) = h(x^*) = 0 \tag{39}$$

satisfaction of $\nabla_x P_{PM}(x^*, \tilde{\lambda}^*, M^*) = 0$ is clear since $\nabla_x P_{PM}(x^*, \tilde{\lambda}^*, M^*) = -\dfrac{dx^*}{dt} = 0$. Satisfaction of equations 38 and 39 are ensured by:

$$\frac{d\tilde{\lambda}^*}{dt} = M^* h(x^*) = 0 \tag{40}$$

$$\frac{dM}{dt} = \alpha M^\beta \left\|h(x)\right\|_2 = 0 \tag{41}$$

The second order sufficient optimality condition requires that Hessian matrix of the penalty function defined by 23 to be positive definite:

$$\nabla_{xx} P_{PM}(x^*, \tilde{\lambda}^*, M^*) = \nabla_{xx} f(x^*) + \sum_{i=1}^{m} [\tilde{\lambda}_i^* + M^* h_i(x^*)] \nabla_{xx} h_i(x^*) + M^* J^T(x^*) J(x^*) \tag{42}$$

According to theorem 17.2 in Nocedal and Wright [28], for a sufficiently large value of M^* the vector of Lagrange multipliers that satisfies the KKT conditions for the NLP defined by 9 can be approximated by $\tilde{\lambda}^* + M^* h(x^*) \simeq \tilde{\lambda}^* \simeq \lambda^*$. Then, replacing the Hessian matrix of problem 9:

$$\nabla_{xx} P_{PM}(x^*; M^*) = \nabla_{xx} L(x^*, \lambda^*) + M^* J^T(x^*) J(x^*) \tag{43}$$

which was shown to be positive definite in Theorem 1. Thereby, $x^*$is a strict local solution of 9. □

Theorem 7. (asymptotic convergence of PP scheme) *Assume that (x^*, λ^*) is a stationary point of equations 10. As usual, suppose LICQ holds at $x^*$and that for any vector d satisfying $J(x^*)d = 0$ holds that: $d^T \nabla_{xx} L(x^*, \lambda^*) d > 0$. Then, if x_0 is close enough to x^*, then $\lim_{t\to\infty} [x(t), M(t)h(t)] = (x^*, \lambda^*)$.*

Proof. For a system of ODEs represented by the following equation:

$$\frac{dx}{dt} = \phi(x) \tag{44}$$

the Poincaré-Lyapunov theorem (theorem 6.9 in Verhulst [29], page 191) states that if ϕ is continuously differentiable, it can be shown that x^* is an asymptotically stable point of 44 if all eigenvalues of the matrix $\nabla_x \phi(x*)$ have negative real value. From equation 20a, $\phi(x^*, M^*) = -\nabla_x P_{PP}(x^*, M^*)$ and $\nabla_x \phi(x^*, M^*) = -\nabla_{xx} P_{PP}(x^*, M^*) = -[\nabla^2_{xx} L(x^*, \lambda^*) + M^* J(x^*)^T J(x^*)]$.

We now show that the matrix $\nabla^2_{xx} L(x^*, \lambda^*) + M^* J(x^*)^T J(x^*)$ is positive definite. Let $\sigma_i \in \mathbb{R}$ be an eigenvalue of $\nabla^2_{xx} L(x^*, \lambda^*) + M^* J(x^*)^T J(x^*)$ and $z_i \in \mathbb{R}^n$ an eigenvector corresponding to σ_i. Then, σ_i and z_i satisfy

$$[\nabla^2_{xx} L(x^*, \lambda^*) + M^* J(x^*)^T J(x^*)] z_i = \sigma_i z_i \tag{45}$$

Multiplying by z_i^T by the left side

$$z_i^T [\nabla^2_{xx} L(x^*, \lambda^*) + M^* J(x^*)^T J(x^*)] z_i = \sigma_i \|z_i\|_2^2 \tag{46}$$

which yields,

$$\sigma_i = (z_i^T \nabla^2_{xx} L(x^*, \lambda^*) z_i + z_i^T M^* J(x^*)^T J(x^*) z_i) / \|z_i\|_2^2 \tag{47}$$

However, by the assumptions of the theorem $z_i^T \nabla^2_{xx} L(x^*, \lambda^*) z_i$ is positive definite, M is positive for every t and $J(x^*)^T J(x^*)$ is also positive definite (provided LICQ holds). Moreover, $J(x^*) z_i = 0$ if z_i satisfies the conditions of the theorem. Therefore, $\sigma_i > 0$ for all $i = 1, 2, ..., n$ and all eigenvalues of $\nabla_x \phi(x^*)$ are strictly negative, and thus by the Poincaré-Lyapunov theorem it follows that $\lim_{t \to \infty} x(t) - x^* = 0$. Finally, since (x^*, λ^*) is a stationary point of 10, it follows from theorems 3 and 4 that:

$$\nabla_x f(x^*) + M^* J(x^*)^T h(x^*) = \nabla_x f(x^*) + J(x^*)^T \lambda^* = 0 \tag{48}$$

and thereby $\lim_{t \to \infty} M(t) h(x) - \lambda^* = 0$ □

Theorem 8. (asymptotic convergence of PM scheme) *Assume that (x^*, λ^*) is a stationary point of equations 10. Suppose that LICQ holds at x^* and that for any vector d satisfying $J(x^*) d = 0$, $d^T \nabla_{xx} L(x^*, \lambda^*) d > 0$, where L is the Lagrangian function for the NLP problem defined by equation 9. If x_0 is close enough to x^*, then $\lim_{t \to \infty} [x(t), \tilde{\lambda}(t) + M(t) h(t)] = (x^*, \lambda^*)$.*

Proof. We start with equation 25a, defining $\phi_3(x, \tilde{\lambda}, M) = [\nabla_x f(x) + J(x)^T \left(\tilde{\lambda} + M h(x)\right)]$and calculating $\nabla^2_{xx} \phi_3(x^*, \tilde{\lambda}^*, M^*)$:

$$\nabla^2_{xx} \phi_3(x^*, \tilde{\lambda}^*, M^*) = -[\nabla^2_{xx} f(x^*) + \sum_{i=1}^{m} [\tilde{\lambda}_i^* + M^* h_i(x^*)] \nabla_{xx} h_i(x^*) + M^* J^T(x^*) J(x^*)] \tag{49}$$

From the Theorem 5 under the assumptions of Theorem 8, $h(x^*) = 0$, $\tilde{\lambda}^* \simeq \lambda^*$ and,

$$\nabla^2_{xx}\phi_3(x^*, \tilde{\lambda}^*, M^*) = -[\nabla_{xx} f(x^*) + \sum_{i=1}^{m} \tilde{\lambda}_i^* \nabla_{xx} h_i(x^*) + M^* J^T(x^*) J(x^*)] \tag{50}$$

$$\nabla^2_{xx}\phi_3(x^*, \tilde{\lambda}^*, M^*) = -[\nabla^2_{xx} L(x^*, \tilde{\lambda}^*) + M^* J^T(x^*) J(x^*)] \tag{51}$$

The right hand side of equation 51 was proven to be positive definite in Theorem 7. Recalling the definition of PM scheme (equation 25a):

$$\phi_3(x, \tilde{\lambda}, M) = \nabla_x P_{PM}(x^*, \tilde{\lambda}^*, M^*) = -\frac{dx^*}{dt} \tag{52}$$

and using the Poincaré-Lyapunov theorem it follows that for equation 25a, $\lim_{t\to\infty} x(t) - x^* = 0$. Finally, since (x^*, λ^*) is a stationary point of 10, it follows from theorems 5 and 6 that:

$$\nabla_x f(x^*) + J(x^*)^T \left(\tilde{\lambda^*} + M^* h(x^*)\right) = \nabla_x f(x^*) + J(x^*)^T \lambda^* = 0 \tag{53}$$

and thereby $\lim_{t\to\infty} \tilde{\lambda}(t) + M(t)h(t) - \lambda^* = 0$. □

Moreover, the following theorem indicates that $P_{PP}(x, M)$ (equation 13) and $P_{PM}(x, M, \lambda)$ (equation 23) are strictly decreasing along a trajectory of $x(t)$ that converges to x^*.

Theorem 9. *Let* $[x(t), M(t)]$ *be a solution trajectory of equations 25a and 20b. For a fixed* $t_0 \geq 0$, *if* $\nabla_x P_{PP}(x(t), M(t)) \neq 0$ *for all* $t > t_0$*, then* $P_{PP}(x(t), M(t))$ is strictly decreasing with respect to $t > t_0$.

In an analogous way the following theorem is defined.

Theorem 10. *Let* $[x(t), M(t), \tilde{\lambda}(t)]$ *be a solution trajectory of equations 25a to 25c. For a fixed* $t_0 \geq 0$*, if* $\nabla_x P_{PM}(x(t), M(t), \tilde{\lambda}(t)) \neq 0$ *for all* $t > t_0$*, then* $P_{PM}(x(t), M(t), \tilde{\lambda}(t))$ is strictly decreasing with respect to $t > t_0$.

