Differentiation Tool Efficiency Comparison for Nonlinear Model Predictive Control Applied to Oil Gathering Systems.

Andrés Codas [*] **Marco Aurélio S. Aguiar** [*,**] **Konstantin Nalum** [*] **Bjarne Foss** [*]

[*] *Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway*
[**] *Department of Automation and Systems Engineering, Federal University of Santa Catarina, Florianópolis, SC 88040-900 Brazil*

Abstract: This paper presents a comparison of gradient computation techniques required to solve a single-shooting formulation of nonlinear model predictive control (NMPC) problems. An oil production system with network structure is considered as test instance. The structure of the network is exploited to improve computational efficiency. Exact gradient sensitivity calculation methods (forward and adjoint) are compared along with the finite difference approximation. Forward and Reverse automatic differentiation for calculating Jacobians are also compared along with the finite difference approximation counterpart. Since there is a trade off involving accuracy and speed when calculating these gradients, the best combination of tools is case dependent and it is determined by the analyses of performance indexes arising when solving specific NMPC problems. A hybrid approach combining finite difference Jacobian calculations with adjoint sensitivity calculations gave the best performance for our test problems.

Keywords: Gas-lift, Gradient Computation, Automatic Differentiation

1. INTRODUCTION

The success of most nonlinear optimization problem solvers, and in particular the efficiency of optimal control algorithms relies on the performance of gradient computation. These problems are solved by iterative algorithms which solve a linearization of the nonlinear problem at each iteration. This linearization consists of Taylor approximations of the optimization problem objective and constraints. A trade-off arises because poor approximations may incur more iterations or divergence while accurate gradients can be computational expensive leading to a prohibitive solution time for real-time purposes.

Gradient calculation algorithms performance depend on the problem instance. In this paper we assess gradient computation techniques for a Nonlinear Model Predictive Control (NMPC) with a single-shooting formulation applied to short-term oil production optimization. The NMPC application typically communicates with a low level control layer which maintains stability (for open loop unstable systems) and ensures high frequency disturbance rejection. Further, the application may receive information from an upper level optimizer which maximizes a long term economical index.

The oil gathering system considered in this work possesses a network structure consisting of a set of wells, manifolds, pipeline-riser and separators. The wells and pipeline-riser systems are modeled by Ordinary Differential Equations (ODE) based on previous work by Eikrem et al. (2008) and Jahanshahi and Skogestad (2011), respectively. The manifold is modeled by a pressure-flow balance with no states, thus it introduces algebraic equations, while the separator is represented by a fixed pressure boundary condition. The coupled system model is an index-1 Differential Algebraic Equation (DAE), which can be easily transformed to ODEs.

The NMPC problem is solved in a sequential approach where the Nonlinear Programing (NLP) solver relies on information provided by the ODE solver. Gradient information can be provided by solving additional ODEs through the implementation of the direct or adjoint sensitivity method or by repeated simulations through the finite difference method (Biegler, 2010).

Implicit ODE solvers and sensitivity calculation methods require the calculation of partial derivatives (or Jacobians) of the system. Automatic differentiation tools, namely the forward and reverse methods, are used to obtain exact (up to machine precision) partial derivatives (Griewank and Walther, 2008). Finite difference approximations are also popular because of their simplicity. In this work we propose to exploit the network structure of the system through the application of the chain rule in order to improve computational efficiency.

Tools for exact differentiation and approximate differentiation are compared measuring their performance when solving one iteration of the NMPC problem. To this end, IPOPT (Wächter and Biegler, 2006) is used as the NLP solver.

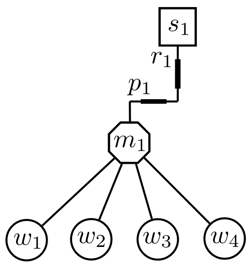

Fig. 1. Oil gathering network structure with four wells, one manifold, pipeline-riser and separator.

In section 2 the oil gathering network model is described. The system equations are not written explicitly, however, the system structure is described and decomposition opportunities are discussed. In section 3 the oil production optimization problem is formulated as a single-shooting NMPC problem. The gradient calculation methods are presented in section 4 and some of their properties are discussed. Section 5 illustrates the performance of the presented tools when tackling the proposed problem. In section 6 the results obtained are summarized and future research directions are pointed. In section 7 the contribution of this paper is briefly stated.

