Computational Geosciences (2022) 26:329–349
https://doi.org/10.1007/s10596-022-10132-y

ORIGINAL PAPER

Derivative-free trust region optimization for robust well control under geological uncertainty

Thiago L. Silva[1,2] · Mathias C. Bellout[1] · Caio Giuliani[3] · Eduardo Camponogara[3] · Alexey Pavlov[1]

Received: 1 February 2021 / Accepted: 11 January 2022 / Published online: 15 February 2022

Abstract

A Derivative-Free Trust-Region (DFTR) algorithm is proposed to solve the robust well control optimization problem under geological uncertainty. Derivative-Free (DF) methods are often a practical alternative when gradients are not available or are unreliable due to cost function discontinuities, e.g., caused by enforcement of simulation-based constraints. However, the effectiveness of DF methods for solving realistic cases is heavily dependent on an efficient sampling strategy since cost function calculations often involve time-consuming reservoir simulations. The DFTR algorithm samples the cost function space around an incumbent solution and builds a quadratic polynomial model, valid within a bounded region (the trust-region). A minimization of the quadratic model guides the method in its search for descent. Because of the curvature information provided by the model-based routine, the trust-region approach is able to conduct a more efficient search compared to other sampling methods, e.g., direct-search approaches. DFTR is implemented within FieldOpt, an open-source framework for field development optimization, and is tested in the Olympus benchmark against two other types of methods commonly applied to production optimization: a direct-search (Asynchronous Parallel Pattern Search) and a population-based (Particle Swarm Optimization). Current results show that DFTR has improved performance compared to the model-free approaches. In particular, the method presented improved convergence, being capable to reach solutions with higher NPV requiring comparatively fewer iterations. This feature can be particularly attractive for practitioners who seek ways to improve production strategies while using an ensemble of full-fledged models, where good convergence properties are even more relevant.

Keywords Derivative-free trust-region algorithm · Well control optimization · Robust optimization under geological uncertainty

1 Introduction

Production optimization problems are important because they complement general reservoir management efforts with systematically-derived information regarding how to operate wells optimally given a determined objective. Production optimization problems involving well controls are typically computationally-demanding since cost function evaluations ordinarily require time-consuming reservoir simulations. Cost function objectives associated with production problems are usually formulated using continuous well controls, e.g., bottom-hole pressures (BHP) and/or production/injection rates, as independent variables. For the most part, these objectives involve maximizing profit, e.g., increasing either the net present value or other more sophisticated economic targets [4, 20]. However, objectives can also include other performance measures and costs, e.g., improving water-flooding sweep, reducing topside energy consumption and environmental objectives. In general, for water-flooding cases, optimal control solutions imply increasing hydrocarbon production while reducing water production and injection.

Production optimization problems with continuous controls are generally considered to exhibit smooth cost function surfaces. These surfaces have been reported to organize in ridge-like topologies consisting of multiple local optima, i.e., different optimal solutions have been observed to yield similar cost function values lying within a plateau range [14]. On a related note, this problem characteristic has been suggested as a general point of support for using gradient-based methods for production optimization [39]. These

✉ Thiago L. Silva
thiago.silva@sintef.no

Extended author information available on the last page of the article.

gradients are most efficiently calculated using an adjoint formulation [23, 25]. However, since the problem is smooth and the control variables are continuous, standard finite-differences methods may be applied, though at a significant computational cost (as discussed later), when the simulator does not provide corresponding sensitivities.

Though calculating the gradient using the adjoint formulation is highly efficient compared to finite-difference and other alternatives, the additional computation necessary to assemble the gradient after simulation has a similar computational runtime as the forward simulation. Moreover, when not readily available from the reservoir simulator, computing adjoint gradients may be difficult because implementing the procedure to assemble these gradients requires an intrusive development within the simulator code. Still, when adjoint-gradients are available and reliable, they can be fairly useful for large-scale production optimization since their added computational cost does not scale with problem dimension [23]. Adjoint-based methods can be broadly split into single shooting methods, see e.g., [23] for a review, or multiple shooting approaches [6, 34]. While the first considers only the well controls as decision variables, the latter embeds the reservoirs states along the simulation time into the optimization problem, thereby increasing the problem complexity but allowing an easier handling of output constraints.

Clearly, a common concern for gradient-based methodologies, in particular those that rely on adjoints, is that their search efficiency and ultimate performance are highly dependent on the accuracy of the calculated gradient. Adjoint-gradient accuracy may be compromised by simulation-based constraints that operate at the simulation time-step level (as opposed to the optimization control-step level). This type of constraints is often imposed on well operations in realistic production strategies. The issue of lost sensitivity concerns not only the enforcement of production- and/or economic-type of constraints such as liquid production or water-cut limits, in the case of water flooding, but also industry-standard heuristic rules of well control switching during simulation. (Well group control heuristics and other facility-wide constraints that may additionally be imposed on the problem are here disregarded.) In particular, [36] shows that enforcing simulator-based economic constraints may trigger non-differentiable unscheduled changes during reservoir simulation. Crucially, these discontinuities can subsequently lead to inconsistencies within the adjoint-gradient formulation and this can eventually translate into decreased performance of a gradient-based algorithm.

Although production optimization is considered a relatively smooth problem, and derivative information can therefore in theory be estimated numerically in a straightforward manner, e.g., using finite differences, doing so is generally not practical for even medium-scale problems because of the computational cost associated with each objective function evaluation. Along these lines, stochastic gradient approximation methods have been studied extensively [5, 39] as a way to estimate cost function sensitivity in a computationally efficient manner. Even disregarding computationally-expensive function evaluations, typical gradient estimation efforts for the control problem may be difficult and unreliable, e.g., due to the difficulty in selecting appropriate perturbation sizes [13] in addition to various types of simulation error that can influence the final cost function value [37].

Multiple methodologies rely on gradients for effective solution of the well control problem, see e.g., [4, 14, 24, 35, 38]. As mentioned, however, gradient information for this type of problem may be unreliable due to numerical errors, cost function nonsmoothness due to simulation-based constraints, or other inaccuracies. From a practical point of view, derivative-free approaches are attractive because they are relatively easy to implement, are noninvasive to simulator code and many of the underlying search strategies are straightforward to parallelize [13]. More importantly, these methods are useful for problems where derivative information is not readily available, e.g., in the case of well placement optimization, and/or when an expanded or global search is required. These approaches are often split into deterministic and stochastic methods. In optimization terms, the stochastic component in the latter type serves to avoid local minima in contrast to gradient-based methods which only have local-search property and are heavily-dependent on initial controls [39].

Though generally less efficient than gradient-based methods, several derivative-free approaches offer global-search properties and most can be implemented efficiently in a distributed environment. Derivative-free methods can be broadly divided into direct-search and model-based type of methods, see e.g., [10]. Direct-search methods conduct their search for improved solutions by making immediate use of their sampled feasible space information according to a predetermined logic. Among these, pattern-search methods sample the feasible space using stencils and exhibit local convergence properties given their stencils comply with specific geometric conditions [16]. Other commonly-used direct-search methods, e.g., Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) rely on population-based approaches that involve a large number of sampling points. Model-based methods, on other hand, construct a model of the objective function with the aim of incorporating higher-order curvature information about the solution space to improve the search.

In this work, a Derivative-Free Trust-Region (DFTR) algorithm is presented to solve realistic well control optimization problems. The DFTR algorithm relies on

a model-based search strategy that uses a quadratic interpolation model to approximate the true cost function within a certain neighborhood (the trust-region). The use of a model to guide the search not only allows for a wider exploration of the search space, but also provides a certain level of robustness against cost function inaccuracies and nonsmooth characteristics. Finally, compared to methods relying on direct-search, a model-based approach can be more effective in reaching reasonably good solutions with fewer function evaluations. This is particularly important when cost function evaluations involve numerical simulations, and for real-time applications where good-enough solutions need to be calculated within short spans of time.

Results from this work demonstrate that DFTR is a practical algorithm with good convergence performance that can be used to obtain results on realistic cases in an efficient manner. In practice, this means that a sufficiently good solution can likely be obtained even though the optimization is halted, e.g., due to limited time or computational budget, before specified convergence criteria have been met. Specifically, we show that the iterative progression of DFTR outperforms the other algorithms tested in this work, a pattern-search and population-based, for the presented case studies. Even when the DFTR solutions are not locally optimal, they are comparable to the solutions using other approaches that were obtained using a considerably higher number of simulations. From a practical point of view, this is a reasonable result for the production optimization problem.

The next section presents a formulation for the well control problem, and provides a broad overview of common solution approaches in the literature for well control optimization. In particular, it describes the use and typical challenges associated with gradient-based and derivative-free approaches in this context. The subsequent section provides a detailed explanation of the various operations comprising the DFTR algorithm. Finally, the last two sections describe the test case, as well as present and analyze case results and offer some conclusions and discussion of further work.

2 Well control optimization

Well control optimization consists of determining the well controls, i.e., production and injection rates or bottom-hole pressures (BHPs), that maximize an objective function of interest, commonly the cumulative oil production or the net present value (NPV). With the right choice of well controls the economic performance of the reservoir can be considerably improved for the field life cycle. In this section, we first formulate the well control optimization problem, including the objective, the control variables and the constraints. Then, we provide a justification for the proposed method in light of currently available methods in the literature for well control optimization.