Proof. Here the proof of theorem 9 is presented. Proof of theorem 10 is analogous and is thus omitted. From equation 20a, $\frac{dx}{dt} = -\phi_{PP}(x(t))$ and the trajectory of $P_{PP}(x(t))$ can be calculated from

$$\frac{dP_{PP}(x(t))}{dt} = \frac{dP_{PP}(x(t))}{dx}\frac{dx}{dt} \tag{54}$$

$$\frac{dP_{PP}(x(t))}{dt} = -\|\phi_{PP}(x(t))\|_2^2 \tag{55}$$

Since $\phi_{PP}(x(t))$ is different than zero when $t > t_0$, then it can be concluded that $\frac{dP_{PP}(x(t))}{dt} < 0$. Thereby, $P_{PP}(x(t))$ is strictly decreasing with respect to $t > t_0$. □

2.3. Comparison with previously reported GF formulations

As stated in the introductory section, the GF approach proposed in this work, unlike previously reported methods, is an infeasible path method. To analyze the reasons behind this behavior, consider the solution of the ODE system represented by equations 25a to 25c. After multiplying equation 25a by $J(x)$ we obtain

$$J(x)\frac{dx}{dt} = \frac{dh(x)}{dt} = -J(x)[\nabla_x f(x) + J(x)^T\left(\tilde{\lambda} + Mh(x)\right)] \Rightarrow$$
$$\frac{dh(x)}{dt} + J(x)J(x)^T Mh(x) = -J(x)[\nabla_x f(x) + J(x)^T\tilde{\lambda}]$$

The last equation corresponds to a linear differential equation in $h(x)$ with variable coefficients. Thus, defining $\nu(x(t)) = J(x(t))J(x(t))^T M$ and $\omega(x) = -J(x)[\nabla_x f(x(t)) + J(x(t))^T\tilde{\lambda}]$ and using an appropriate integration factor, the trajectory of $h(x)$ can be implicitly expressed as

$$h(x(t)) = e^{-\int_0^t \nu(x(t))dt}\left[\int\limits_0^t \omega(x(t))e^{\int_0^t \nu(x(t))dt}dt + C\right]$$

Now, suppose that for some $t_0 > 0$, $h(x(t_0)) = 0$. Then,

$$e^{-\int_0^{t_0} \nu(x(t))dt}\left[\int\limits_0^{t_0} \omega(x(t))e^{\int_0^{t_0} \nu(x(t))dt}dt + C\right] = 0 \Rightarrow$$
$$C = -\int\limits_0^{t_0} \omega(x(t))e^{\int_0^{t_0} \nu(x(t))dt}dt$$

Therefore, for $t > t_0$:

$$h(x(t)) = e^{-\int_{t_0}^t \nu(x(t))dt}\left[\int\limits_{t_0}^t \omega(x(t))e^{\int_{t_0}^t \nu(x(t))dt} + C\right] \tag{56}$$

and the only possibility for $h(x(t))$ to be zero for $t > t_0$ is that both C and $\omega(x(t))$ are zero for $t > t_0$, implying that also $\omega(x(t_0))$ needs to be equal to zero. These conditions can be satisfied if at time t_0 not only $h(x(t_0)) = 0$, but also $\nabla_x f(x(t)) + J(x(t))^T\tilde{\lambda} = 0$. Clearly, a point satisfying both conditions will also be the optimal solution of the problem. Thereby, unlike the formulation presented by Wang et al. [23], once the ODE system reaches a feasible point, it will generally not remain feasible.

Unlike previously proposed formulations based on the Gradient Flow approach, our GF scheme does not requires the calculation of inverse matrices (such as $J(x)J^T(x)$, see equations 2 and 4). Thereby, each integration step taken using the GF formulations presented in this work is less computationally demanding compared, for example, to the GF formulation presented by Wang et al. [23]. This is confirmed by the results presented in the next section. Moreover, although the squared slacks used in equations 6 to 8 may cause the Jacobian to be linearly dependent, this will occur at the point where the constraints are satisfied as equalities with zero slack, which in turns only occurs at the solution since the proposed approach is an infeasible path method as shown previously. Therefore, the use of squared slacks does not pose a problem for the GF schemes proposed in this work.

2.4. Implementation

The original NLP problems (equation 5) were automatically converted to a system of differential equations (PP, equations 20a and 20b, and PM, equations 25a to 25c) using a code developed in Wolfram Mathematica™ 10.3 that takes full advantage of this Computer Algebra System (CAS).

The ODE equations are processed into a format suitable for Mathematica's built-in differential equation solver `NDSolve` using an open package developed in Mathematica™ with flexible data structures that also allows system structural analysis [30]. `NDSolve` options were used with default values except for AccuracyGoal and PrecisionGoal which were increased to tighten constraint satisfaction when required. The option WhenEvent in NDSolve was used to reset the value of the penalty parameter when it exceeds a large value (10^{12}). Of course, this approach is only implemented for the formulation that incorporates multipliers (PM) since the formulation PP requires a large penalty value to achieve convergence. For the case study Problem 6, MATLAB™ `ODE15s` was used to obtain the solution of the problem when the PM formulation was tested. MATLAB™ was used since it allows tailoring the execution of the integration. Specifically, for large problems RAM memory usage was limited by using short integration steps, storing the solution at the final integration time ($x(t_f)$) and reinitializing the integration using the stored values ($x_0 = x(t_f)$).

The ODE systems produced by PP and PM formulations are integrated until the merit function defined by equation 57 reaches a value below a prescribed tolerance equal to 10^{-6}, unless otherwise stated. This value was chosen to be similar to the default tolerances used by CONOPT and IPOPT, $10^{-7}$and 10^{-6}, respectively.

$$Merit\, function = \|\nabla_x f(x) + J^T(x)\mu\|_2 + \|h(x)\|_2 \tag{57}$$

where μ was calculated as $\mu^* = M^* h(x^*)$ for PM formulations and as $\tilde{\lambda^*}$ for PP formulations. Finally, for the GF formulation *PP,* integration was stopped if the value of the penalty parameter M exceeds 10^{12} to avoid numerical problems (overflow).

The GF formulations proposed in this work are compared against the formulations reported in Wang et al. [23] using two approaches: WM and WA, For problems with a small number of variables (problems 1 and 2 in Section 3) a calculation procedure termed WM (Wang's Method) was implemented in Mathematica™. In this scheme, matrix $J(x)^T J(x)$ in equation 4 is symbolically inverted and the resulting function is stored in Mathematica™ for its use by NDSolve. Therefore, matrix $J(x)^T J(x)$ is not inverted at each integration step but the stored function is used to evaluate it. Moreover, the CPU time reported for the WM scheme, as for other GF schemes, corresponds to the CPU time consumed by the execution of NDSolve, thus for the WM scheme it does not include the time required to calculate the inverse of $J(x)^T J(x)$ (symbolically).

As the number of variables increases (problems 3 to 7 in Section 3), the time required to symbolically invert the matrix $J(x)^T J(x)$ in WM approach rises to impractical levels. Therefore, in order to compare the GF approaches presented in this paper with the method proposed by Wang et al. [23] a new approach was required. This approach, termed WA (Wang's Algorithm) was also presented in Wang et al. [23] and is based in the discretization of equation 4 using the implicit backward Euler's scheme. According to this algorithmic scheme, the matrix $J(x)^T J(x)$ is not symbolically inverted but in each step $[J(x)^T J(x)]^{-1}$ is numerically calculated.

The GF schemes presented in this work and WM are compared in terms of CPU times, number of integration steps and function evaluations used by NDSolve to achieve the same merit function value. When WA and conventional NLP solvers (CONOPT, MINOS, SNOPT and IPOPT) are used, CPU times and number of iterations are compared.

3. Case studies

The new Gradient Flow schemes were numerically tested against five standard constrained nonlinear problems and two nonlinear problems derived from optimal control problems, all obtained from the open literature.

3.1. Constrained non-linear problems

3.1.1. Problem 1

Problem 1 corresponds to a constrained nonlinear optimization problem with a nonlinear objective function, linear constraints and variable bounds. Its optimal solution is $x = \{0, 1, 2, 0\}$ with an objective function value equal to -1.5. Problem 1 is a convex one with linear constraints; thereby, its difficulty is low and a global solution is expected, both for conventional NLP solvers and for the GF schemes presented in this work.

$$
\begin{aligned}
&\min_x 1.5x_1^2 - x_1x_2 + 1.5x_2^2 + x_1 - 3x_2 \\
\text{subject to:}& \\
&-x_1 + 2x_2 + x_3 = 4 \qquad (58) \\
&x_1 + x_2 + x_4 = 1 \qquad (59) \\
&\qquad (60) \\
&0 \le x_i \le 10;\ i = \{1, ..., 4\}
\end{aligned}
$$

Problem 1 was solved using $x = [2, 1, 3, 4]$ as initial values and with $\alpha = 10^6$ for scheme *PM* with $\beta = 0$, and $\alpha = 10^3$ for scheme *PM* with $\beta = 1$ and for scheme *PP*. Table 1 shows a summary of the numerical results when the integration was stopped after a merit function value equal or lower than 10^{-6} was achieved for every GF formulation. Every solver tested, including this work's and Wang's formulation [23], finds the optimal solution of the problem. CPU time and RAM memory usage obtained using GF schemes are competitive with conventional solvers. Using an identical termination criteria, the GF formulation proposed by Wang and coworkers with their tuning parameter set to $\tau = 1$ [23] requires nearly 129 times more CPU time compared to the GF formulations presented in this work. This difference increases as τ increases, with 23 CPU seconds for $\tau = 1000$, due to the stiffness of the problem that forces the integrator to use small integration steps. The quality of the solution, in terms of constraint satisfaction and objective function is similar for the conventional solvers and the GF formulations.

Table 1: Solution summary for Problem 1 including results obtained using commercial optimization solvers and the GF formulations proposed in this work.