2. SYSTEM MODELING

Figure 1 illustrates the oil gathering network structure treated in this work. $w_i, i \in \{1, \ldots, 4\}$ represent wells, while m_1, p_1, r_1 and s_1 represent a manifold, a horizontal pipeline segment, a riser, and a separator, respectively.

2.1 Well Model

Wells operated by gas-lift are considered. A model with three states was developed inspired by Eikrem et al. (2008). The states correspond to the mass of gas in the well annulus, and the mass of gas and the mass of liquid in the well production tubing.

The inflow performance relationship (IPR), i.e., the model of the flow from the oil reservoir into the well, is modeled by a quadratic function of the bottom-hole pressure. The black-oil fluid model is used to represent the reservoir fluid which is assumed to have constant water-cut (WCUT) and gas-oil ratio (GOR). The pressures in the tubing and the annular are a consequence of the hydrostatic pressures and the ideal gas law. The well is assumed to be connected to one manifold through a choke valve. Since the control inputs in the later experiments are gas injection rates the choke valve is considered always fully open and its pressure drop is assumed to be proportional to the square of the mass flow.

2.2 Manifold-Pipeline-Riser-Separator Model

The manifold is modeled by an algebraic equation, no dynamic variables are used. The manifold pressure is assumed to be the same as the boundary pressure of the pipeline-riser model.

The horizontal pipeline and the riser dynamic models are based on (Jahanshahi and Skogestad, 2011) which considers two phases (gas and liquid). Since the liquid input has variable water-cut the liquid phase is separated in an oil phase and a water phase. Within the horizontal pipeline the flow is considered stratified while in the riser the flow is considered mixed. Three states model the fluid mass for each phase (gas, oil and water) in both the horizontal pipeline and riser. The pressure gradient along both segments is modeled as a consequence of the gravitational gradient and a frictional gradient. Jahanshahi and Skogestad (2011) consider an inclined pipeline to induce slug flow effects. Since we are not interested in this behavior we assume a perpendicular connection between the horizontal pipeline and the riser.

Orifice equations model the transfer of fluids from the horizontal pipeline to the riser. At the riser outlet, the outflow is modeled by valve equations. The virtual orifice connection area between pipeline and riser depends on the volume of flowing liquids which are considered incompressible. The flow of oil and water through the orifice is proportional to their mass fractions in the pipeline.

Since the liquid properties vary as a consequence of the water-cut variation, the equations involving density, viscosity and superficial velocity had to be modified compared to the approach by Jahanshahi and Skogestad (2011). A friction correlation for fluid flow in the pipeline is calculated by using the Haaland equation (Johnson, 1998).

2.3 System Coupling

Let $\mathcal{W}$ be the set of wells and $\mathcal{P}$ the set of horizontal pipelines-risers pairs. Let $\mathcal{R} \subseteq \{(w,p) \in \mathcal{W} \times \mathcal{P} | (w,p) \in \mathcal{R}, (w,p') \in \mathcal{R}, p = p'\}$ be the set establishing the routing configuration, then $R(p) = \{w \in \mathcal{W} | (w,p) \in \mathcal{R}\}$ and $R(w) = \{p \in \mathcal{P} | (w,p) \in \mathcal{R}\}$ which is a singleton since a well can only be connected to one pipeline.

The dynamic systems described in sections 2.1 and 2.2 can be represented with the following system of differential-algebraic equations (DAE):

$$\dot{\mathbf{x}}_w = f_w\left(\mathbf{x}_w, u_w, \sum_{p \in R(w)} \mathbf{y}_{w,p}\right), \forall w \in \mathcal{W} \tag{1a}$$

$$\dot{\mathbf{x}}_p = f_p\left(\mathbf{x}_p, \mathbf{y}_p\right), \forall p \in \mathcal{P} \tag{1b}$$

$$\mathbf{y}_{w,p} = g_{w,p}\left(\mathbf{x}_w, \mathbf{x}_p\right), \forall (w,p) \in \mathcal{R} \tag{1c}$$

$$\mathbf{y}_p = \sum_{w \in R(p)} \mathbf{y}_{w,p}, \forall p \in \mathcal{P} \tag{1d}$$

where:

- $\mathbf{x}_w$ and $\mathbf{x}_p$ are the states relative to well w (3 states) and the states relative to pipeline-riser p (6 states), respectively.
- u_w represents the gas injection rate for the well w.
- $\mathbf{y}_{w,p}$ represents the flow rate from well w to the pipeline p.
- f_w and f_p are the dynamical equations for the well w and the pipeline p, while $g_{w,p}$ is the function establishing the pressure-flow boundary between the well and pipeline.