2.1 Nominal problem formulation

The objective function is an economic function that contains certain monetary parameters such as the oil price and the costs associated with injection and processing of water. The objective function employed in this work is the NPV, which results from a time series of cash flows over a period of time. For two-phase flow of oil and water, the NPV is defined by:

$$\mathrm{NPV}(\mathbf{q}_\mathrm{u}) = \sum_{t=1}^{T} \frac{\Delta t}{(1+d)^{t\cdot\Delta t/\tau}} \cdot \left[r_\mathrm{op}^t \cdot q_\mathrm{op}^t - r_\mathrm{wp}^t \cdot q_\mathrm{wp}^t - r_\mathrm{wi}^t \cdot q_\mathrm{wi}^t \right] \quad (1)$$

where $\mathbf{q}_\mathrm{u}$ is a vector of the total oil production rate q_op^t, the total water production rate q_wp^t, and the total water injection rate q_wi^t for all time steps $t = 1, \ldots, T$. The associated oil price and water treatment cost are r_op^t and r_wp^t, respectively, whereas the water injection cost is r_wi^t. The total field production time is split into a set of time steps $\mathscr{T} := \{1, \ldots, T\}$ with a step size of Δt, typically corresponding to a couple of days, whereas the parameter τ is a normalization term, typically taken as the number of days in a year. The revenue obtained with the field production is discounted over time by a discount factor term d.

The decision variables $\mathbf{u}$ are the controls of the wells in $i \in \mathscr{W} := \{1, \ldots, N_\mathrm{w}\}$ for a set of control steps $k \in \mathscr{K} := \{1, \ldots, N_\mathrm{t}\}$. The controls of each well $\mathbf{u}_i$ are piecewise-constant values for the well bottom-hole pressure, one for each control step $k \in \mathscr{K}$. There is a surjective function $\kappa(t) : \mathscr{T} \to \mathscr{K}$ which maps the simulation steps $\mathscr{T}$ into control steps in $\mathscr{K}$, with $|\mathscr{K}| \leq |\mathscr{T}|$ and $N_\mathrm{t} \leq T$. The total number of well controls N_u is given by the product of the total number of wells and the number of control steps, i.e., $N_\mathrm{u} = N_\mathrm{w} \cdot N_\mathrm{t}$. Notice that the number of variables of the well control optimization problem is proportional to the number of control steps, which means that a refined control strategy has more degrees of freedom at the expense of an increase in the problem dimension.

The nominal well control optimization problem for a single reservoir realization is formulated as follows:

$$\underset{\mathbf{u}}{\text{maximize}} \quad \mathrm{NPV}(\mathbf{q}_\mathrm{u}) \quad (2a)$$

subject to:

$$R(\mathbf{x}_0, \mathbf{x}_\mathrm{u}, \mathbf{u}) = 0 \quad (2b)$$

$$\mathbf{q}_\mathrm{u} = Q(\mathbf{x}_\mathrm{u}) \quad (2c)$$

$$\mathbf{u}_\mathrm{lb} \leq \mathbf{u} \leq \mathbf{u}_\mathrm{ub} \quad (2d)$$

$$\mathbf{c}(\mathbf{u}) \geq 0 \quad (2e)$$

where $\text{NPV}(\mathbf{q}_u)$ is the objective function defined in Eq. 1, $\mathbf{x}_u$ are the states of the system of reservoir equations $R(\mathbf{x}_0, \mathbf{x}_u, \mathbf{u})$, over the production planning horizon, for the initial condition $\mathbf{x}_0$ and the control sequence $\mathbf{u}$, and $\mathbf{q}_u$ are the field flow rates computed from the states using the equation $Q(\mathbf{x}_u)$. The constraints (2d) and (2e) are bounds on the well controls and some additional inequality constraints respectively. The lower and upper bounds $\mathbf{u}_{lb}$ and $\mathbf{u}_{ub}$ are needed to avoid infeasibility of the control strategy, such as too low pressures in the producers causing lift die-out, or pressures at the injectors exceeding the maximum allowable formation or equipment pressures. The inequality constraints (2e) on the input variables $\mathbf{u}$ are typically employed to impose field-wide constraints, such as limits on the total water injected into the field and also environmental constraints.

2.2 Robust problem formulation

In addition to the nominal well control problem, we also formulate the robust well control optimization problem. For that, we will use an ensemble of reservoir models to address uncertainty. Then, instead of maximizing one NPV, as in the previous section, we propose the maximization of the expected value for the NPV. As all the realizations of the ensemble are considered equally probable, the expected value is given by the average of the NPV computed across all realizations, namely:

$$f(\mathbf{q}_u) = \frac{1}{M} \sum_{i=1}^{M} \text{NPV}_i(\mathbf{q}_u), \tag{3}$$

in which $\text{NPV}_i(\mathbf{q}_u)$ is the economic function resulting from the i^{th} realization, assuming a well control sequence $\mathbf{u}$, and M is the number of reservoir realizations.

The constraints which are written on the decision variables, such as bounds (2d) and the inequality constraints on control variables (2e), are kept unaltered. Constraints which depend on the simulation outputs have to be dealt with individually. The feasibility of the simulation, established by Eq. 2b, has to be enforced for every model of the ensemble. The robust formulation for well control optimization is as follows:

$$\underset{\mathbf{u}}{\text{maximize}} \quad f(\mathbf{q}_u) \tag{4a}$$

subject to:

$$R_i(\mathbf{x}_0, \mathbf{x}_u, \mathbf{u}) = 0, \qquad \forall i = 1, \ldots, M \tag{4b}$$

$$\mathbf{q}_u = Q(\mathbf{x}_u) \tag{4c}$$

$$\mathbf{u}_{lb} \leq \mathbf{u} \leq \mathbf{u}_{ub} \tag{4d}$$

$$\mathbf{c}(\mathbf{u}) \geq 0 \tag{4e}$$

2.3 Solution methods

Production optimization problems have been treated using both gradient-based [2, 3, 26, 32] and derivative-free approaches [13, 22]. While many gradient-based approaches rely on adjoints for the efficient computation of derivatives [23], derivative-free methods are commonly accelerated making extensive parallelization of the objective function sampling routines. The following discussion focuses primarily on features and associated limitations of typical methodologies for gradient-based and derivative-free well control optimization, which can be either applied in a single reservoir or an ensemble of models. In this respect, gradient-based approaches are discussed with regard to adjoint-gradient computation and accuracy, while derivative-free approaches are treated with regard to local/global search attributes and large objective function sampling requirements.

The well control optimization problems (2) and (4) require the solution of the set of differential (2b) and (4b), i.e., one and multiple reservoir simulations, respectively, for each objective function evaluation. When the reservoir simulator provides the cost function gradients associated to the well controls, it is convenient to use a gradient-based approach, which typically exhibits faster convergence rate and provides robust guarantees for optimality. Within several studies, the adjoint-gradient has been used by various numerical algorithms to solve production optimization problems. Several of these applications involve the Sequential Quadratic Programming (SQP) algorithm [2, 6, 11, 12, 32, 37], while a few use the Method of Moving Asymptotes (MMA) [3], among others [39].

A simulation-based error source impacting overall adjoint-based algorithm performance, e.g., leading to premature convergence, is inconsistency in adjoint-gradient accuracy during the optimization process. This inconsistency stems from the adaptive time-stepping strategy implemented by the simulator, which can generate time step refinements that vary significantly from one simulation to the next during optimization [37]. The varying time discretizations lead to altered objective function value calculations that also arbitrarily affect the corresponding gradients across subsequent simulations. In a wider sense, this error source is also related to the implementation of simulation-based constraints, e.g., well control-switching constraints, which can cause a high degree of time step refinement, particularly if time step sizes and tuning parameters are not set appropriately. Though this type of inconsistency may be avoided by using a fixed time-stepping strategy, simulations of large-scale production systems are usually run with adaptive strategies as a trade-off between substantial performance improvement and the need to accurately capture significant changes in dynamic behavior due to

common phenomena, e.g., water-breakthrough and/or gas coning.

Overall, the application of gradient-based approaches can be challenging due to inaccurate calculation of derivatives caused by simulation errors or impossible if well control sensitivities are unavailable. In these situations, approaches based on derivative-free methods become a practical alternative because of their robust search properties.

In this work we propose a derivative-free model-building method relying on a trust-region search approach [7] for well control optimization with control bound constraints. The Derivative-Free Trust-Region algorithm (DFTR) is applied to both a nominal reservoir case and for an ensemble of models, as defined in (2) and (4), respectively. The method works by constructing a polynomial interpolation model with simulated data-points which provides a locally valid approximation of the underlying cost function. A minimization of the approximate model is then performed to find the next solution candidate. Compared to direct-search sampling methods, the inference of curvature information through the approximate model is expected to result in a more efficient search of the feasible space.

DFTR can be compared to the SQP method in that both methods build a quadratic model of the objective function. In its scheme, the SQP relies on a Hessian approximation built using gradients from previous iterations. As previously discussed, this approximation is susceptible to gradient inconsistencies that may result in inaccurate curvature information which is likely to degrade search performance [37]. The model-building process of the DFTR method, on the other hand, relies on a polynomial interpolation of objective function values. The subsequent search for new tentative solutions relying on this interpolation is regarded as less sensitive to objective function noise and error. New iterates are found through efficient minimization of the quadratic interpolation model, and the model updates are controlled in a dynamic way by considering a measure comparing predicted and actual improvement in the objective.