	IPOPT	CONOPT	MINOS	SNOPT	$PP_{(\beta=0)}$	$PP_{(\beta=1)}$	$PM_{(\beta=0)}$	$PM_{(\beta=1)}$	WM ($\tau = 1$)
CPU (s)	0.11	0.02	0.08	0.05	0.02	0.02	0.02	0.03	2.37
Memory (Mb)	3.0	2.0	3.0	3.0	1.1	1.0	1.8	2.0	11.2
Nfun	89	NA^a	4	2	1183	1384	914	1639	1172
Iterations	15	9	2	1	-	-	-	-	-
Integration steps	-	-	-	-	510	511	490	546	757
Obj	-1.50	-1.50	-1.50	-1.50	-1.50	-1.50	-1.50	-1.50	-1.50
	$7.1 \cdot 10^{-9}$	0	0	$2.2 \cdot 10^{-16}$			$2.7 \cdot 10^{-15}$	$3.4 \cdot 10^{-12}$	$6.8 \cdot 10^{-12}$
Merit function	NA^b	NA^b	NA^b	NA^b	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$

^{a}Not reported by GAMS CONOPT. ^{b}Default values in GAMS were used

The effect of varying the value of α in the PP, PM and Wang's (increasing τ) formulations [23] is shown in Figure 2, indicating that the satisfaction of the constraints is largely independent of the value of parameters α and τ for this problem (where only linear constraints are present). However, as the value of these parameters increases the system of differential equations becomes stiff, requiring more integration steps (and computational time) to achieve the same norm of the constraints. This is evident in Wang's formulation (WM) where the merit function value oscillates for $\tau = 10^6$. Since in PP and PM formulations the value of the penalty parameter M is tied to the violation of the constraints, it is not necessary to use large values of α, especially with the PM formulation where multipliers are also used.

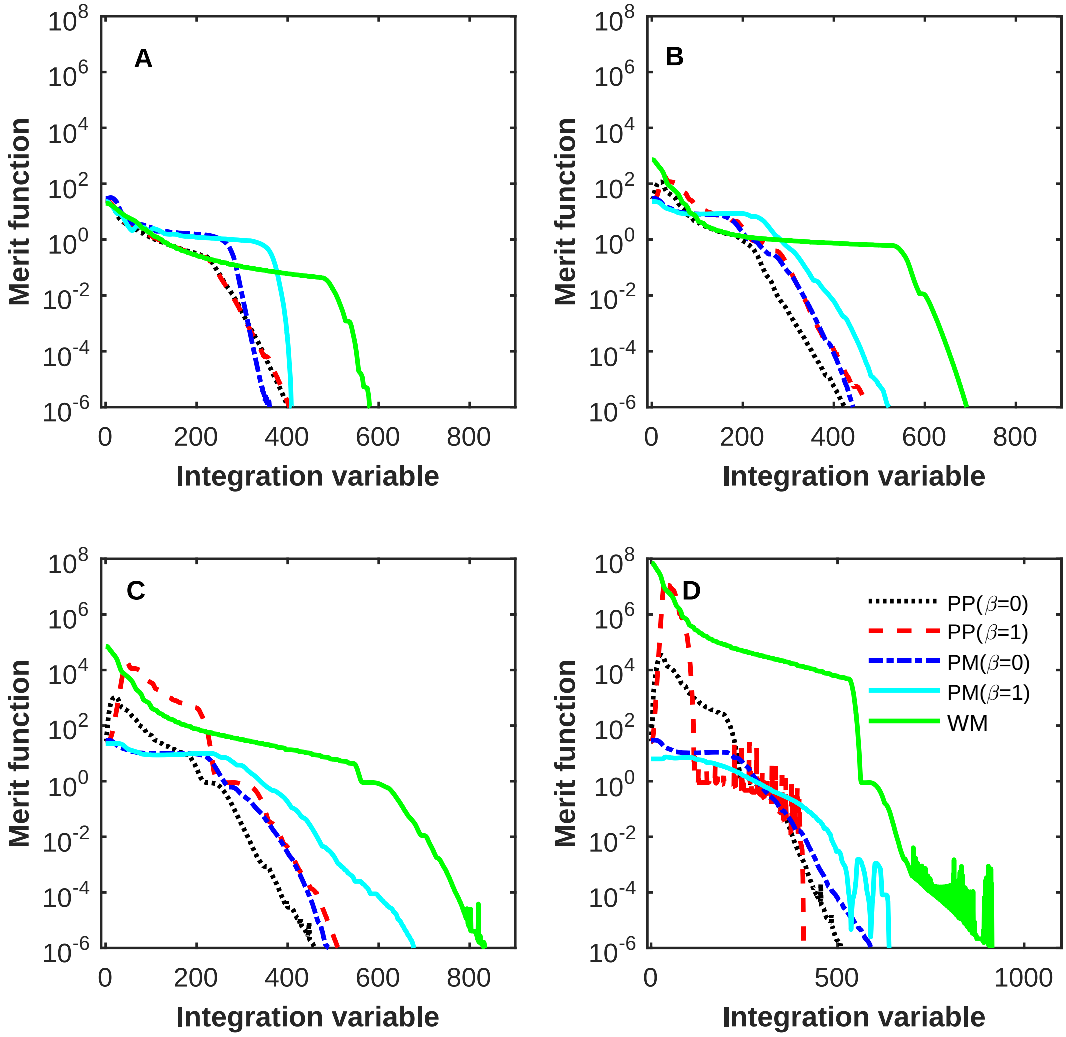

Figure 2: Norm of the constraints vector as the integration proceeds for Problem 1 and GF formulations using a penalty parameter scheme (PP) or a penalty and multipliers scheme (PM). For panels A, B, C and D the value of α is 10^{-2},1, 10^3 and 10^6 respectively.

3.1.2. Problem 2

This problem has been taken from Example 6.8 in Biegler [25], and it was designed to challenge Newton-based interior point methods when starting from an infeasible point. The problem has only one solution at $x^* = [1, 0, 0.5]$.

$$
\begin{aligned}
&\min_{x} x_1 && (61)\\
&\text{subject to:}\\
&x_1^2 - x_2 - 1 = 0\\
&x_1 - x_3 - 0.5 = 0\\
\\
&-10 \leq x_1 \leq 10\\
&0 \leq x_i \leq 10;\ i = \{2, 3\}
\end{aligned}
$$

The infeasible initial point for this problem is $x = [-2, 3, 1]$ and for the GF formulations an α value of 10^6, for PM with $\beta = 0$, and 10^3 for the rest of the formulations was chosen. Table 2 shows the numerical results for this problem where the merit function satisfaction was set to 10^{-6} for GF based formulations (including the one proposed by Wang et al. [23]). As expected, IPOPT fails in finding the optimal solution, but CONOPT, MINOS and SNOPT provide the solution. The solutions provided by GF schemes are optimal, show excellent constraint satisfaction and are obtained using less RAM memory and CPU time compared to the conventional NLP solvers. On the other hand, the GF formulation proposed by Wang et al. [23] reaches the limit of iterations (10^5) without achieving the solution of the problem when $\tau = 1$, finds the solution for $\tau = 1000$ and fails again for a larger value of this parameter. This result is indicative of the importance of the selection of the τ value for NLP problems with nonlinear constraints, unlike the situation shown for Problem 1, where τ values only affect the stiffness of the problem. The independence to the α value shown by the algorithms presented in this paper is a consequence of the self-tuning behavior of the penalty parameter in the GF schemes presented in this work. On the contrary, Wang et al. [23] reports linear rate of convergence for small values of the penalty parameter and quadratic rate of convergence for large values. Thereby, in their approach the value of the penalty parameter needs to be tuned carefully for each problem to obtain a quadratic convergence and to avoid a stiff problem that prevents finding a solution.

3.1.3. Problem 3

Problem 3 corresponds to the flowsheet optimization problem of the Williams-Otto process [31], adapted from Biegler [25]. The process flowsheet is shown in Figure 3 where also the reaction network for the synthesis of P (main product), E (by-product) and G (waste product) is presented. Two feed streams with pure A and B components (streams F_A and F_B) are fed to a stirred tank reactor whose operating temperature, T, is subject to optimization. The effluent stream is cooled and sent to a centrifuge to separate G (in stream F_G). The clarified stream is fed to a column separator to recover P where 90% of the product P is recovered in the column's top stream. This stream is separated into purge (F_{purge}) and a recycled stream (F_R) that is recycled to the reactor.

The optimization problem is represented by equations 62 to 74. Variable bounds, initial values, optimal values and the values obtained using formulation PM with $\beta = 0$ are shown in Table 3.

Table 2: Solution summary for Problem 2.

	IPOPT	CONOPT	MINOS	SNOPT	$PP_{(\beta=0)}$	$PP_{(\beta=1)}$	$PM_{(\beta=0)}$	$PM_{(\beta=1)}$	WM ($\tau = 1$)	WM ($\tau = 10^3$)	WM ($\tau = 10^7$)
CPU (s)	0.39	0.18	0.28	0.23	0.02	0.01	0.02	0.018	65.3	0.60	0.2
Memory (Mb)	1.3	0.7	2.1	1.2	1.4	0.9	1.7	2.0	96.9	2.2	1.7
Nfun	89	NA	36	16	1239	873	936	873	225694	4131	2504
Iterations	13	7	7	3	-	-	-	-	-	-	-
Integration steps	-	-	-	-	754	507	554	623	10^5	1912	1162
Obj	INF^b	1.00	1.00	1.00	1.00	1.00		1.00	0.883	1.00	0.885
$\|h(x)\|$	1.5	0	$3.3 \cdot 10^{-9}$	$2.4 \cdot 10^{-9}$	$6.9 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$1.4 \cdot 10^{-9}$	$6.3 \cdot 10^{-12}$	0.221	$9.2 \cdot 10^{-10}$	0.221
Merit Function	NA^b	NA^b	NA^b	NA^b	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	0.377	$1.0 \cdot 10^{-6}$	$3.9 \cdot 10^{10}$

^{a}Not reported by GAMS CONOPT. ^{b}Default values in GAMS were used

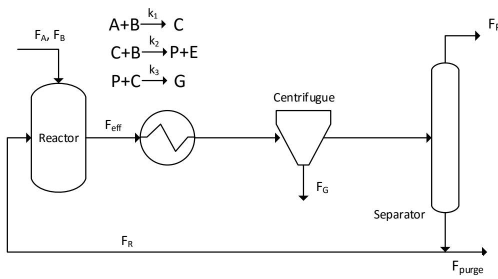

Figure 3: Flowsheet for Problem 3, the Williams-Otto process.