$\mathbf{y}_{w,p}$ and $\mathbf{y}_p$ can be replaced by means of (1c) and (1d), turning the DAE into ODEs, however, these equations are left explicit in order to take advantage of the system structure. When running simulations of the system the ODE representation is used.

The system equations f_w, f_p and $g_{w,p}$ are continuous, piecewise-differentiable and explicit. The finite difference gradient approximation is affected by the termination conditions of iterative algorithms to solve implicit functions, thus explicit equations are more convenient for this purpose. Piecewise-differentiable functions are required to be able to define derivatives in all points, if in a particular point the function is not differentiable then the derivative from one side is taken.

3. PROBLEM FORMULATION

Let $\psi(\mathbf{x}(t_f))$ be a function of the state variables $\mathbf{x} = \{\mathbf{x}_{w_1}, \ldots, \mathbf{x}_{w_{n_w}}, \mathbf{x}_{p_1}, \ldots, \mathbf{x}_{p_{n_p}}\}$ at the prediction horizon final time t_f. The DAE system (1) can be written as ODEs with $\dot{\mathbf{x}} = f_c(\mathbf{x}, \mathbf{u}_c(t))$, where $\mathbf{u}_c(t) = (u_{w_1}, \ldots, u_{w_{n_w}})$. We model the control profiles $\mathbf{u}_c(t)$ as piecewise constant functions, thus, given a time discretization set $\mathcal{T} = \{t_0, \ldots, t_{n_t} = t_f\}$, $\mathbf{u}_c(t) = \mathbf{u}^k$ if $t_{k-1} \le t < t_k$. Finally an optimal control problem can be stated as an optimization problem with embedded ODEs:

$$\begin{aligned} \min \quad & \psi(\mathbf{x}(t_f)) && (2a)\\ s.t. \quad & \dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u}) && (2b)\\ & \mathbf{x}(0) = \mathbf{x}_0 && (2c)\\ & c_I(\mathbf{x}(t_I), \mathbf{u}) \le \mathbf{0} && (2d)\\ & \mathbf{u}_{\min} \le \mathbf{u} \le \mathbf{u}_{\max} && (2e) \end{aligned}$$

where:

- f is analog to f_c but taking the discrete control parameters. The time t is also added in order to resolve which of the control parameters $\mathbf{u} = \{\mathbf{u}^1, \ldots, \mathbf{u}^{n_t}\}$ is active.
- $c_I(\mathbf{x}(t_I), \mathbf{u})$ are nonlinear functions describing constraints, where t_I represents any time in the range $(0, t_f]$ representing the prediction horizon.
- If the objective or a constraint needs to be defined as an integral function of the state variables, for instance $\psi(\mathbf{x}(t_f)) = -\int_0^{t_f} q^{\mathrm{o}}(\mathbf{x})dt$, where q^{o} is the oil outflow, then an additional state x_ψ can be added to the system.

Assuming that the initial value problem given by (2b) and (2c) has a unique solution, problem (2) can be written (with some notation abuse) as a single shooting problem depeding on the variables $\mathbf{u}$:

$$\begin{aligned} \min \quad & \psi(\mathbf{u}) && (3a)\\ s.t. \quad & c_I(\mathbf{u}) \le \mathbf{0} && (3b)\\ & \mathbf{u}_{\min} \le \mathbf{u} \le \mathbf{u}_{\max} && (3c) \end{aligned}$$

4. GRADIENT CALCULATIONS

Single shooting problems like (3) which do not posses known convexity, are typically tackled with Sequential Quadratic Programming (SPQ) or Interior Point (IP) methods (Diehl et al., 2002; Biegler et al., 2002), given the assumption that they are locally convex close to a local optimal solution. Efficient SQP and IP algorithms require gradients ($\nabla_{\mathbf{u}}\psi$ and $\nabla_{\mathbf{u}}c_I$) and Hessian approximations. These computations are the most time-consuming part in single-shooting algorithms, see e.g. Imsland et al. (2010). Linear model predictive control algorithms are relatively easy to implement due to the simplicity of the gradients calculations, in this case the predicted states are linear w.r.t. the input and gradients are obtained from the step/impulse response of the system. In addition, state of the art linear optimization solvers are able to solve problems with hundreds of thousands of variables in few minutes, see e.g. Codas et al. (2012). On the other hand, nonlinear model predictive control algorithms present a challenge because iterative algorithms such as SQP and IP require gradients and Hessians at each iteration.