In general, the motivations for applying the DFTR algorithm for well control optimization are that (1) the adjoint-gradient may not be entirely reliable nor sufficiently accurate in the case of problems with typical simulation-based constraints, which are common in realistic applications; that (2) direct-search methods are often inefficient for realistic applications, despite substantial parallel resources; and that (3), in comparison, the DFTR approach, though involving complex algorithms for model-building, nevertheless enables a robust search procedure that can yield significant improvement over few iterations. A detailed presentation of the proposed DFTR algorithm is given next.

3 Derivative-free trust-region algorithm

In this section we present the algorithm implemented to solve the well control optimization problem formulated previously. Though the individual elements of the proposed algorithm are present in the literature, their orchestration into an efficient methodology for well control optimization is regarded as the main contribution of this work. Derivative-free trust-region algorithms conduct an optimization by modeling response surfaces using results from black-box functions (obtained by simulation). These methods attempt to achieve greater increase of the cost function using fewer time-consuming simulations, compared to direct-search methods which do not include cost function curvature information explicitly in their searches.

Here, the objective function will be denoted f, which can be any black-box function of interest (for example the NPV, or its average). The variables of the problem will be denoted in a single vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. As the algorithm runs, f is sampled and their values are used to build a model. At each iteration k, the incumbent model m_k of the objective is optimized inside the trust-region, which is a ball around the iterate $\mathbf{x}_k$. Only then is f evaluated once more, so steps can be accepted or rejected as a new iterate. Through the iterations, the model is improved by incorporating more recent evaluations of f.

Next we turn to the issue of generating the models m_k, which interpolate the objective f in a set of points $\mathscr{Y}_k$ using quadratic polynomials. For the sake of notation, we will omit the iteration number k for models and interpolation sets during this explanation. In order to state the interpolating equations, we define the basis employed for a problem of n variables:

$$\begin{aligned}\phi &= \{\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \ldots, \phi_L(\mathbf{x})\} \\ &= \left\{1, x_1, x_2, \ldots, \frac{1}{2}x_1^2, x_1x_2, \frac{1}{2}x_2^2, x_1x_3, \ldots, \right. \\ &\qquad \left. x_{n-2}x_n, x_{n-1}x_n, \frac{1}{2}x_n^2\right\}, \end{aligned} \tag{5}$$

which includes all monomials of degree at most 2. The model is a linear combination of these terms

$$m(\mathbf{x}) = \sum_{i=0}^{L} \alpha_i \phi_i(\mathbf{x}). \tag{6}$$

For every point y_i of $\mathscr{Y}$, $m(y_i) = f(y_i)$. Then, the coefficients of Eq. 6 can be obtained solving this interpolation condition for all the points:

$$\begin{aligned} \alpha_0\phi_0(\mathbf{y}_0) + \alpha_1\phi_1(\mathbf{y}_0) + \cdots + \alpha_L\phi_L(\mathbf{y}_0) &= f(\mathbf{y}_0) \\ \alpha_0\phi_0(\mathbf{y}_1) + \alpha_1\phi_1(\mathbf{y}_1) + \cdots + \alpha_L\phi_L(\mathbf{y}_1) &= f(\mathbf{y}_1) \\ \vdots \qquad\qquad\qquad\qquad\qquad\qquad &\quad \vdots \\ \alpha_0\phi_0(\mathbf{y}_L) + \alpha_1\phi_1(\mathbf{y}_L) + \cdots + \alpha_L\phi_L(\mathbf{y}_L) &= f(\mathbf{y}_L), \end{aligned}$$

which is a system of linear equations that can be written

$$M(\phi, \mathscr{Y})\alpha = \mathbf{f}(\mathscr{Y}). \tag{7}$$

The total number of terms of the basis is $L = (n+1)(n+2)/2$, so for this system to be fully determined $\mathscr{Y}$ should have the same number of interpolation points. For our application case with 40 variables, the number of terms of the basis is $L = 861$.

It is, of course, impractical to demand that many reservoir simulations before being able to take steps. Indeed, quadratic interpolation would have no place in the derivative-free optimization literature if it were not for the use of under-determined models. The DFTR algorithm in this work follows [8] in allowing interpolation sets $\mathscr{Y}$ with varying number of points. In fact, the first model is built using only two points. As the algorithm progresses, more simulations are performed, so the set $\mathscr{Y}$ is gradually increased. Up until $n+1$ points, the models are computed by a coarse finite differences method, involving only part of the linear terms of the basis (5). These may be called sub-basis models.

When the trust-region algorithm reaches iteration k with an iterate $\mathbf{x}_k$ and the number of points which are available for model building is between $n+1$ and $L = (n+1)(n+2)/2$, another computation is employed. In this case, the number of interpolating conditions is insufficient to fully determine a quadratic model. The remaining degrees of freedom are used to compute a model with minimum Frobenius norm Hessian around $\mathbf{x}_k$, that is, the model computed is of the form

$$m(\mathbf{x}_k + \mathbf{s}) = m(\mathbf{x}_k) + \nabla m(\mathbf{x}_k)^T s + \frac{1}{2} s^T \nabla^2 m(\mathbf{x}_k) s,$$

where the Hessian matrix $\nabla^2 m(x_k)$ has minimum Frobenius norm. This is obtained by first shifting the set of interpolation points $\mathscr{Y} = \{y_0, y_1, y_2, \ldots\}$ to the center of the trust-region:

$$\widehat{\mathscr{Y}} = \left\{\hat{\mathbf{y}}_0, \hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots\right\} = \left\{\mathbf{y}_0 - \mathbf{x}_k, \mathbf{y}_1 - \mathbf{x}_k, \mathbf{y}_2 - \mathbf{x}_k, \ldots\right\}.$$

Then, the system (7) is written partitioning the interpolation matrix M in two: M_L and M_Q, with columns corresponding to the linear and to the quadratic terms respectively. The interpolation condition can be written

$$M_L\left(\phi, \widehat{\mathscr{Y}}\right)\alpha_{\mathbf{L}} + M_Q\left(\phi, \widehat{\mathscr{Y}}\right)\alpha_{\mathbf{Q}} = \mathbf{f}(\mathscr{Y}).$$

The model is computed by solving the quadratic problem

$$\begin{aligned} &\text{minimize} && \|\alpha_{\mathbf{Q}}\|^2 \\ &\text{subject to:} \\ & && M_L\left(\phi, \widehat{\mathscr{Y}}\right)\alpha_{\mathbf{L}} + M_Q\left(\phi, \widehat{\mathscr{Y}}\right)\alpha_{\mathbf{Q}} = \mathbf{f}(\mathscr{Y}). \end{aligned}$$

Such models capture not only first-order information, but also some curvature. We must stress that while there are different approaches for the formulation of models [30, 31], the possibility of under-determined quadratic interpolation is a main feature for the success of many model-based derivative-free algorithms available.

As we fixed the basis of monomials at the beginning, the matrices $M(\phi, \mathscr{Y})$, M_L and M_Q depend only on the set of points $\mathscr{Y}$. Special maintenance procedures are performed to ensure that these matrices are well-conditioned so the interpolation can be solved accurately. These procedures operate only on the set of points: the inclusion of some points may be rejected and some points may need to be replaced in order to maintain good conditioning. These procedures are explained in [9] and [10, Chapter 6] and will not be detailed here.

Following [8], we chose to perform the model maintenance in blocks, by the use of Newton Fundamental Polynomials. In this approach, quadratic monomials are only included after all the linear ones. This also ensures that any model with more than $n+1$ points is a good first-order representation of the objective f. Another important consequence is that a model with at least $n+1$ points, has bounded modeling error:

$$|f(\mathbf{x}+\mathbf{s}) - m(\mathbf{x}+\mathbf{s})| \leq \kappa_{ef}\Delta^2, \qquad \|\mathbf{s}\| < \Delta, \tag{8a}$$

$$\|\nabla f(\mathbf{x}+\mathbf{s}) - \nabla m(\mathbf{x}+\mathbf{s})\| \leq \kappa_{eg}\Delta, \qquad \|\mathbf{s}\| < \Delta, \tag{8b}$$

where κ_{ef} and κ_{eg} depend on the function, and Δ is the radius of the trust region. A model satisfying such bounds is said to be Fully Linear (FL). Having defined this, we may turn to the description of the derivative-free trust-region algorithm.

At the beginning of each iteration, the model m_k is optimized inside of the trust-region, which is a ball of radius Δ_k around the iterate x_k:

$$\underset{\mathbf{s}}{\text{maximize}} \quad m_k(\mathbf{x}_k + \mathbf{s}) \tag{9a}$$

subject to:

$$\|\mathbf{s}\|_\infty \leq \Delta_k \tag{9b}$$

$$\mathbf{x}_{\text{lb}} \leq \mathbf{x}_k + \mathbf{s} \leq \mathbf{x}_{\text{ub}} \tag{9c}$$

This is called the trust-region subproblem. We elected the infinity norm because it aligns with variable bounds, making such constraints easier to handle.[1] The subproblem above may be NP-hard, depending on the model [29], but this is not a big issue in this context. Convergence can be obtained with a coarse approximation of the solution of (9): it suffices to take the steepest ascent direction. Also, as the dimension of the problem is quite limited in general, it pays to put some effort in solving the trust-region subproblem as best as possible in order to save function evaluations (which involve lengthy reservoir simulations).