$$\min_{x} -\frac{2207 \cdot F_P + 50 \cdot F_{purge} - 168 \cdot F_A - 252 \cdot F_B - 2.22 \cdot F_{eff}^{sum} - 84 \cdot F_G - 60 \cdot V \cdot \rho}{6 \cdot \rho \cdot V} \tag{62}$$

subject to:

$$k_1 - 5.9755 \cdot 10^9 e^{-\frac{120}{T}} = 0 \tag{63}$$

$$k_2 - 2.5962 \cdot 10^{12} e^{-\frac{150}{T}} = 0 \tag{64}$$

$$k_3 - 9.6283 \cdot 10^{15} e^{-\frac{200}{T}} = 0 \tag{65}$$

$$F_{eff}^P - 0.1 F_{eff}^E - F_P = 0 \tag{66}$$

$$F_A + F_B - F_G - F_P - F_{purge} = 0 \tag{67}$$

$$\frac{-k_1 F_{eff}^A F_{eff}^B V \rho}{(F_{eff}^{sum})^2} - \frac{F_{purge} F_{eff}^A}{F_{eff}^{sum} - F_G - F_P} + F_A = 0 \tag{68}$$

$$\frac{(-k_1 F_{eff}^A F_{eff}^B - k_2 F_{eff}^B F_{eff}^C) V \rho}{(F_{eff}^{sum})^2} - \frac{F_{purge} F_{eff}^B}{F_{eff}^{sum} - F_G - F_P} + F_B = 0 \tag{69}$$

$$\frac{((2k_1 F_{eff}^A - k_2 F_{eff}^C) F_{eff}^B - k_3 F_{eff}^C F_{eff}^P) V \rho}{(F_{eff}^{sum})^2} - \frac{F_{purge} F_{eff}^C}{F_{eff}^{sum} - F_G - F_P} = 0 \tag{70}$$

$$\frac{2k_2 F_{eff}^B F_{eff}^C V \rho}{(F_{eff}^{sum})^2} - \frac{F_{purge} F_{eff}^E}{F_{eff}^{sum} - F_G - F_P} = 0 \tag{71}$$

$$\frac{(k_2 F_{eff}^B - 0.5 k_3 F_{eff}^P) F_{eff}^C V \rho}{(F_{eff}^{sum})^2} - \frac{F_{purge}(F_{eff}^P - F_P)}{F_{eff}^{sum} - F_G - F_P} - F_P = 0 \tag{72}$$

$$\frac{-1.5 k_3 F_{eff}^C F_{eff}^P V \rho}{(F_{eff}^{sum})^2} - F_G = 0 \tag{73}$$

$$F_{eff}^A + F_{eff}^B + F_{eff}^C + F_{eff}^E + F_{eff}^P + F_G - F_{eff}^{sum} = 0 \tag{74}$$

Table 3: Bounds, initial conditions, optimal solution and solution obtained using formulation $PM(\beta = 0)$ for Problem 3 .

Variable	x^L	x^U	x_0	x^*	$x^*_{PM(\beta=0)}$	Variable	x^L	x^U	x_0	x^*	$x^*_{PM(\beta=0)}$
F_{eff}^{sum}	0.0	1000	52	366.369	370.523	F_{purge}	0.0	100	0.1	35.910	36.126
F_{eff}^{A}	0.0	100	10	46.907	43.675	V	0.03	0.1	0.06	0.03	0.03
F_{eff}^{B}	0.0	500	30	145.444	149.517	F_A	0	100	30	13.357	13.170
F_{eff}^{C}	0.0	100	3	7.692	6.989	F_B	0.0	100	20	30.442	31.071
F_{eff}^{E}	0.0	1000	3	144.033	147.479	T	2	6.8	5.8	6.744	6.782
F_{eff}^{P}	0.0	100	5	19.115	19.511	k_1	0.0	200	6.2	111.7	123.6
F_P	0	4.763	0.5	4.712	4.763	k_2	0	1000	15.2	567.6	643.8
F_G	0.0	100	1	3.178	3.352	k_3	0	1500	10.2	1268.2	1500

Computational results obtained using conventional NLP solvers and the GF formulations presented in this work are shown in Table 4. Commercial solver CONOPT achieves a feasible local optimum, while the commercial solver MINOS reports the problem as infeasible. On the other hand, GF formulations achieve the demanded value for the merit function ($\leq 10^{-6}$), although only local minima solutions are attained. However, this is not unexpected since the GF formulations do not incorporate provisions to achieve global optima. Using the algorithmic version of the formulation proposed by Wang and coworkers (WA), an algorithm based on the use of the implicit Euler method, with a small integration step of 0.01, the algorithm reaches a merit function value of $2.75 \cdot 10^{25}$ in four iterations producing numerical errors . The solution of this problem using Wang's method (WM) implemented in NDSolve (as in Problems 1 and 2) was also not possible, since this requires the calculation of a 44×44 inverse matrix with symbolic entries which proves extremely time consuming.

Table 4: Solution summary for Problem 3.

	IPOPT	CONOPT	MINOS	SNOPT	$PP_{(\beta=0)}$	$PP_{(\beta=1)}$	$PM_{(\beta=0)}$	$PM_{(\beta=1)}$
CPU (s)	0.4	0.25	0.26	0.61	54.8	97.8	0.55	8.8
Memory (Mb)	0.4	1.8	2.1	1.6	31.5	39.6	21.2	22.7
Nfun	152	NA	385	4406	12692		936	7866
Iterations	33	71	42	233	-	-	-	-
Integration steps	-	-	-	-	4751	6490	1151	2507
Obj	-121.1	10.0	INF[b]	-121.1	-118.9	-118.9	-120.2	-71.3
$\|\|h(x)\|\|$	$6.4 \cdot 10^{-7}$	0	2.9	$1.1 \cdot 10^{-12}$	$8 \cdot 10^{-6}$	$1.9 \cdot 10^{-6}$	$1.6 \cdot 10^{-9}$	$3.7 \cdot 10^{-11}$
Merit Function	NA[c]	NA[c]	NA[c]	NA[c]	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$

[a]Not reported by GAMS CONOPT. [b]Solver reports an infeasible solution. [c]Default values in GAMS were used

Using the solution obtained with $PM(\beta = 1)$ scheme as a starting point, CONOPT declares the initial point as feasible after two iterations and finds the optimal solution in 18 iterations. Starting from the solution given by $PM(\beta = 0)$, CONOPT reports a feasible solution in two iterations and an optimal solution in 19 iterations. Although, MINOS fails in finding a feasible solution from the starting point given in Table 3, it reports an optimal solution after 12 iterations starting from the solution provided by $PM(\beta = 1)$ and in 10 iterations when starting from $PM(\beta = 0)$.

Therefore, the GF formulations presented in this paper result useful as an initialization method for this highly infeasible problem. It is important to stress that the commercial solvers only fail in the initial point reported in Table 3 and in other highly infeasible starting points, while they are able to solve the problem to optimality for most initial points.

3.1.4. Problem 4

This problem corresponds to the simplified alkylation process presented in Berna et al. [32], including 14 continuous variables, 1 linear constraint, 5 nonlinear constraints and a nonlinear objective function. Bounds for variables are shown in Table 5.

$$\min_x -6.3x_4x_7 + 5.04x_1 + 0.35x_2 + x_3 + 3.36x_5 \tag{75}$$

subject to:

$$x_4 - (x_1 + x_5)/1.22 = 0 \tag{76}$$

$$0.98x_3 - x_6(\frac{x_4x_9}{100} + x_3) = 0 \tag{77}$$

$$10x_2 + x_5 - x_1x_8 = 0 \tag{78}$$

$$x_4x_{11} - x_1(1.12 + 0.13167x_8 - 0.0067x_8^2) = 0$$

$$0.8635 + \frac{(1.098x_8 - 0.038x_8^2)}{100} + 0.325(x_6 - 0.89) - x_7x_{12} = 0 \tag{79}$$

$$35.82 - 22.2x_{10} - x_9x_{13} = 0 \tag{80}$$

$$-1.33 + 3x_7 - x_{10}x_{14} = 0 \tag{81}$$

Table 5: Bounds and initial conditions for Problem 4.

	x^L	x^U		x^L	x^U
x_1	0.0	2	x_8	3.0	12.0
x_2	0.0	1.6	x_9	1.2	4.0
x_3	0.0	1.2	x_{10}	1.45	1.62
x_4	0.0	5	x_{11}	0.99	1.01
x_5	0.0	2	x_{12}	0.99	1.01
x_6	0.85	0.93	x_{13}	0.90	1.11
x_7	0.90	0.95	x_{14}	0.99	1.01

Problem 4 can be solved to optimality by every conventional NLP solver, and also by all the GF formulations proposed in this work (see Table 6) with demanded values of the merit function below 10^{-5} and $10^{-6}$for PP and PM formulations. CPU time are smaller and RAM memory usage are generally larger for GF formulations, except when compared to IPOPT.. As shown in Table 6, the algorithm proposed by Wang and coworkers fails in achieving a solution, even for a small value of the integration step.

Table 6: Solution summary for Problem 4.