The formulation of the control problem (3) has embedded the ODE system (2b)-(2c). Therefore, the required gradients depend on how the control variables affect the ODEs. In the following we discuss the available tools for gradient computation for this equation system. Extensions for differential algebraic systems can be found in (Biegler, 2010).

4.1 Finite Difference Approximation (FD)

Given a direction vector $\mathbf{p} \in \mathbb{R}^{n_{\mathbf{u}}}$ a directional derivative approximation can be obtained by:

$$\nabla_{\mathbf{p}}\psi(\mathbf{u}) = \lim_{h\to 0} \frac{\psi(\mathbf{u} + h\mathbf{p}) - \psi(\mathbf{u})}{h} \approx \frac{\psi(\mathbf{u} + h_0\mathbf{p}) - \psi(\mathbf{u})}{h_0} \qquad (4)$$

where h_0 is a value sufficiently small to approximate the limit. It should be noticed that the solution of $\psi(\mathbf{u})$ is often an approximation controlled by the ODE solver tolerance ϵ_s. If the difference $\psi(\mathbf{u} + h_0\mathbf{p}) - \psi(\mathbf{u})$ is in the same order of magnitude or smaller than ϵ_s, then the gradient approximation can be severely affected. The complicated tuning relation between ϵ_s and h_0 besides potential computational efficiency improvements motivate the next gradient computation method.

4.2 Forward (Direct) Sensitivity Calculations (FS)

Considering the initial value problem (2b)-(2c), we focus on computing $\nabla_u\psi$ where $u \in \mathbf{u}$ is a single variable that parametrizes the control input of the NMPC problem. The forward sensitivity method principle resides on first calculating $\nabla_u\mathbf{x}(t_f)$ and then relating this gradient to ψ. Following the derivation strategy by Biegler (2010), we define $S = \frac{d\mathbf{x}}{du}$. $S(t_f)$ is then given by the solution of the following system:

$$\frac{dS}{dt} = \frac{\partial f}{\partial \mathbf{x}}S(t) + \frac{\partial f}{\partial u}^T, \quad S(0) = 0. \qquad (5a)$$

$$\nabla_u\psi = \frac{\partial \psi}{\partial \mathbf{x}}S(t_f) + \frac{\partial \psi}{\partial u} \qquad (5b)$$

The computational effort related to compute the sensitivity equations is proportional to the number of states $n_{\mathbf{x}}$ times the number of control parameters $n_{\mathbf{u}}$ (i.e., $n_{\mathbf{x}} \times n_{\mathbf{u}}$).

We solve the sensitivity equation system (5a) running the solver once for all $u \in \mathbf{u}$ along with the system (2b)-(2c) simulation to take advantage of the common computations of the partial derivatives $\frac{\partial f}{\partial \mathbf{x}}$. An alternative implementation would take advantage of parallelization capabilities by solving the sensitivities independently for each $u \in \mathbf{u}$.

4.3 Adjoint (Reverse) Sensitivity Calculations (AS)

When the number of control variables is large the Forward Sensitivity Calculation is inefficient due to the size of the ODE system (5a). This motivates the use of the Adjoint Sensitivity Calculation (Biegler, 2010) which equation system is:

$$\frac{d\lambda}{dt} = -\frac{\partial f}{\partial \mathbf{x}}^T \lambda, \quad \lambda(t_f) = \frac{\partial \psi}{\partial \mathbf{x}} \tag{6a}$$

$$\nabla_{\mathbf{u}} \psi = \frac{\partial \psi}{\partial \mathbf{u}} + \int_0^{t_f} \lambda^T \frac{\partial f}{\partial \mathbf{u}}\, dt. \tag{6b}$$

The ODE system (6a) has dimension $n_{\mathbf{x}}$, and $n_{\mathbf{u}}$ integrals must be solved in addition (which are less computationally expensive than ODEs). However, it is important to remark that system (6) should be solved for each function in the optimization problem, thus the computational effort is proportional to $n_x(1+n_{c_I})$. When there are more functions than control parameters the FS method typically performs better than the AS method and vice versa.