[1] See [7, Section 7.8] for a discussion on the choice of trust-region norms

Notice that this maximization involves only the model m_k, available at the beginning of the iteration, to compute the trial-point $\mathbf{x}_k^+ = \mathbf{x}_k + \mathbf{s}_k$. Only then, the objective is evaluated once more, rendering $f(\mathbf{x}_k^+)$ which may be accepted or rejected as the next iterate, by the use of the following factor:

$$\rho_k = \frac{f(\mathbf{x}_k^+) - f(\mathbf{x}_k)}{m_k(\mathbf{x}_k^+) - m_k(\mathbf{x}_k)}, \tag{10}$$

which assesses both the ascent obtained and the degree to which the model agrees with the actual function [10, Chapter 10], [27, Chapter 4]. Notice that point $\mathbf{x}_k^+$ is computed only if $\mathbf{x}_k$ is not the optimum of model m_k. Since it is computed maximizing model m_k, $m_k(\mathbf{x}_k^+) > m_k(\mathbf{x}_k)$ and the denominator of Eq. 10 is strictly positive. From this fact, follows that not only the numerator $f(\mathbf{x}_k^+) - f(\mathbf{x}_k)$, but also the resulting ρ_k measure ascent of f from $\mathbf{x}_k$ to $\mathbf{x}_k^+$ (and nonpositive values signal lack of ascent). Since ρ_k is the ratio between actual ascent and predicted ascent, it also serves the purpose of measuring how the model agrees with the actual objective. Accurate models render a ρ_k close to 1, and less accurate models will result in ρ_k further from 1 (either too high or too low).

If factor (10) is large enough ($\rho_k \geq \eta_1 > 0$), then the step is accepted as the next iterate ($\mathbf{x}_{k+1} = \mathbf{x}_k^+$). It may also be accepted with a threshold $\eta_0 \geq 0$ smaller than η_1: if $\rho_k > \eta_0 \geq 0$, then $\mathbf{x}_{k+1} = \mathbf{x}_k^+$ as long as the model is already FL (and the bounds (8) hold). Upon the acceptance of a new iterate, the model is necessarily updated so it interpolates f at $\mathbf{x}_{k+1}$.

At this point, the derivative-free approach distances itself from traditional gradient-based trust-region methods. If ρ_k is small, it may be either because the radius is too big (and the local approximation degrades), or because the model is not accurate (not FL). If the model is FL, reducing the radius will reduce the error, according to Eqs. 8a and 8b. That can not be said if the model is not yet FL: in such case, the model must be improved with appropriate maintenance procedures [9].

To that end, we first try to include in the model the trial-point just rejected, since the simulation has already been performed and it conveys information that the model is lacking. If the model maintenance procedures are unable to include such point in the interpolation set, the model must be improved with the computation of another point (with the corresponding objective function evaluation). We chose to make model improvements incremental: they include only one point at a time and it may take a few iterations until the model becomes FL. With this scheme the algorithm performs at most two simulations per iteration.

Notice that the management of points in the model plays a central role in the method as it affects both the method's robustness and efficiency. Even the points evaluated which do not improve the model are stored in a list of cached points for further use in future iterations. As storing many points in memory might be problematic for problems with many decision variables, the oldest points, or those which lie further away from the trust-region, are discarded after some time. Further, there is a model maintenance procedure in the algorithm that marks the model as old if the distance of the points being used in the model is larger than a pre-specified distance to the trust-region area. If so, the model is rebuilt using sampled points that are nearer the incumbent solution. This criterion is a tuning parameter of the algorithm that weighs model accuracy and computational cost, i.e. allowing models to use only the points which are near the trust-region center could yield more accurate approximations, but at the expense of more frequent model updates. An alternative approach, taking more emphasis on the distance of the interpolation points is addressed in [33].

The value of ρ_k is also used to decide on the radius update. If $\rho_k \geq \eta_1$, the radius may be increased. In this case we set

$$\Delta_{k+1} = \min\left[\max(\Delta_k,\ 2\|\mathbf{s}_k\|),\ \Delta_{\max}\right],$$

where the value $\Delta_{\max}$ defines the maximum radius allowed through the algorithm run. This scheme allows steps to grow, while trying to avoid unnecessary increases of the radius.

If $\rho_k < \eta_1$ and the iteration model m_k is FL, the error may be reduced decreasing the radius (according to Eqs. 8a and 8b). In this case we set

$$\Delta_{k+1} = \frac{1}{2}\Delta_k.$$

If $\rho_k < \eta_1$ and the model can not be guaranteed to be FL, there is no point in reducing the radius, so it is kept the same ($\Delta_{k+1} = \Delta_k$). We also defer the radius decrease if the trial point is successfully included in the interpolation set.

A criticality measure is used to assess the convergence of the algorithm. For unconstrained problems, the norm of the gradient is used. For problems with bounds on variables, a practical measure is the projected gradient:

$$\sigma[m_k, \mathbf{x}_k] = \left\|\max\left\{\min[\mathbf{x}_k + \nabla m_k(\mathbf{x}_k),\ \mathbf{x}_{\text{ub}}],\ \mathbf{x}_{\text{lb}}\right\} - \mathbf{x}_k\right\|,$$

which is equivalent to the distance of the point $\mathbf{x}_k$ and the projection of the point $\mathbf{x}_k + \nabla m_k(\mathbf{x}_k)$ onto the feasible set [7, p. 450].

Models are expected to be accurate when they can be certified to be FL, so the error is bounded by Eqs. 8a and 8b. When it appears that convergence is near, the gradient $\nabla m(\mathbf{x}_k)$ of the model becomes small, so the gradient $\nabla f(\mathbf{x}_k)$ of the actual objective is small, provided that the bound (8b) is sufficiently tight. If the radius Δ_k is overly large, the two gradients may be far apart, both in magnitude

and direction. This issue, if not corrected, may slow the algorithm to the point of preventing convergence. This is addressed by the criticality step [33], [10, Chapter 10].

When convergence appears to be near ($\sigma[m_k, \mathbf{x}_k] < \epsilon_c$, for a small $\epsilon_c > 0$), the criticality step is performed to ensure that the radius of the trust-region is comparable to the criticality measure, reducing it if necessary. Since $\sigma[m_k, \mathbf{x}_k]$ depends on the model m_k, which in turn depends on the radius Δ_k, the reduction of the radius must be iterative. Therefore, the radius is reduced by a factor $\omega \in (0, 1)$, so $\Delta^{(1)} = \omega\Delta_k$, and the model maintenance procedures make a new model $m^{(1)}$, FL in a trust-region with radius $\Delta^{(1)}$. If the criticality measure of the new model is not sufficiently high ($\sigma[m^{(1)}, \mathbf{x}_k] < \mu\Delta^{(1)}$, for $\mu > 0$), the procedure is repeated, otherwise it can be stopped with a new model $m^{(1)}$ and radius $\Delta^{(1)}$. This procedure is formalized in Algorithm 1, which was based in [10, Chapter 10]. In the end, it is possible to increase the radius a bit, so the reduction is not excessive.

Algorithm 1 Criticality step.

Inputs: Initial model m_k, and trust-region radius Δ_k^{bct}. Iterate $\mathbf{x}_k$ of main trust-region algorithm.
Parameters: $\mu > \beta > 0$ for the acceptance of the radius, fixed throughout the main algorithm, and $\omega \in (0, 1)$ for reducing the radius.
Set radius $\Delta^{(1)} = \Delta_k^{\text{bct}}$.
Generate model $m^{(1)}$ by improving model m_k^{bct} until it is FL in a trust-region of radius $\Delta^{(1)}$.
for $i = 1, 2, \ldots$ **do**
  if $\sigma[m^{(i)}, \mathbf{x}_k] > \mu\Delta^{(i)}$ **then**
    break
  else
    Update radius $\Delta^{(i+1)} = \omega\Delta^{(i)}$.
    Generate model $m^{(i+1)}$ by improving model $m^{(i)}$ until it is FL in a trust-region of radius $\Delta^{(i+1)}$.
  end if
end for
Increase radius

$$\Delta_k = \min\left[\max(\Delta^{(i)},\ \beta\sigma[m^{(i)}, \mathbf{x}_k]),\ \Delta_k^{\text{bct}}\right].$$

Set $m_k = m^{(i)}$.
return radius Δ_k and model m_k.

Now we are in position of presenting the full derivative-free trust-region algorithm (Algorithm 2). It takes from [8] the use of Newton Fundamental Polynomials to manage the set of interpolation points and model improving procedures. It is also based on the framework of [10, Chapters 10 and 11], which includes an explicit criticality step. Since the criticality step may be invoked, changing the model and the trust-region radius, we introduced the superscript "bct" to denote radius and model Before Criticality Test. A computational analysis of the proposed DFTR algorithm in a batch of 300+ mathematical optimization problems may be found in the recently accepted paper [19], which was a development from the present algorithm. Preliminary computational studies of DFTR on analytical optimization problems can be found in [18].

We included the practical condition of stopping the algorithm when the radius falls below a tolerance Δ_{tol}. Since the convergence is asymptotic, theoretically this could stop the search far from an actual solution.