	IPOPT	CONOPT	MINOS	SNOPT	$PP_{(\beta=0)}$	$PP_{(\beta=1)}$	$PM_{(\beta=0)}$	$PP_{(\beta=1)}$	WA ($h=10$)
CPU (s)	0.3	0.29	0.29	0.25	0.45	0.13	0.15	0.25	1612.08
Memory (Mb)	22.9	1.7	2.1	1.6	6.5	5.6	13.5	11.5	0.2
Nfun	79	NA[a]	316	192		1981	1955	2785	-
Iterations	15	19	13	22	-	-	-	-	10^4
Integration steps	-	-	-	-	1081	835	1109	1070	-
Obj	-1.765	-1.765	-1.765	-1.765	-1.765	-1.765	-1.765	-1.765	$-9.7 \cdot 10^6$
$\|h(x)\|$	$8.3 \cdot 10^{-9}$	$4.7 \cdot 10^{-9}$	$2.5 \cdot 10^{-11}$	$4.3 \cdot 10^{-8}$	$6.3 \cdot 10^{-5}$	$8.2 \cdot 10^{-7}$	$1.9 \cdot 10^{-9}$	$6.4 \cdot 10^{-12}$	$5.6 \cdot 10^6$
Merit Function	NA[b]	NA[b]	NA[b]	NA[b]	$1.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.3 \cdot 10^7$

[a]Not reported by GAMS CONOPT. [b]Default values in GAMS were used

3.1.5. Problem 5

Problem 5 was adapted from Wang et al. [23], where only seven variables were considered. In this work, the problem was modified to accommodate an arbitrary number of variables (n_v) while maintaining its qualification as a constrained concave programming problem. The problem has multiple local minima and a global minimum of -1.0.

$$\min_x - \sum_{i=1}^{n_v} x_i^2 \tag{82}$$

subject to:

$$\sum_{i=1}^{n_v} x_i - 1 + slack = 0 \tag{83}$$

$$0 \leq x_1 \leq 0.8 \tag{84}$$

$$slack \geq 0 \tag{85}$$

$$0 \leq x_i \leq 1; \; i = \{2, ..., n_v\}$$

For all solvers, the initial point was taken as $x_i = 0.5$, $i = \{1, ..., n_v\}$. Table 7 shows the solution of the problem for 50 to 600 variables using conventional NLP solvers and the GF formulations introduced in this work when integration was stopped after the merit function achieves values lower than 10^{-6}. Using the algorithm presented by Wang et al. [23], only problems with 50 and 100 variables were solved in less than 3600 CPU seconds. Every conventional solver reports a feasible solution, however, all fail to find the global minimum as they are all local solvers. Although the GF formulations presented in this work find the globally optimal solution of this problem with multiple local minima, there is no theoretical reason to support this behavior. Moreover, using a different starting point the commercial solvers also achieve the global solution for this problem.

The CPU time required by GF formulations is competitive with the commercial solvers for 50 and 100 variables but for a large number of variables, commercial solvers find a local optima for the problem using less CPU time and RAM memory. Considering that the commercial solvers represent the state of the art , rely on extensive preprocessing of the problem to achieve an efficient solution and are coded on a faster platform, this is an expected result.

We point the reader's attention to the fact that the GF formulation proposed in this work are implemented using Mathematica NDSolve, a general purpose algebraic solver. This explains the increase in RAM memory usage, as NDSolve stores the solution of the problem as polynomial splines. Clearly, an algorithmic implementation of the GF formulations, where the trajectories are not stored, will consume less RAM memory. Despite the difference in performance as the number of variables increase, the GF formulations presented in this work achieve optimal solutions with sharp constraint satisfaction.

As shown in Table 7, the algorithmic implementation of the GF formulation proposed by Wang and coworkers also achieves an optimal solution, however the CPU times required are several hundred times larger.

Table 7: Solution summary for Problem 5 .

	$n_v = 50$	$n_v = 100$	$n_v = 300$	$n_v = 600$		$n_v = 50$	$n_v = 100$	$n_v = 300$	$n_v = 600$
IPOPT, CPU(s)	0.32	0.33	0.34	2.3	MINOS, CPU(s)	0.18	0.17	0.18	0.19
Memory (Mb)	30.9	30.4	13.1	30.3	Memory (Mb)	0.8	2.2	2.3	2.8
Nfun	104	79	69	74	Nfun	49	99	299	599
Obj	-0.6408	-0.6404	$-3.4 \cdot 10^{-3}$	$-1.7 \cdot 10^{-3}$	Obj	0.0000	0.0000	0.0000	0.0000
$\Vert h(x) \Vert$	$2.6 \cdot 10^{-10}$	$3.0 \cdot 10^{-11}$	$2.1 \cdot 10^{-10}$	$5.4 \cdot 10^{-9}$	$\Vert h(x) \Vert$	0	0	0	0
CONOPT, CPU(s)	0.24	0.29	0.27	0.31					
Memory (Mb)	1.8	1.9	2.4	3.4					
Obj	-0.5000	-0.5000	-0.5000	-0.5000					
$\Vert h(x) \Vert$	0	0	0	0					
$PP(\beta = 0)$, CPU(s)	0.55	1.8	18.8	49.5	$PP(\beta = 1)$, CPU(s)	0.72	2.11	21.6	76.0
Memory (Mb)	21.1	52.3	194.3	327.4	Memory (Mb)	31.6	64.4	225.4	432.3
Nfun	2489	2965	3809	2989	Nfun	2682	2785	3647	3567
Iterations	1058	1266	1571	1305	Iterations	1379	1501	1814	1780
Obj	-1.0000	-1.0000	-1.0000	-1.0000	Obj	-1.0000	-1.0000	-1.0000	-1.0000
$\Vert h(x) \Vert$	$1.8 \cdot 10^{-6}$	$9.3 \cdot 10^{-8}$	$3.1 \cdot 10^{-8}$	$3.5 \cdot 10^{-7}$	$\Vert h(x) \Vert$	$1.6 \cdot 10^{-8}$	$8.7 \cdot 10^{-9}$	$1.8 \cdot 10^{-9}$	$1.5 \cdot 10^{-9}$
Merit Function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	Merit Function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$6.9 \cdot 10^{-7}$	$1.0 \cdot 10^{-6}$
$PM(\beta = 0)$, CPU(s)	0.54	1.8	19.2	93.4	$WA(h = 1000)$, CPU(s)	155.9	1594.4		
Memory (Mb)	34.5	78.5	287.1	577.01	Memory (Mb)	0.5	1.0		
Nfun	1829	2189	2719	2714	Nfun	-	-		
Iterations	907	1077	1347	1364	Iterations	48	63	-	-
Obj	-1.0000	-1.0000	-1.0000	-1.0000	Obj	0.0000	-1.000		
$\Vert h(x) \Vert$	$3.8 \cdot 10^{-12}$	$2.7 \cdot 10^{-12}$	$5.1 \cdot 10^{-12}$	$2.7 \cdot 10^{-13}$	$\Vert h(x) \Vert$	$3.1 \cdot 10^{-7}$	$4.6 \cdot 10^{-7}$		
Merit Function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	Merit Function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$		

Figure 4 shows the value of the constraint as integration proceeds for increasing values of α

(panels A to C) and the values of the objective function (panel D). As it can be seen, constraint satisfaction when the maximum number of integration steps was limit to 2000 units depends on the value of α, at least for formulations $PP(\beta = 0)$ and $PP(\beta = 1)$. For example, Panel A shows that formulation $PP(\beta = 0)$ achieves a constraint satisfaction in the order of 10^{-2} at the end of integration, requiring nearly 2000 integration steeps. However, this does not mean that formulation $PP(\beta = 0)$ is unable to produce sharp constraint satisfaction. In fact, as shown in Table 7, a constraint satisfaction of $9.3 \cdot 10^{-8}$ can be attained using a longer integration time (controlled by demanding a value of the merit function smaller than a certain threshold, 10^{-6} for this problem)