4.4 Jacobian Calculations

A key aspect for accurate and efficient sensitivity computation is the performance of Jacobian calculations. As can be seen in (5) and (6) partial derivatives are needed in each single step of the sensitivity calculation. Implicit ODE solvers also use the same partial derivatives to simulate (2b)-(2c).

System (1) has a special structure that can be exploited to solve its partial derivatives efficiently. The wells and pipelines states depend on each other indirectly through the algebraic variables $\mathbf{y}_{w,p}$ and $\mathbf{y}_p$. The dependency is determined by the routing configuration given by the set $\mathcal{R}$.

Taking advantage of the network structure, (7) shows how to calculate the non-zero members of the Jacobian by applying the chain rule.

$$\frac{\partial \dot{\mathbf{x}}_w}{\partial \mathbf{x}_w} = \frac{\partial f_w}{\partial \mathbf{x}_w} + \frac{\partial f_w}{\partial \mathbf{y}_{w,p}} \frac{\partial g_{w,p}}{\partial \mathbf{x}_w}, \quad \forall w \in \mathcal{W}, p \in R(w) \tag{7a}$$

$$\frac{\partial \dot{\mathbf{x}}_w}{\partial \mathbf{x}_p} = \frac{\partial f_w}{\partial \mathbf{y}_{w,p}} \frac{\partial g_{w,p}}{\partial \mathbf{x}_p}, \quad \forall w \in \mathcal{W}, p \in R(w) \tag{7b}$$

$$\frac{\partial \dot{\mathbf{x}}_p}{\partial \mathbf{x}_p} = \frac{\partial f_p}{\partial \mathbf{x}_p} + \frac{\partial f_p}{\partial \mathbf{y}_p} \sum_{w \in R(p)} \frac{\partial g_{w,p}}{\partial \mathbf{x}_p}, \quad \forall p \in \mathcal{P} \tag{7c}$$

$$\frac{\partial \dot{\mathbf{x}}_p}{\partial \mathbf{x}_w} = \frac{\partial f_p}{\partial \mathbf{y}_p} \frac{\partial g_{w,p}}{\partial \mathbf{x}_w}, \quad \forall (w,p) \in \mathcal{R} \tag{7d}$$

It should be noted that all partial derivatives in (7) are dense and dependent on the well and pipeline-riser models. Further sparsity exploitation depends on the specific implementation of these models.

When calculating the system sensitivity, the full evaluation of (7) is required depending on the chosen ODE solver. Implicit ODE solvers require complete Jacobians, while explicit methods require the evaluation of the Jacobians in a single direction. Efficiency gains can be obtained by calculating the Jacobians only in the required direction.

Automatic differentiation tools can be used to compute exact partial derivatives. Two modes are available, forward mode (FM) and reverse mode (RM), which are relatively efficient for calculating directional derivatives in the column space and in the row space of the Jacobian, respectively. On the other hand, the finite difference method described in Section 4.1 can be also used to get gradient approximations (Nocedal and Wright, 2006).

5. EXPERIMENTAL RESULTS

All computational experiments were executed on an Intel Core2Duo E8400 @ 3.00 GHz machine with 4.00 GB RAM running under Windows 7 SP1 64bits. Matlab 2012b was used to solve the ODE equations with the solver "ode23t" which implements the trapezoidal rule and is recommended for moderately stiff systems (Shampine et al., 1999). The single-shooting optimization problem (3) was solved with IPOPT v3.10.2 (Wächter and Biegler, 2006) interfaced to Matlab by the "OPTI" toolbox v1.71 (Currie and Wilson, 2012). The automatic differentiation tool used was ADiMat v0.5.8-3496 (Bischof et al., 2002).