Another practical aspect that is worth mentioning regards the selection of the initial point. The initial point for many algorithms is often selected based on domain knowledge about the process. In well control optimization problems addressed in this work, we selected three initial points distributed in the feasible range for the well controls. As the DFTR algorithm is a local search method, it is expected to converge to a local optimum in the neighborhood of the starting point. This means that, in the first iterations up until $n + 1$ points, the models are computed by a coarse approximation, involving only part of the linear terms of the basis, and thus have a large sensitivity to the initial point.

The initial trust-region size is an important tuning parameter in this context as it limits the steps performed by the model initially, allowing the algorithm to adjust the trust-region size gradually until it starts using models with minimum Frobenius norm Hessian around the incumbent solution. Such models capture not only first-order information, but also some curvature in the approximation, and thus are a better representation of the function being optimized. Further, model-based derivative-free algorithms, such as the trust-region method presented in this work, can benefit from previous simulation runs by warm-starting the algorithm with a set of points. If the model is warm-started with a sufficient number of points to capture some curvature, the trust-region size could be chosen larger as the algorithm will use a more accurate model since the first iteration [18, Chapter 5].

4 Case studies

This section presents two case studies that compare the performance of the proposed method, i.e., the DFTR algorithm, against two direct-search methods, APPS [21] and PSO [28]. In both cases, the performance of the algorithms is investigated for well control optimization using the synthetic reservoir OLYMPUS model [15]. The first case study investigates the performance of the algorithms in a nominal well control optimization problem using a single realization of the OLYMPUS benchmark.

Algorithm 2 Derivative-free trust-region algorithm.

Inputs: initial point $\mathbf{x}_1$, variable bounds $\mathbf{x}_{\text{lb}}$, $\mathbf{x}_{\text{ub}}$.
Parameters: Thresholds for acceptance of steps $1 > \eta_1 > 0$, $\eta_0 \geq 0$, threshold for criticality step ϵ_c, initial radius Δ_1^{bct}, maximum radius allowed $\Delta_{\max}$.

Compute initial model m_1^{bct}.
for $k = 1, 2, \ldots$ **do**
  if $\sigma\left[m_k^{\text{bct}}, \mathbf{x}_k\right] < \epsilon_c$ **then**
    Perform criticality step (Algorithm 1) to compute new model m_k and trust-region radius Δ_k
  else
    Set $m_k = m_k^{\text{bct}}$, $\Delta_k = \Delta_k^{\text{bct}}$.
  end if
  if $\Delta_k < \Delta_{\text{tol}}$ **then**
    stop and return solution $\mathbf{x}_k$, with value $m_k(\mathbf{x}_k)$.
  end if
  Solve trust-region subproblem to compute the step $\mathbf{s}_k$:

$$\underset{s}{\text{maximize}} \quad m_k(\mathbf{x}_k + \mathbf{s})$$

subject to:

$$\|\mathbf{s}\|_\infty \leq \Delta_k$$
$$\mathbf{x}_{\text{lb}} \leq \mathbf{x}_k + \mathbf{s} \leq \mathbf{x}_{\text{ub}}$$

  Evaluate objective function at the trial point $\mathbf{x}_k^+ = \mathbf{x}_k + \mathbf{s}_k$ and compute

$$\rho_k = \frac{f(\mathbf{x}_k^+) - f(\mathbf{x}_k)}{m_k(\mathbf{x}_k^+) - m_k(\mathbf{x}_k)},$$

  if $\rho_k \geq \eta_1$ or ($\rho_k > \eta_0$ and the model is FL) **then**
    Set $\mathbf{x}_{k+1} = \mathbf{x}_k^+$.
    Include $\mathbf{x}_k^+$ in the interpolation set $\mathscr{Y}$, computing an updated model m_{k+1}^{bct}.
  else
    Set $\mathbf{x}_{k+1} = \mathbf{x}_k$.
    Try to include $\mathbf{x}_k^+$ in the interpolation set $\mathscr{Y}$ to generate an improved model m_{k+1}^{bct}.
    if $\mathbf{x}_k^+$ was not successfully included in $\mathscr{Y}$ **then**
      Compute another point to generate an improved model m_{k+1}^{bct}.
    end if
  end if
  if $\rho_k \geq \eta_1$ **then**
    Increase radius:

$$\Delta_{k+1}^{\text{bct}} = \min\left[\max(\Delta_k,\ 2\|\mathbf{s}_k\|),\ \Delta_{\max}\right].$$

  else if Iteration model was FL and (the criticality step was executed or no point was added) **then**
    Decrease radius

$$\Delta_{k+1}^{\text{bct}} = \frac{1}{2}\Delta_k.$$

  else
    Keep the same radius: $\Delta_{k+1}^{\text{bct}} = \Delta_k$.
  end if
end for

The second case study analyzes the performance of the algorithms in a robust well control optimization problem using a reduced-size ensemble consisting of the first 10 realizations of the OLYMPUS benchmark.

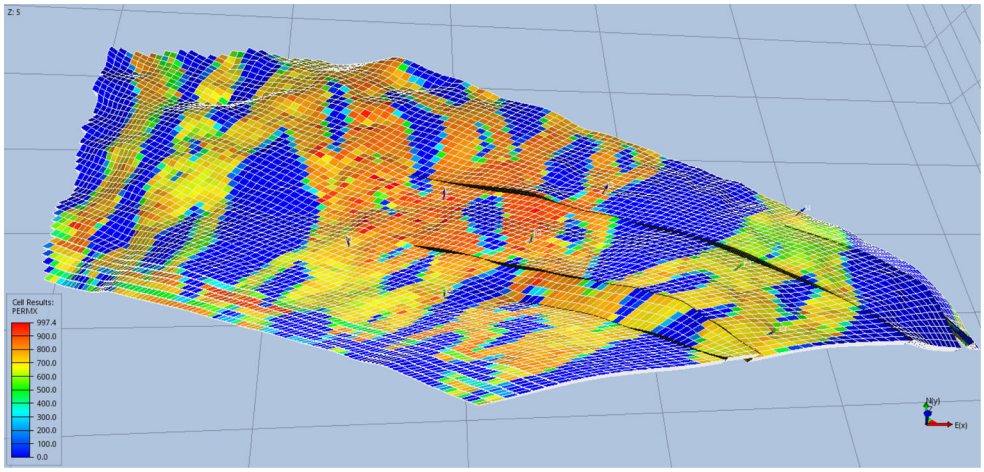
The optimization algorithms used in the case studies, i.e., APPS, PSO and DFTR, are all implemented in the FieldOpt framework [1] for field development optimization. All the algorithms have parallel implementation using the Message Passing Interface (MPI). However, due to its inherent sequential nature, the DFTR algorithm runs only one simulation at a time in the nominal well control problem investigated in case study 1. Therefore, in this case study, we consider both the objective evolution with respect to the number of simulations and per iteration in the performance comparison of the algorithms. In case study 2, the DFTR algorithm can take advantage of parallelization capabilities similarly to the other algorithms by simulating the ensemble of realizations in parallel at each iteration. Thus, in case study 2, we present the objective evolution only with respect to the number of iterations as all algorithms utilize the same computational resources.

The remaining of this section is divided in three parts. The first describes the numerical reservoir model utilized in the experiments. Then, we provide a description of the optimization problems being tackled as well as the results obtained for case study 1. The third and last part presents the optimization problem description and the results obtained for case study 2.

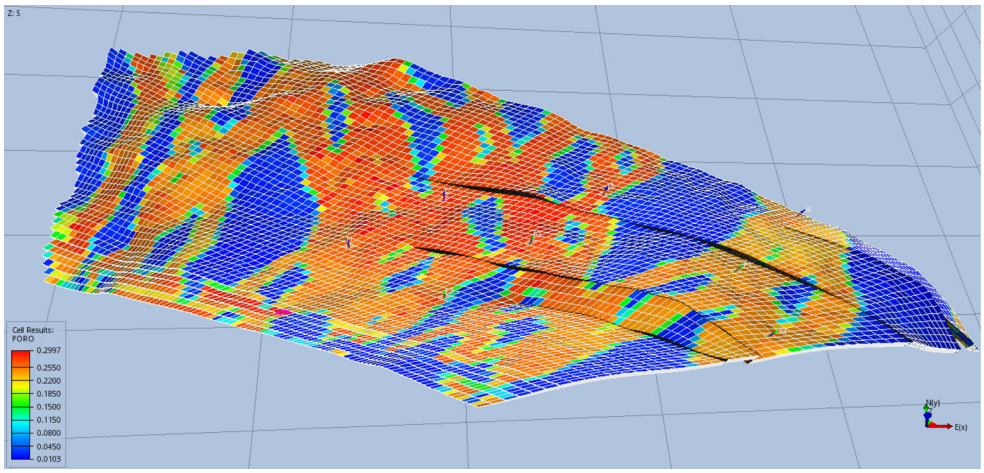
4.1 Reservoir model

The OLYMPUS reservoir model is a synthetic channelized reservoir inspired in the Norwegian Continental Shelf. It spans a 9 km by 3 km area, is 50 m thick and split into 16 layers. The model consists of grid cells of approximately 50 m × 50 m × 3 m each, resulting in a total of 192,750 active cells. Permeabilities in the X and Y coordinates are identical, whereas the permeability in the Z coordinate is 10% of the permeability in the X coordinate. The Oil-Water Contact (OWC) is set to 2090 m deep with a local hydrostatic pressure of 206 bar.