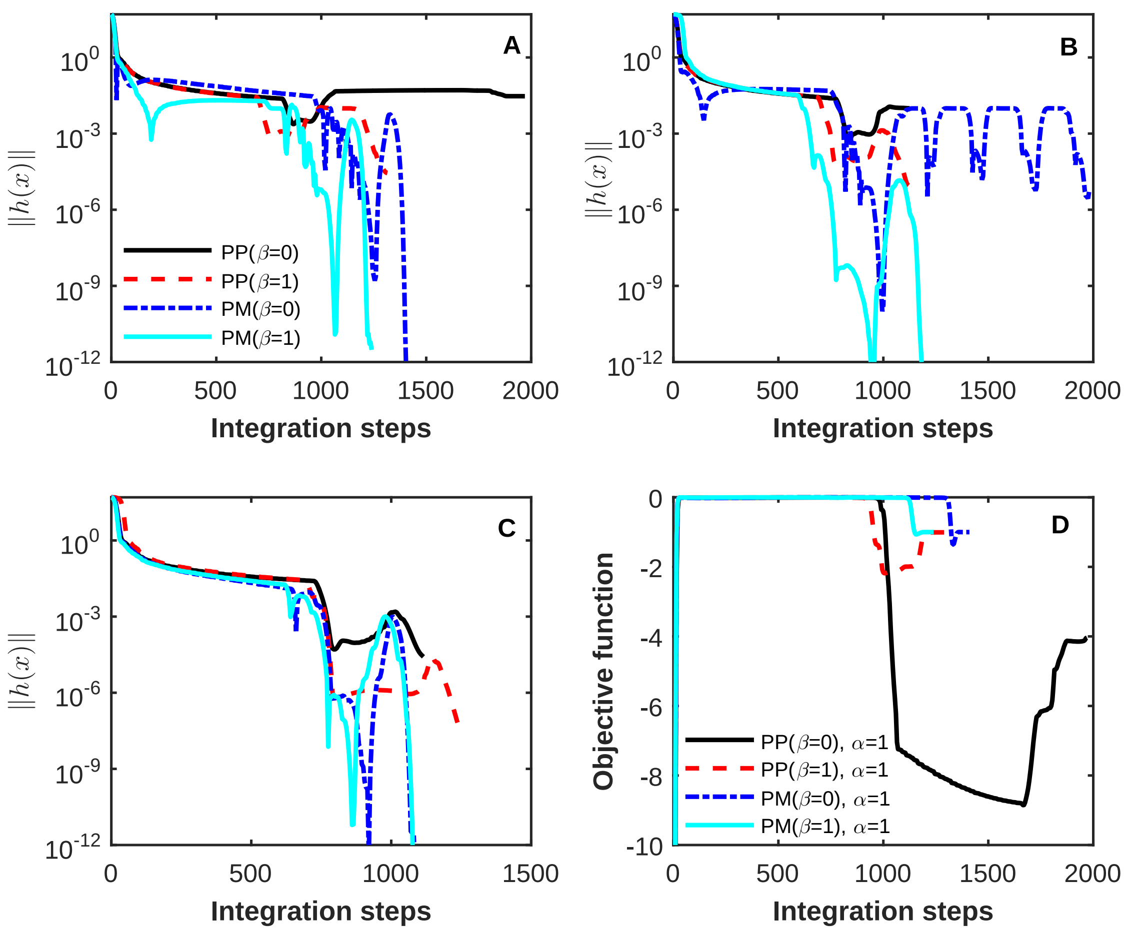

Figure 4: Constraint satisfaction for Problem 5 (with $n_v = 100$) for $\alpha = 1$(panel A), $\alpha = 10$ (panel B) and $\alpha = 1000$ (panel C). Panel D shows the value of the objective function for the Gradient Flow formulations with $\alpha = 1$. Integration time was set to 500 units.

As shown in Figure 4, once a feasible point is achieved (for example $\|h(x)\| = 10^{-12}$ in panel C for formulation PM) the trajectory of the ODE system does not remain feasible, unlike the Gradient

Flow approach presented by Wang et al. [23] for the solution of NLP problems (see Section 2.3).

3.2. Optimal control problems as NLP problems

3.2.1. Problem 6

Problem 6 corresponds to the determination of the optimal acceleration along time such that the total travel time is minimized for a car, subject to a path constraint (speed should be less than 10 units), final point constraint (distance should be equal to 300 units ($y_2 = 300$)), final velocity should be zero ($y_1 = 0$) and bounds for acceleration. The optimal control problem is defined by

$$
\begin{aligned}
& \min_{t_f,u} t_f && (86)\\
\text{subject to:} & && \\
& y_1'(t) = u(t) && (87)\\
& y_2'(t) = y_1(t) && (88)\\
& y_1(t) \leq 10 && (89)\\
& y_1(t_f) = 0 && (90)\\
& y_2(t_f) = 300 && (91)\\
& -2 \leq u(t) \leq 1 && \\
& 0 \leq t_f \leq 50 && (92)
\end{aligned}
$$

Using the Euler backward difference formula, the optimal control problem can be written as a finite dimensional NLP problem:

$$
\begin{aligned}
& \min_{t_f,u} t_f && (93)\\
\text{subject to:} & && \\
& y_{1,i} - y_{1,i-1} - \left(\frac{t_f}{n_h}\right) u_i = 0 && (94)\\
& y_{2,i} - y_{2,i-1} - \left(\frac{t_f}{n_h}\right) y_{1,i} = 0 && (95)\\
& y_{1,0} = 0 && (96)\\
& y_{2.0} = 0 && \\
& y_{1,n_h} = 0 && (97)\\
& y_{2,n_h} = 300 && (98)\\
& 0 \leq y_i \leq 10;\ i = \{0, 1, ..., n_h\} && (99)\\
& -2 \leq u_i \leq 1;\ i = \{0, 1, ..., n_h\} && (100)
\end{aligned}
$$

where n_h is the number of integration elements. For PP it was necessary to reduce the α value to 10, while the value of this parameter for formulation PP was maintained in 10^3.

Results are presented in Table 8 showing that every conventional NLP solver achieved an optimal solution for 5 and 20 intervals. Formulation *PP ($\beta = 1$)* was unable to achieve the optimal solution as the integration terminated when the penalty parameter value exceeded 10^{12}. On the other hand, formulation PM achieves feasibility and locally optimal solutions for 5 and 20 intervals in a fraction of the time required by the PP formulation. Still, the time consumed by the PM formulation to attain the solution of the problem is larger compared to the one required by CONOPT or other commercial NLP solvers. Using the algorithm presented by Wang et al. [23], WA with $h = 10^{-5}$and $\tau = 1000$ for $n_h = 5$, no solution was attained after 1000 iterations and 7492 CPU seconds, reaching a merit function value equal to $7.9 \cdot 10^5$ with very slow progress towards constraints satisfaction.

Table 8: Solution summary for Problem 6. The set of ODEs generated by the schemes $PM(\beta = 1)$ and $PM(\beta = 0)$ were solved in MATLAB.

	$n_h = 5$	$n_h = 20$		$n_h = 5$	$n_h = 20$
IPOPT, CPU (s)	0.8	1.1	CONOPT	0.7	0.7
Memory (Mb)	22.3	29.2	Memory (Mb)	0.6	2.1
Iters	14	16	Iters	20	63
Obj	39.56	37.62	Obj	39.64	37.62
$\|h(x)\|$	$1.0 \cdot 10^{-11}$	$3.1 \cdot 10^{-11}$	$\|h(x)\|$	0	0.00
MINOS, CPU (s)	0.9	0.46	SNOPT	0.5	0.7
Memory (Mb)	2.0	2.1	Memory (Mb)	0.7	0.9
Iters	51	88	Iters	23	109
Obj	39.56	37.62	Obj	36.56	37.62
$\|h(x)\|$	$2.8 \cdot 10^{-14}$	$3.2 \cdot 10^{-7}$	$\|h(x)\|$	$1.6 \cdot 10^{-7}$	$9.1 \cdot 10^{-8}$
$PP(\beta = 0)$, CPU (s)	0.77	272.6	$PP(\beta = 1)$	2.06	310.1
Memory (Mb)	72.3	344.4	Memory (Mb)	10.4	425.4
Nfun	17648	17205	Nfun	14572	26905
Integration steps	8935	10195	Integration steps	7340	13551
Obj	49.97		Obj	50	50.1
$\|h(x)\|$	$2.1 \cdot 10^{-6}$	$3.5 \cdot 10^{-7}$	$\|h(x)\|$	$2.4 \cdot 10^{-8}$	0.60
Merit function	$1 \cdot 10^{-6}$	$9.93 \cdot 10^{-7}$	Merit Function	$8.9 \cdot 10^{-7}$	168.1*
$PM(\beta = 0)$, CPU (s)	0.64	12.1	$PM(\beta = 1, \alpha = 1)$	1.6	15.1
Memory (Mb)	28.0	39.2	Memory (Mb)	76.4	13.0
Nfun	2897	16760	Nfun	8527	20683
Integration steps	1662	7970	Integration steps	4637	10431
Obj	50	37.62	Obj	50	37.62
$\|h(x)\|$	$2.1 \cdot 10^{-6}$	$9.3 \cdot 10^{-6}$	$\|h(x)\|$	$3.4 \cdot 10^{-13}$	$3.7 \cdot 10^{-6}$
Merit function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-5}$	Merit function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-5}$

* Integration stops when $M > 10^{12}$.

In Table 8 the system of differential equations $PM(\beta = 1)$ reaches the demanded value for the merit function after 10431 integration steps reporting a local optimum. However, if the integration is allowed to continue and smaller values of α are used, a feasible point is achieved after 2500 integration steps, and after several integration steps where the value of the objective function remains fixed, it starts decreasing towards the optimal value. When the norm of the constraint vector is plotted against the number of integration steps for $PM(\beta = 1)$ and $n_h = 20$, a behavior similar to Problem 5 is observed (compare figures 4 and 5). Interestingly, smaller α values require fewer integration steps to move from the plateau value of 50 units of time to the globally optimum value reported by the commercial solvers. This can be explained by the stiffness caused by large α

503 values.