5.1 Single Shooting Optimization Problem

The problem solved for the performance assessment of the tools is equivalent to problem (2) with $\psi(t_f) = \frac{1}{2}\int_0^{t_f} (\mathbf{x}-\mathbf{x}_S)^T Q (\mathbf{x}-\mathbf{x}_S) + (\mathbf{u}_c-\mathbf{u}_S)^T R (\mathbf{u}_c-\mathbf{u}_S)\, dt$ $+ \frac{1}{2}(\mathbf{x}(t_f)-\mathbf{x}_S) P (\mathbf{x}(t_f)-\mathbf{x}_S)$. Q is a diagonal constant positive definite matrix with $Q[k] = (1/\mathbf{x}_S[k])^2$ ($[k]$ denotes the kth component), and $R = r_0 I$ where r_0 is a positive scalar tuned to have the convergence rate of the system at the optimal solution similar to the open loop system dynamics. The matrix P is the positive solution of the Algebraic Ricatti Equation (ARE) with state cost Q and control cost R for the LTI system obtained after the linearization of (1) around the point $(\mathbf{x}_S, \mathbf{u}_S)$.

The set of constraints c_I describes linear input rate constraints. No state constraints are implemented. $u_{\min}$ and $u_{\max}$ are selected to bound the lift-gas rate to the steady-state feasible operational region. $\mathbf{x}_0$ and $\mathbf{x}_S$ satisfy $0 = f_c(\mathbf{x}_0, \mathbf{u}_0)$ and $0 = f_c(\mathbf{x}_S, \mathbf{u}_S)$, respectively, so they are steady-state solutions of the system for $\mathbf{u}_0$ and $\mathbf{u}_S$. Then the solution of this problem gives the optimal control sequence to change the system set-point. The time t_f is chosen longer than the system's dominant dynamics.

For the intent of this work routing is not relevant, so we assume it fixed. Therefore, the behavior of manifolds subsystems can be analyzed independently. To perform the tests we choose a system with 4 wells routed to a single manifold. The control inputs are parametrized as piecewise constant time functions with 15 equal intervals for each well, i.e., the optimization problem has 60 continuous decision variables. The complete model implementation, gradient calculation algorithms and the experimental results can be downloaded from (Codas et al., 2013).

5.2 Jacobian Efficiency/Accuracy Assessment

Jacobian calculations are required to simulate the system and to calculate the sensitivities. The "ode23t" implementation requires just an approximation of the Jacobian, which is checked and updated if necessary at each step (Shampine et al., 1999). The sensitivity calculations need a Jacobian-vector product, i.e., a member of the column space or row space. The forward mode (reverse mode) Jacobian calculation method can obtain the Jacobian-vector (vector-Jacobian) product without calculating the whole Jacobian, which is more expensive.

Table 1 illustrates the impact of the network structure exploitation on (7). The first 3 rows report the computational time for the Jacobian computation, while the last 3 rows show Jacobian-vector products. We observe that the structure exploitation makes the Jacobian calculation more efficient and calculating one product of the Jacobian is faster than calculating the Jacobian. The next analyses will always take advantage of this structure exploitation.

Table 1. Jacobian calculation time comparison

Time in seconds	Structure Exploitation Without	With	Improvement ratio
FM	0.29	0.07	4.15
RM	1.60	0.53	3.03
FD	0.06	0.03	2.48
FM (right product)	0.22	0.04	5.19
RM (left product)	0.50	0.42	1.20
FD (right product)	0.05	0.02	2.68

In order to assess the gradient calculation performance, Table 2 reports the time taken by the algorithms to calculate $\nabla\psi_{\mathbf{u}}$ and the relative distance of the solution to the gradient by FM/AS method. The RM method is discarded because it is faster to compute exact Jacobians with the FM method, even if only a vector-Jacobian product is required.

Table 2. $\nabla\psi_{\mathbf{u}}$ calculation performance

		Sensitivities					
		Running Time (s)			Relative Error(%)		
		FS	AS	FD	FS	AS	FD
$\partial f/\partial \mathbf{x}$	FM	127.1	11.1	14.8	0.33	0	26.2
	FD	1010.5	4.6	6.8	0.33	0.06	26.2

As expected, the AS method is faster than the others because the gradients are calculated for a unique function depending on 60 variables. While the AS method solves 2 relatively small ODEs, the FS method solves a big ODE system, and the FD method require $n_{\mathbf{u}} + 1 = 61$ ODEs.

Individual Jacobian-vector product calculations are not computed in the FS algorithm without calculating the Jacobian first. Since the ODE solver is instantiated once for all control parameters, it is faster to calculate the Jacobian once than calculating a Jacobian-vector product for each $u \in \mathbf{u}$.