The OLYMPUS model was developed for benchmark studies in field development optimization. It consists of 50 geological realizations with different permeabilities, porosities and fault multipliers. In this work, we use the geological realization #1 for case study 1, and an ensemble with the realizations from #1 to #10 in case study 2. The horizontal permeability and porosity fields of the model used in case study 1 are depicted in Fig. 1. Differently from the original well control optimization challenge proposed by [15], the reservoir model has 8 wells, of which 4 are producers and 4 are injectors. The injectors



(a) Horizontal permeability field



(b) Porosity field

Fig. 1 Static properties of the OLYMPUS reservoir model, realization #1

are placed in fault-divided regions of the reservoir, as it can be seen in Fig. 2b. The model has pressure support from an aquifer at the west, as shown in the saturation field in Fig. 2a. All wells perforate vertically the grid blocks from the first to the fourth layer, and are placed within channelized regions of the reservoir. The injectors are strategically placed in locations surrounding larger oil-containing volumes.

The horizontal permeability fields of the realizations #1 to #10 are shown in Fig. 3. There are large contrasts in permeability and porosity between the channels and outer surrounding volumes. Notice that each geological realization has different settings for the channels, and therefore the sweep occurs differently in each model. Because of such differences between the models, a control strategy that yields good performance in one model might result in early water breakthrough in another realization.

4.2 Case Study 1: Nominal well control optimization

The problem formulated in Eqs. 2a–2e is instantiated for the nominal well control optimization of the realization #1 of the reservoir model described previously. In the following we present the parameters for the optimization algorithms, the NPV parameters and the initial well controls used as starting points for the algorithms.

4.2.1 Problem setup

The total simulation time is set to $T = 20$ years, which is split into $N = 5$ control steps of 4 years each. The wells controls $\mathbf{u}$ are constant bottom-hole pressures for each control step $k \in \mathscr{K} := \{1, \ldots, N_t\}$, thus the total number of optimization variables is calculated as the Cartesian product of the number of wells $N_w = 8$ and the number of control steps $N_t = 5$, i.e. $N_{\mathbf{u}} := N_w \times N_t = 40$.

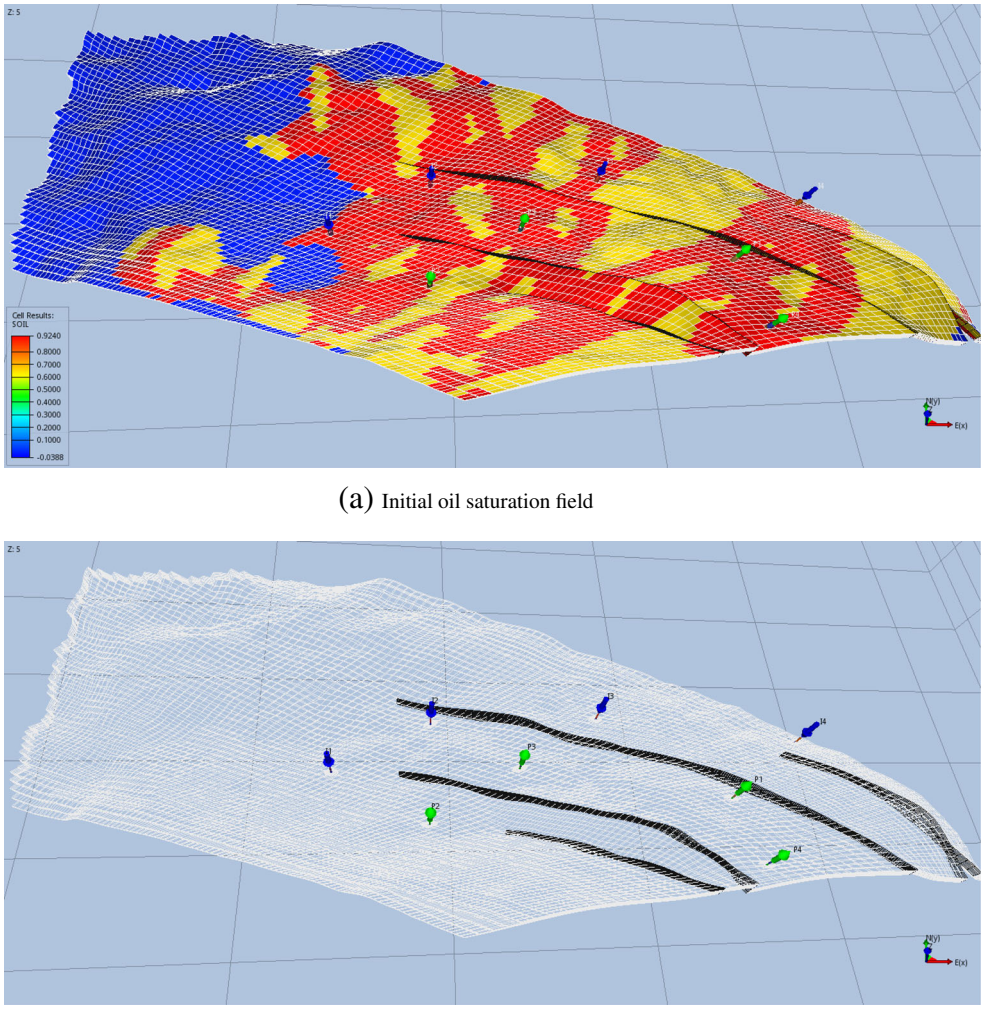


Fig. 2 Initial saturation field, and well locations

(a) Initial oil saturation field

(b) Well locations

The objective function is the NPV as defined in Eq. 2a. The economic parameters of the NPV are defined in Table 1.

Since an economic analysis is not the focus of the studies we consider constant costs for the entire field life cycle. The time intervals Δt_i are set to calendar months. Because all wells are assumed to be drilled and completed at the start time of the field life cycle, the drilling costs are assumed to be negligible. Objective function evaluations are calculated by running reservoir simulations using a commercial fully implicit black oil simulator [17].

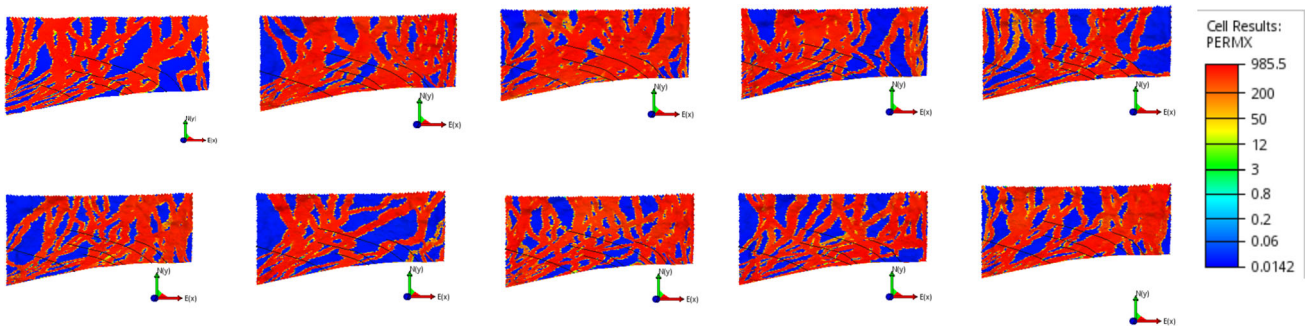


Fig. 3 Horizontal permeability map of the ensemble used in case study 2 (from the upper left realization #1 to the lower right realization #10). The channelized regions represent highly permeable zones of the reservoir, whereas the surrounding volumes are low permeability regions

Table 1 NPV economic parameters

Oil price	r_{op}^t	450 \$/Sm 3
Water injection cost	r_{wp}^t	12.5 \$/Sm 3
Water production cost	r_{wi}^t	12.5 \$/Sm 3
Annual discount factor	d	10 %
Normalization term	τ	365 days
End of the life cycle period	T	20 years

Table 3 Initial controls

Starting point	BHP of producers	BHP of injectors
P1	120	250
P2	145	300
P3	170	350

The bottom-hole pressure (BHP) of the wells needs to range within certain bounds. BHPs for the producers are set to be between 80 bars and 210 bars, while injectors are set to operate within the range from 200 to 400 bars. For simplicity, in this work, we impose no rate constraints on the production and injection rates of the wells.

Different initial parameters are selected for the algorithms as shown in Table 2. By varying the initial parameters we investigate the sensitivity of the algorithms with respect to their tuning parameters.

Except for PSO, which starts with a random set of particles, the performance and local solution obtained by the other algorithms can vary with the starting point. Thus, we chose three different initial well controls as starting points, as shown in Table 3.

4.2.2 Simulation results

In this section we present the objective function evolution presented by the algorithms Asynchronous Parallel Pattern Search (APPS), Particle Swarm Optimization (PSO), and the Derivative-Free Trust-Region (DFTR) with different parameters using initial points P1, P2 and P3. The direct-search algorithms, i.e. APPS and PSO, executed in parallel mode over 8 cores, whereas the DFTR algorithm executed sequentially.