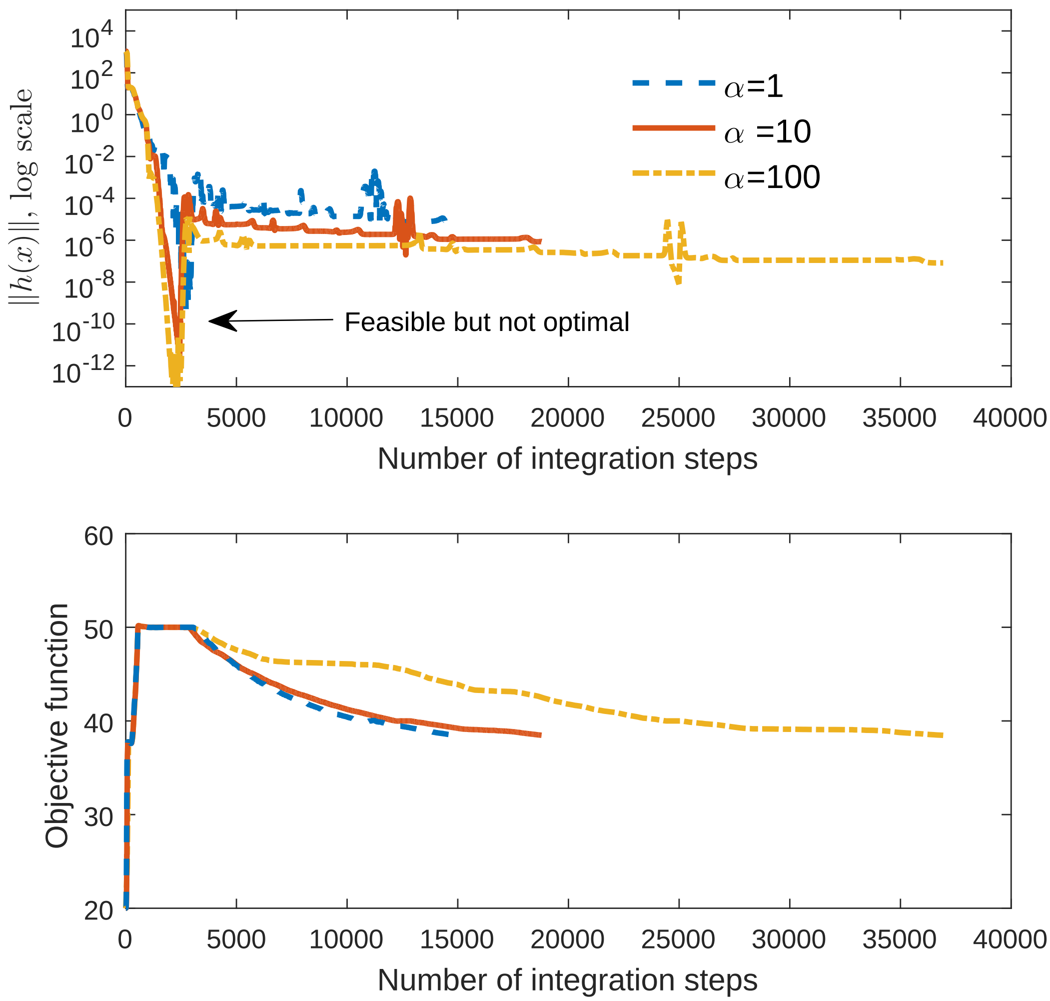

Figure 5: Norm of the constraints vector in Problem 6 (with $n_h = 20$) as the integration proceeds (as number of integration steps) for increasing values of α. Integration time was set to 500 units.

504 *3.2.2. Problem 7*

Problem 7 corresponds to the maximization of the harvested amount of a biological resource, provided this resource grows as time passes. The growth rate is assumed to be proportional to the amount of the resource present at a given time. The optimal control problem is defined by

$$\min_u - \int_0^{t_f} \sqrt{u(t)}dt \tag{101}$$

subject to:

$$x'(t) = (\varphi - 1)x(t) - u(t) \tag{102}$$

$$x(0) = x(t_f) \tag{103}$$

$$x(t) \geq 0 \tag{104}$$

$$u(t) \geq 0 \tag{105}$$

$$0 \leq t \leq t_f \tag{106}$$

$$x(0) = x_0 \tag{107}$$

where $x(t)$ is a scalar function representing the amount of a biological resource, $(\varphi - 1) > 0$ is the growth rate and $u(t)$ is the amount of the resource extracted at a given time. Moreover, the problem demands that the amount of resource at the final time be equal to its initial amount. By using a discretization in time of 1 unit, the problem can be rewritten as the following finite-dimensional NLP problem [33]:

$$\min_{u_k} - \sum_{k=0}^{N-1} \sqrt{u_k} \tag{108}$$

subject to:

$$x_{k+1} = \varphi x_k - u_k \tag{109}$$

$$x_0 = x_N \tag{110}$$

$$0 \leq u_k \leq u_k^U;\ k = \{0, 1, ..., N\} \tag{111}$$

$$0 \leq x_k \leq x_k^U;\ k = \{0, 1, ..., N\} \tag{112}$$

The optimal value of the NLP problem represented by equations 108 to 112 is known analytically to be [33] given by

$$J^* = \sqrt{\frac{x_{ini}(\varphi^N - 1)^2}{\varphi^{N-1}(\varphi - 1)}} \tag{113}$$

The optimal control problem was solved for N with values of 10, 20, 30 and 50, $x_{ini} = 1$ and $\varphi = 1.1$. For $PP(\beta = 0)$ and $PM(\beta = 0)$, α was set as 10^4 while for $PM(\beta = 0)$ and $PM(\beta = 1)$, α was set as 10^2. The systems of ODEs were integrated until the merit function value was less than 10^{-6}. The upper bounds for the amount of resource (x_k^U) and control variable (u_k^U) need to be increased as N increases. Hence, for values of $N = \{10, 30, 50\}$, $x_k^U = \{2, 5, 30\}$ and $u_k^U = \{1, 2, 10\}$for every k.

Table 9 shows the computational results for this case study. Solvers MINOS and SNOPT report infeasible solutions for every value of N, despite the problems having a moderate number of variables $(2 \cdot N)$. CONOPT and IPOPT achieve the optimal solution values, -3.282 for $N = 10$, -13.060 for $N = 30$ and -35.629 for $N = 50$. GF formulations achieve between 98.2% to 100% of the optimal solution with excellent constraint satisfaction, especially for $PM(\beta = 0)$ and $PM(\beta = 1)$ formulations.

Table 9: Solution summary for Problem 7.

	$N = 10$	$N = 30$		$N = 10$	$N = 30$
IPOPT, CPU(s)	0.32	0.30	CONOPT	0.23	0.25
Memory (Mb)	30.8	31.0	Memory (Mb)	1.8	1.9
Nfun	54	74	Iters	NA[a]	NA
Obj	-3.282	-13.060	Obj	-3.282	-13.060
$\|h(x)\|$	$1.0 \cdot 10^{-11}$	$1.0 \cdot 10^{-11}$	$\|h(x)\|$	$2.2 \cdot 10^{-16}$	$3.6 \cdot 10^{-8}$
$PP(\beta = 0)$, CPU(s)	0.23	0.81	$PP(\beta = 1)$	0.14	0.66
Memory (Mb)	9.0	28.7	Memory (Mb)	8.2	34.9
Nfun	1680	2398	Nfun	1801	3060
Integration steps	853	1075	Integration steps	902	1360
Obj	-3.281	-13.061	Obj	-3.282	-13.061
$\|h(x)\|$	$9.4 \cdot 10^{-7}$	$4.8 \cdot 10^{-7}$	$\|h(x)\|$	$8.4 \cdot 10^{-7}$	$3.4 \cdot 10^{-7}$
Merit function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	Merit function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$
$PM(\beta = 0)$, CPU(s)	0.078	0.55	$PM(\beta = 1)$	0.12	0.68
Memory (Mb)	7.3	45.2	Memory (Mb)	12.9	58.3
Nfun	678	2083	Nfun	1293	2701
Integration steps	363	956	Integration steps	684	1207
Obj	-3.282	-13.060	Obj	-3.282	-13.060
$\|h(x)\|$	$6.2 \cdot 10^{-9}$	$1.8 \cdot 10^{-9}$	$\|h(x)\|$	$8.3 \cdot 10^{-10}$	$1.7 \cdot 10^{-10}$
Merit function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$	Merit function	$1.0 \cdot 10^{-6}$	$1.0 \cdot 10^{-6}$

[a]Not reported by GAMS CONOPT. [b]Solver reports an infeasible solution

An attempt to use the algorithm presented by Wang et al. [23], WA, to solve this problem with $N = 10$ was undertaken. After a systematic search of τ and integration step values (h), the values producing the best solution were $\tau = 1000$ and $h = 0.001$. The algorithm achieves a merit function value of 0.66, objective function value of -5.2 and norm of the constraint vector equal to $1.2 \cdot 10^{-4}$ in 1797 CPU seconds and 3000 iterations.

Figure 6 shows the progress of merit function and constraint satisfaction as well as the obtained solution for the amount of resource and the control variable. Panels C and D show the trajectory of the state and control variables. Trajectories produced using formulation $PM(\beta = 1)$ are slightly different compared to those obtained using CONOPT or IPOPT, which explains the near optimal objective function value calculated for $PM(\beta = 1)$ and $N = 30$ in Table 9.

Summarizing our results, the case studies show that the GF approach presented in this work is faster, in terms of CPU time, compared to a previously reported GF formulation [23], which is representative of a family of formulations used to transform NLP problems to a system of ODEs, and requires fewer function evaluations and integration steps (WM implementation versus PP and PM schemes). This family of formulations makes extensive use of the inverse matrix $[J(x)J^T(x)]^{-1}$, introducing a computational overhead that it is eliminated in our formulations thanks to a simpler approach to penalizing functions. Moreover, in the GF formulation reported by Wang et al. [23] a penalty parameter τ has to be tuned for each problem, and for the algorithmic version of the GF scheme reported bythe authorsnot only τ has to be tuned for each problem but also the integration step of the backward Euler's scheme (h). Results obtained for Problem 2 show that in WM method,

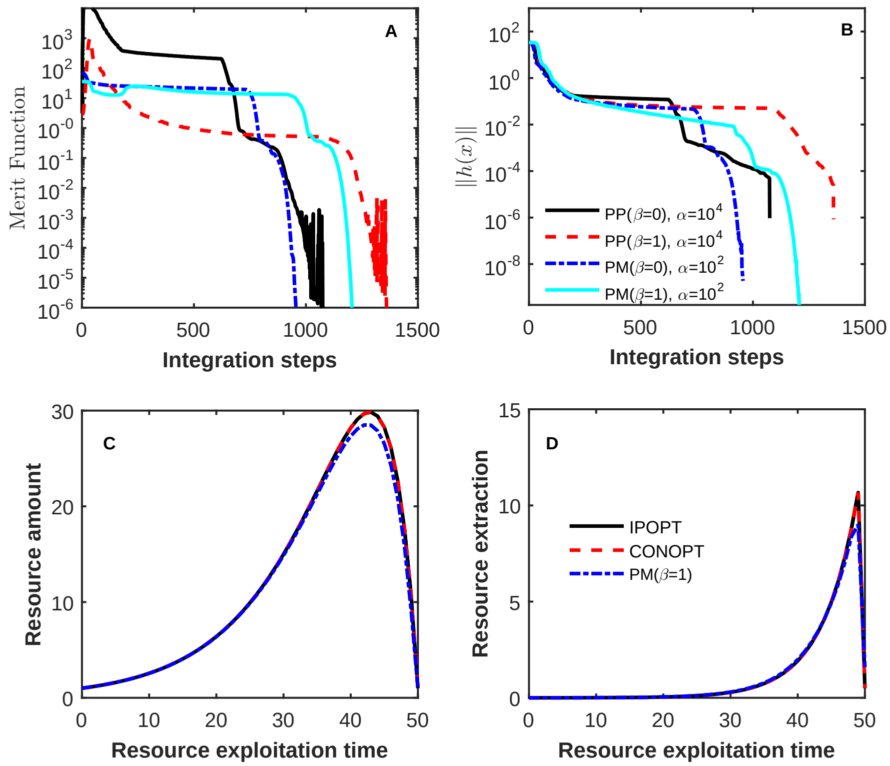

Figure 6: Progress towards the solution as integration proceeds for Problem 7.Merit function for Problem 7 (with $N = 30$, panel A) , norm of the constraints vector value (panel B), Panels C and D show the trajectory of the state variables ($x(t)$, representing the amount of resource) and control variable ($u(t)$, amount of resource extracted at a given time) for $N = 30$.