In theory FM/AS and FM/FS provide exact gradients. The reported mismatch is a consequence of the ODE solver approximation. The approximations given by the FD sensitivity method are significantly different (26%) and are consequence of the parameter h_0 discussed in section 4.1. Several values of h_0 were tested obtaining similar results. Comparing the results given by the methods FM/AS and FD/AS, it is observed that the error introduced by the Jacobian approximation was not significant for this instance.

Table 3 shows the performance of the gradient calculation approaches when solving 100 instances of the optimization problem described in section 5.1 with IPOPT. The instances differ in the the initial conditions of the system, which is always in steady state, and in the set-points. The methods chosen for this comparison are FD/FD for being the easiest to implement, the FD/AS for being the fastest for gradient calculations and the FM/AS because it is the fastest with exact Jacobian calculation.

Table 3. Algorithms overall performance with IPOPT.

Index	Average (Sample standard deviation) FD/FD	FD/AS	FM/AS
Solving time (s)	106.62 (83.29)	13.07 (1.14)	20.60 (1.46)
IPOPT iterations	20.64 (15.18)	3.75 (0.44)	3.75 (0.44)
Gradient calls	23.38 (15.56)	5.75 (0.44)	5.75 (0.44)
Gradient time (s)	83.72 (58.21)	12.66 (1.09)	20.04 (1.42)
Jacobian calls	1777 (1345)	299.43 (21.79)	260.01 (18.46)
Jacobian time (s)	19.71 (14.89)	3.45 (0.30)	11.24 (0.80)

The solution time comparison indicates that the FD/AS method is the fastest. Specifically FD/AS takes over the FM/AS when calculating Jacobians. The relative lower accuracy of FD Jacobians does not affect significantly the number of Jacobian iterations and the number of gradient calls remains the same. A comparison of the returned objective values indicates that the solutions are equivalent. These results suggest that FD Jacobians are more efficient in spite of being relatively inaccurate.

The FD/FD algorithm did not converge for 37 instances. The indexes shown in Table 3 are related to the resources taken by IPOPT to find an optimal solution or to indicate the problem unbounded or infeasible. This result leads to the conclusion that the gradients obtained by this method are not accurate enough for this application.

6. DISCUSSION

The intention of this work is to explore dynamic optimization tools for the integration of dynamic simulators. Within the oil industry, models and simulators are often developed by different teams or companies, complicating the next integration step. Here we propose the integration of dynamical models developed by different groups, e.g., well models and pipeline models, for use in production optimization. Our approach takes advantage of the network structure of the system and disjoint optimization and modeling layers by making use of differentiation tools.

A trade-off between efficiency and accuracy is important. The Jacobians given by Automatic Differentiation (AD) tools are accurate up to machine precision, but they come at a relatively higher cost. Our computational experiments point out that Finite Difference (FD) approximations of Jacobians lead to faster convergence of the whole algorithm, even if more Jacobians are required due to the inaccuracy of the method. However, it is important to remember that the Optimal Jacobian Accumulation problem, which seeks to calculate Jacobians with the minimum number of operations, is itself NP-Complete and heuristics for making this calculation more efficient can be embedded in AD tools.

These results are encouraging for dynamic system integration through optimization where the FD method is commonly used due to its simplicity. It is more difficult to propagate gradients through independent systems. The development of AD tools or the direct gradient coding are complex processes which are error-prone. On the other hand, the FD method implementation effort is minimal, but special attention should be focused on the consequence of its approximation errors. While the FD method for sensitivity calculations should be avoided, the FD Jacobian calculation had the best performance, when tested on IPOPT. Similar results are expected with SQP algorithms such as SNOPT (Gill et al., 2005).

For future studies we propose to investigate the impact of FD approximations on Hessian approximations. In our experiments, no Hessian approximation was given to IPOPT. Further, exploitation of parallel computation should be considered. The Jacobian calculation can be performed concurrently taking advantage of the network structure. These structures appear frequently in process simulators and are a consequence of their physical topology.

Regarding the computational time spent in our experiments, real-time applications are promising. Matlab uses an interpreted language, which results in slower executions compared to compiled languages such as C/C++. Therefore, we expect that implementations using ODE solvers such as ACADO (Houska et al., 2011) and CVODES (Serban and Hindmarsh, 2003) comply with real-time requirements. In addition these tools provide sensitivity calculations funcionalities which diminish the coding effort.