Figure 4 shows the NPV evolution of the algorithms starting from P1. The NPV evolution by the total number of simulations is depicted for a total of 1000 simulations. We are considering that simulations are computationally expensive and take long time. Plotting by number of simulations we can show the ascent obtained with the use of this costly resource. This would not be shown in plots by number of algorithmic iterations. Further, we believe showing the NPV evolution by the number of simulations is a fairer comparison of the algorithms performance as it shows their efficiency with respect to the required computational budget.

In this plot, it can be seen that the DFTR algorithm clearly outperforms both PSO and APPS across all the different algorithm parameters. It not only achieves a solution with the best NPV values at the end, but also achieves a solution near the optimum after a few iterations, namely with less than 100 simulations. Although the convergence of PSO is slower than the DFTR algorithm, it achieves a final solution at the end that is not far from the solution obtained by the DFTR algorithm. APPS is outperformed by both PSO and the DFTR algorithm.

The performance of the algorithms starting from point P2 is shown in Fig. 5. This figure shows that APPS is outperformed by both PSO and the DFTR algorithm in all cases. Two out of three final objective function values obtained using DFTR are close to the ones obtained with PSO, however, for the same number of simulations, DFTR shows equal or better progression early on in the optimization process. One plausible explanation for this is the fact that the DFTR algorithm got trapped into a local minimum when performing the search, whereas PSO found improved solutions because of the stochasticity of its search.

The last performance comparison is performed with the three algorithms starting from P3, as shown in Fig. 6. When the algorithms are compared with respect to their evolution by the number of simulations, DFTR reaches good candidate solutions with only a few simulations, and in most cases converges to solutions with slightly higher objectives than the ones obtained with PSO. APPS is considerably slower than both PSO and the DFTR in all cases.

4.2.3 Field production analysis

In this section we analyze the various well control solutions with respect to field production performance.

Figure 7 shows the NPV, and the corresponding field cumulative profiles for the best runs obtained using APPS, PSO and DFTR. The Field Oil Production Total (FOPT) evolution over the years for the three algorithms is shown in Fig. 7(a). The FOPT in the first three years is similar for all algorithms. From the fourth year on, the FOPT obtained with APPS distances itself from the production obtained with PSO and DFTR. The FOPT obtained with the latter

Table 2 Parameters of optimization algorithms

Algorithm	Parameter	Values
APPS	Initial step size	50, 75, 150
Trust-Region	Initial radius	50, 75, 150
PSO	Swarm size	20, 25, 40

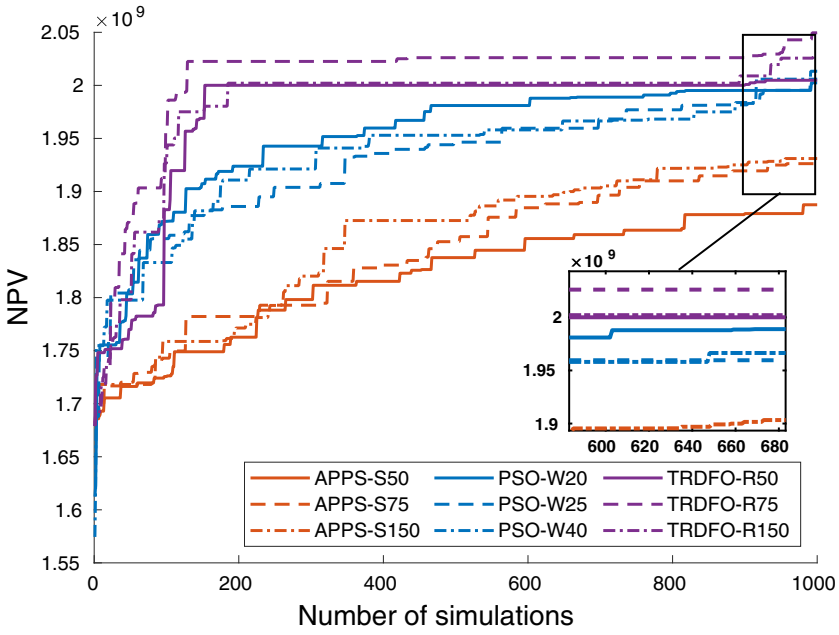


Fig. 4 Comparison of performance of the algorithms with the initial point P1

methods progresses at a similar pace, whereas the larger FOPT is obtained with PSO at the end of the field life cycle.

PSO achieved a solution that yielded the highest accumulated oil production at the end of the field life cycle. On the other hand, when it comes to the Field Water Production Total (FWPT), as shown in Fig. 7(c), PSO's best solution is the one that produced the most water in total at the end of the field life cycle. APPS' solution has the largest FWPT during the first 10 years, when it is surpassed by PSO's solution. The solution obtained with DFTR is the one that produces the least water during most of the field life cycle, and in cumulative terms at the end.

Figure 7(d) shows the Field Water Injection Total (FWIT) evolution over the years for all the algorithms. PSO yields a solution which injects the most water during the whole field life cycle. APPS on the other hand yields a solution that injects the least water during most of the life cycle period. DFTR's solution achieves the least FWIT at the end.

Among all the optimization runs, PSO yields the solution with higher NPV, as shown in Fig. 7(b). The final NPV achieved using DFTR is not far from this result. DFTR's NPV evolution is somewhat comparable to PSO's, and was practically the same over various time periods, namely in the first 8 years and also close to the end of the field life cycle,

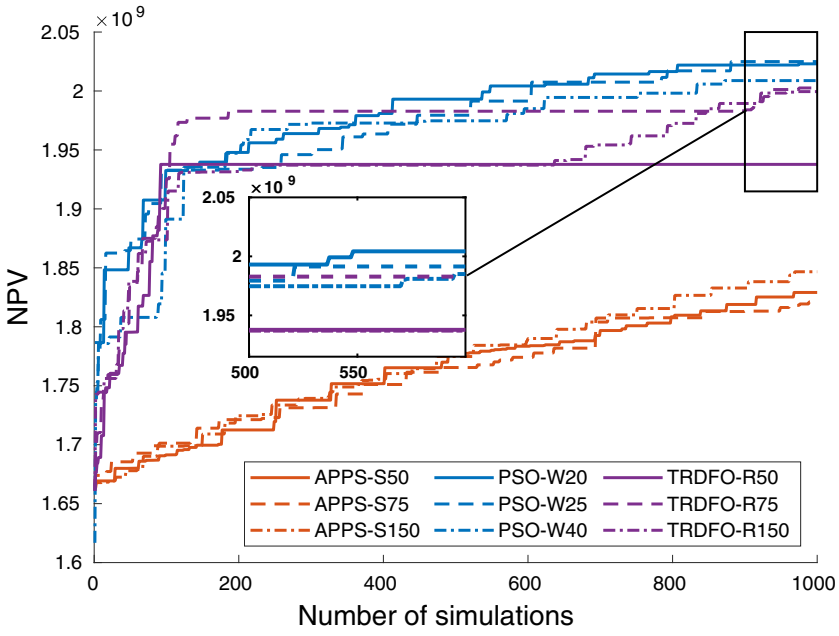


Fig. 5 Comparison of performance of the algorithms with the initial point P2

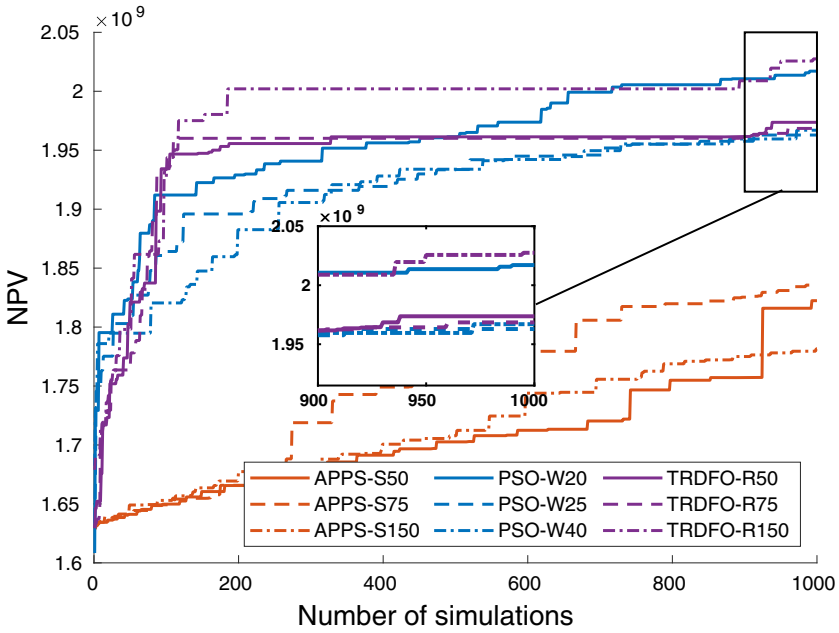


Fig. 6 Comparison of performance of the algorithms with the initial point P3

from 2032 to 2034. Recall that PSO's ability to achieve an improved solution comes at the expense of performing numerous reservoir simulations, namely 10 times more compared to DFTR and APPS in this particular comparison. Meanwhile, the DFTR achieved a fairly good solution within a few function evaluations performed sequentially. The NPV achieved by APPS is considerably lower than the others from the fourth year on of field activity.