selection of τ plays a critical role, while low values produce slow converge, too high values produce a very stiff system of ODEs. For problem 4, 5, 6 and 7 WA scheme fails to reach a solution for the problem. For these problems, the reported solutions correspond to the best solution achieved after testing several combination of h and τ values, being the best ones obtained for low values of h and high values of τ, however, the low h values produce a slow convergence, a result that can be expected from the analysis presented by Wang et al. [23], who claims a linear rate of convergence for small h values. The GF schemes presented in this work successfully avoid this issues by using a self-tuning approach of the penalty parameters.

In our approach, the solution of the problem is approached from the exterior of the feasible set at every step, except when the optimal solution is attained. This in turns allow the use of conventional slacks to bound variables. Compared to the state-of-the-art NLP solvers used in this work as benchmark, the proposed GF formulation remains competitive for problems with less than one hundred variables, being especially useful for problems with highly non-linear constraints, either as a solution or an initialization method.

4. Conclusions

This work presents a novel Gradient Flow formulation for the solution of nonlinear optimization problems with equality and inequality constraints. The proposed schemes were shown theoretically to converge (asymptotically) to a local minimum of the original problem under conventional assumptions on the objective function and constraints. These formulations were theoretically and numerically compared to other reported Gradient Flow formulations for the solution of nonlinear constrained optimization problems, showing that the proposed formulations outperforms the reference method in terms of computational time and the size of problems that can be solved. Moreover, the self-tuning nature of the proposed approach reduces the numerical problems introduced by increasing the value of the penalty parameter.

Numerical experiments using a set of seven specially selected problems, ranging from 3 to 600 variables, show that the proposed schemes are robust and converge to feasible points and local minima, irrespective of the choice of the value of parameter α used in the formulations, due to the self-tuning properties of the penalty parameters introduced in this work.

Moreover, results suggest that, for the set of problems analyzed, the GF formulations were able to find the optimal solution to problems where conventional NLP solvers fail. Primarily this is due to the fact that constraints are not linearized at intermediate iterations, with the solution being approached from the exterior of the feasible set.

As shown by the computational experiments, the GF formulations presented in this work achieve feasibility for problems with difficult nonlinear constraints. The combined multiplier and penalty approach in formulation PM provides solutions in shorter times and with sharp constraint satisfaction for almost every problem compared to the other GF formulations presented and compared in this work.

Most likely, if a customized integrator were used to solve the ODE systems produced by this and the other GF formulations, solution times and the number of function evaluations and iterations will be significantly reduced.

Future work includes an algorithmic implementation of the formulations presented in this work using a customized integrator and the exploration of the possibility of exploiting the special structure of the linear constraints and bounds as to reduce the ODE system size, for example incorporating the ideas presented in Schropp and Singer [24] and Shikhman and Stein [34], that might allow the decoupling of the original variables from the variables used to enforce bounds, thereby reducing the size of the dynamical system.

References

[1] I. E. Grossmann, Advances in mathematical programming models for enterprise-wide optimization, Computers & Chemical Engineering 47 (2012) 2–18.

[2] R. Amrit, J. B. Rawlings, L. T. Biegler, Optimizing process economics online using model predictive control, Computers & Chemical Engineering 58 (2013) 334–343.

[3] D. Yue, M. A. Kim, F. You, Design of Sustainable Product Systems and Supply Chains with Life Cycle Optimization Based on Functional Unit: General Modeling Framework, Mixed-Integer Nonlinear Programming Algorithms and Case Study on Hydrocarbon Biofuels BT - ACS Sustainable Chemistr, Sustainable Chemistry & Engineering (2013) 1–15.

[4] R. C. Baliban, J. A. Elia, C. A. Floudas, Biomass to liquid transportation fuels (BTL) systems: process synthesis and global optimization framework, Energy Environ. Sci. 6 (1) (2013) 267–287.

[5] F. Scott, F. Venturini, G. Aroca, R. Conejeros, Selection of process alternatives for lignocellulosic bioethanol production using a MILP approach, Bioresource Technology 148 (2013) 525–534.

[6] R. H. Byrd, M. E. Hribar, J. Nocedal, An Interior Point Algorithm for Large-Scale Nonlinear Programming, SIAM Journal on Optimization 9 (4) (1999) 877–900.

[7] R. J. Vanderbei, D. F. Shanno, An Interior-Point Algorithm for Nonconvex Nonlinear Programming, Computational Optimization and Applications 13 (1/3) (1999) 231–252.

[8] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical Programming 106 (1) (2006) 25–57.

[9] P. E. Gill, W. Murray, M. A. Saunders, SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization, SIAM Review 47 (1) (2005) 99–131.

[10] A. S. Drud, CONOPT–A Large-Scale GRG Code, INFORMS Journal on Computing 6 (2) (1994) 207–216.

[11] M. R. Bussieck, A. Meeraus, General Algebraic Modeling System (GAMS), in: Modeling Languages in Mathematical Optimization, J. Kallrath (Ed.), vol. 88 of *Applied Optimization*, Springer US, 137–157, 2004.

[12] R. Fourer, D. Gay, B. W. Kernighan, The AMPL book, Duxbury Press, Pacific Grove, 2002.

[13] J. Bisschop, M. Roelofs, AIMMS - Language Reference, Lulu.com, ISBN 1411698975, 2006.

[14] A. A. Brown, M. C. Bartholomew-Biggs, Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations, Journal of Optimization Theory and Applications 62 (2) (1989) 211–224.

[15] C. Botsaris, Differential gradient methods, Journal of Mathematical Analysis and Applications 63 (1) (1978) 177–198.

[16] A. A. Brown, M. C. Bartholomew-Biggs, ODE versus SQP methods for constrained optimization, Journal of Optimization Theory and Applications 62 (3) (1989) 371–386.

[17] Y. G. Evtushenko, V. G. Zhadan, Stable barrier-projection and barrier-Newton methods in linear programming, Computational Optimization and Applications 3 (4) (1994) 289–303.

[18] G. V. Smirnov, Convergence of Barrier-projection methods of optimization via vector Lyapunov functions, Optimization Methods and Software 3 (1-3) (1994) 153–162.

[19] R. J. Orsi, A Dynamical Systems Analysis of Semidefinite Programming with Application to Quadratic Optimization with Pure Quadratic Equality Constraints, Applied Mathematics and Optimization 40 (2) (1999) 191–210.

[20] K. Tanabe, An algorithm for constrained maximization in nonlinear programming, J. Oper. Res. Soc. Jpn 17 (1974) 184–201.

[21] J. Rosen, The gradient projection method for nonlinear programming. Part II. Nonlinear constraints, Journal of the Society for Industrial and Applied Mathematics 9 (4) (1961) 514–532.

[22] H. Yamashita, A differential equation approach to nonlinear programming, Mathematical Programming 18 (1) (1980) 155–168.

[23] S. Wang, X. Yang, K. Teo, A Unified Gradient Flow Approach to Constrained Nonlinear Optimization Problems, Computational Optimization and Applications 25 (1/3) (2003) 251–268.

[24] J. Schropp, I. Singer, A dynamical systems approach to constrained minimization, Numerical functional analysis and optimization 21 (3-4) (2000) 537–551.

[25] L. T. Biegler, Nonlinear Programming, Society for Industrial and Applied Mathematics, 2010.

[26] M. R. Hestenes, Multiplier and gradient methods, Journal of Optimization Theory and Applications 4 (5) (1969) 303–320.

[27] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, Nonlinear Programming, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[28] J. Nocedal, S. Wright, Numerical optimization, Springer Series in Operations Research, Springer, second edition edn., 1999.

[29] F. Verhulst, Nonlinear differential equations and dynamical systems, January 1996, 1990.

[30] A. Navarro, V. Vassiliadis, Computer algebra systems coming of age: Dynamic simulation and optimization of DAE systems in Mathematica, Computers & Chemical Engineering 62 (2014) 125–138.

[31] T. J. Williams, R. E. Otto, A generalized chemical processing model for the investigation of computer control, Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 79 (5) (1960) 458–473.

[32] T. J. Berna, M. H. Locke, A. W. Westerberg, A new approach to optimization of chemical processes, AIChE Journal 26 (1) (1980) 37–43.

[33] Z. Michalewicz, C. Z. Janikow, J. B. Krawczyk, A modified genetic algorithm for optimal control problems, Computers & Mathematics with Applications 23 (12) (1992) 83–94.

[34] V. Shikhman, O. Stein, Constrained optimization: projected gradient flows, Journal of optimization theory and applications 140 (1) (2009) 117–130.