7. CONCLUSIONS

In this work we presented a comparative study of tools for gradient calculations. Our experiments indicate that FD Jacobian calculations had the best performance, however, AD tools are not much slower. For this application FD Jacobians are accurate enough, but AD tools may be appropriated in applications requiring more accuracy. We exploit the system structure to enhance the Jacobian calculation efficiency. Further, we show that for this application adjoint sensitivity calculations is superior to the FD sensitivity calculation both in accuracy and speed.

8. ACKNOWLEDGMENT

We acknowledge the support of the Center of Integrated Operations at NTNU, Johannes Willkomm who gave support on ADiMat and Eduardo Camponogara for his valuable comments and corrections. Marco Aguiar is partially supported by a CNPq-Brazil scholarship.

REFERENCES

Biegler, L.T. (2010). *Nonlinear programming: Concepts, algorithms, and applications to chemical processes.* Mps-Siam Series on Optimization. Society for Industrial and Applied Mathematics (SIAM).

Biegler, L.T., Cervantes, A.M., and Wächter, A. (2002). Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4), 575–593. doi:10.1016/S0009-2509(01)00376-1.

Bischof, C., Bucker, H., Lang, B., Rasch, A., and Vehreschild, A. (2002). Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs. In *Proceedings of the Second IEEE international workshop on source code analysis and manipulation*, 65–72. IEEE Comput. Soc. doi: 10.1109/SCAM.2002.1134106.

Codas, A., Aguiar, M.A.A., Nalum, K., and Foss, B. (2013). Oil production system model implementation, gradient computation and computational results. URL `http://itk.ntnu.no/nolcos2013/oilnetwork.zip`.

Codas, A., Campos, S., Camponogara, E., Gunnerud, V., and Sunjerga, S. (2012). Integrated production optimization of oil fields with pressure and routing constraints: The Urucu field. *Computers & Chemical Engineering*, 46, 178–189. doi: 10.1016/j.compchemeng.2012.06.016.

Currie, J. and Wilson, D.I. (2012). OPTI: Lowering the barrier between open source optimizers and the industrial MATLAB user. In N. Sahinidis and J. Pinto (eds.), *Foundations of Computer-Aided Process Operations.* Savannah, Georgia, USA.

Diehl, M., Bock, H., Schlöder, J.P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4), 577–585. doi: 10.1016/S0959-1524(01)00023-3.

Eikrem, G., Aamo, O., and Foss, B. (2008). On instability in gas lift wells and schemes for stabilization by automatic control. *SPE Production & Operations*, 23(2). doi:10.2118/101502-PA.

Gill, P.E., Murray, W., and Saunders, M.a. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1), 99–131. doi:10.1137/S0036144504446096.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: Principles and techniques of algorithmic differentiation.* SIAM, Philadelphia, PA, 2nd edition.

Houska, B., Ferreau, H.J., and Diehl, M. (2011). ACADO toolkit-An open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3), 298–312. doi:10.1002/oca.939.

Imsland, L., Kittilsen, P., and Schei, T. (2010). Model-based optimizing control and estimation using Modelica models. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 31(3), 107–121. doi:10.4173/mic.2010.3.3.

Jahanshahi, E. and Skogestad, S. (2011). Simplified dynamical models for control of severe slugging in multiphase risers. In *18th IFAC World Congress*, 1634–1639. Milan. doi:10.3182/20110828-6-IT-1002.00981.

Johnson, R. (ed.) (1998). *The Handbook of Fluid Dynamics.* Mechanical engineering. Crc Press, Springer.

Nocedal, J. and Wright, S.J. (2006). *Numerical optimization.* Springer.

Serban, R. and Hindmarsh, A.C. (2003). CVODES: the sensitivity-enabled ODE solver in SUNDIALS. *ACM Transactions on Mathematical Software*, 5, 1–18.

Shampine, L.F., Reichelt, M.W., and Kierzenka, J.a. (1999). Solving index-1 DAEs in MATLAB and Simulink. *SIAM Review*, 41(3), 538–552. doi:10.1137/S003614459933425X.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57. doi: 10.1007/s10107-004-0559-y.