For completeness, we present the optimal well controls of producers and injectors in Fig. 8. Notice that the control strategies calculated by each algorithm are quite different (see for instance the controls of the injection well i4 at the lower right plot in Fig. 8). With APPS the bottom-hole pressure of the well is kept constant at a low value from the fourth year until near to the end of the field life cycle, whereas the well controls calculated by PSO and DFTR switch often during the same time period.

4.3 Case study 2: Robust well control optimization

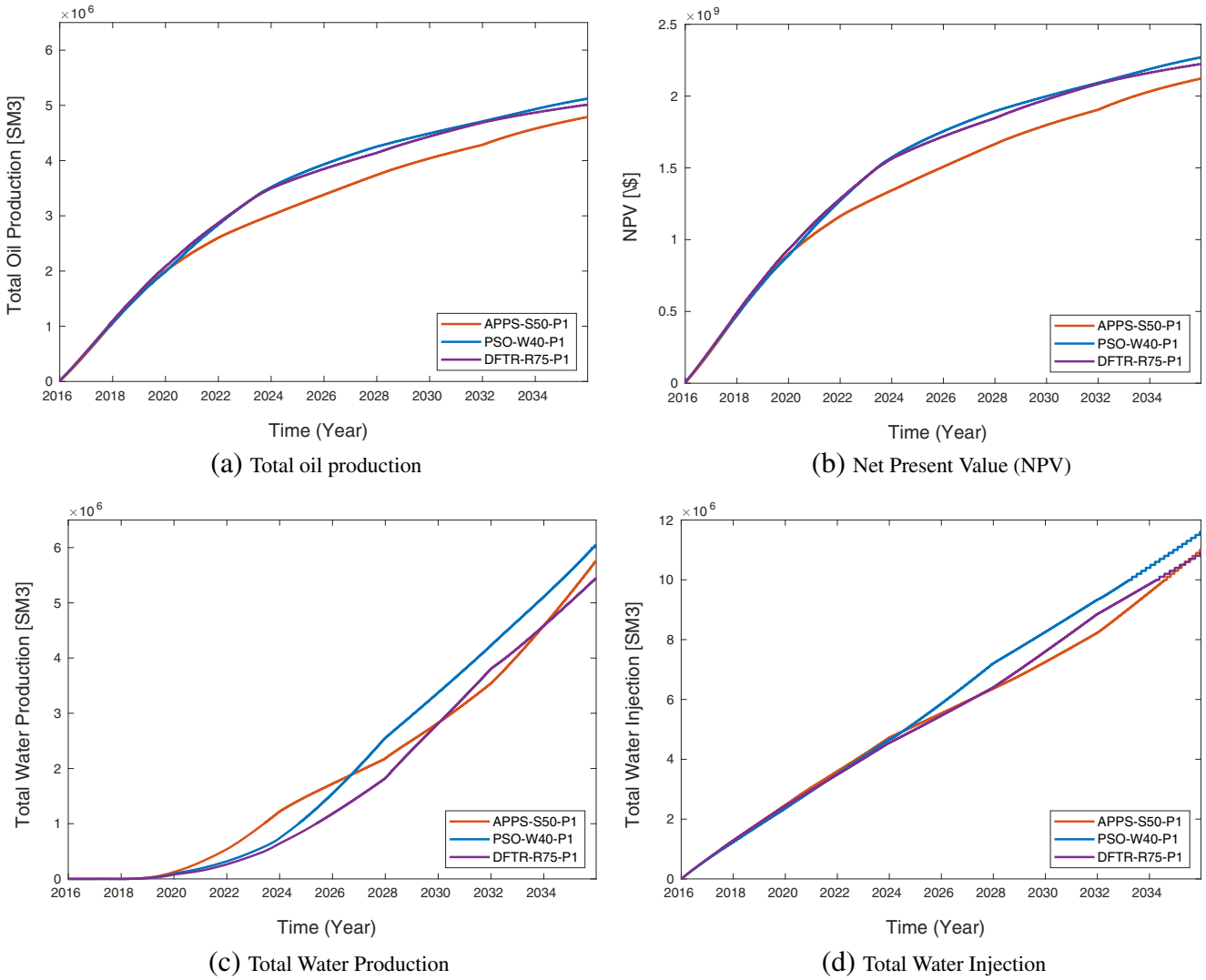
In the second case study we analyze the performance of the algorithms DFTR, APPS and PSO in the robust well control optimization problem formulated in Eqs. 4a–4e. The optimization is applied to the Olympus ensemble described in Fig. 3 which consists of the first 10 realizations of the original Olympus benchmark.

4.3.1 Problem setup

The wells are located in the same positions as the nominal case in all the ensemble members as illustrated in Fig. 2b. The simulation schedule is also kept the same, i.e., a total simulation time of $T = 20$ years, split into $N = 5$ control steps of 4 years. Although the performance of a given well control strategy might vary from one realization to the other, the well controls are the same for all realizations independently of the model to which it is applied. Therefore, the number of optimization variables in the robust problem remains the same of the nominal problem, i.e., it is the number of wells times the number of control steps $N_{\mathbf{u}} := N_{\mathrm{w}} \times N_{\mathrm{t}} = 40$.

The objective is the cost function formulated in Eq. 3 and the economic parameters utilized in each term of the equation are defined in Table 1. Objective function evaluations are calculated by running reservoir simulations of all equally probable ensemble realizations with a certain well control strategy and taking the average NPV. In the case of inherently sequential algorithms like the DFTR, these reservoir simulations can be executed in parallel, thereby leveraging its computational efficiency with respect to other naturally parallel algorithms such as APPS and PSO.

Analogously to the nominal problem, the decision variables in the robust optimization problem are the well controls. The wells are controlled through piecewise-constant values for the bottom-hole pressures, which are bounded in the interval between 80 bars and 210 bars for producers, and range from 200 to 400 bars for injectors. Except for PSO, which starts with a random set of particles, both DFTR and APPS start from the initial point P2 from Table 3, as the bottom-hole pressures of the wells in this initial point lie at the center of the feasible pressure intervals. The initial parameters of the algorithms are set



(a) Total oil production (b) Net Present Value (NPV)

(c) Total Water Production (d) Total Water Injection

Fig. 7 Field rates and NPV obtained with the best optimization runs of APPS, PSO, and DFTR

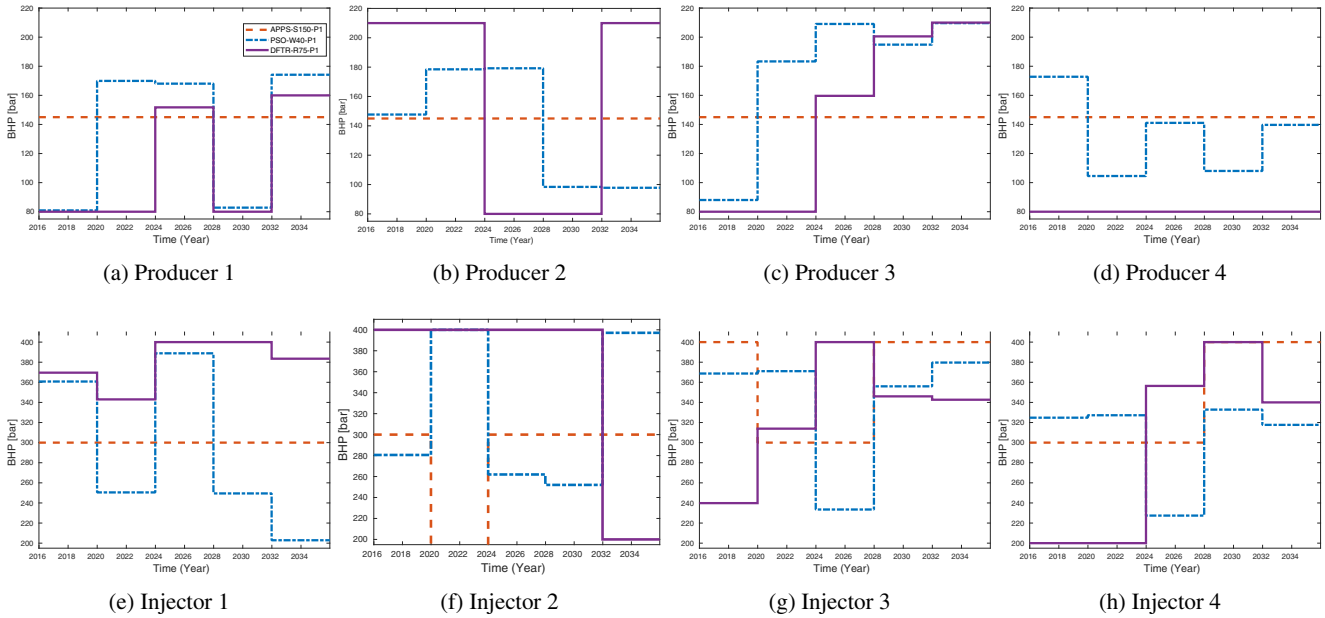
to the values which yielded the best performance in most nominal well control problems, i.e., APPS with a initial step size of 150, DFTR with an initial radius of 75, and PSO with a swarm size of 20.

4.3.2 Simulation results

In this section we present the simulation results obtained with the algorithms in the robust well control optimization problem described in the previous section. As the DFTR algorithm performs each iteration sequentially, and in each iteration a batch of simulations for the ensemble members can be performed in parallel, we chose to execute all algorithms in parallel over 10 cores, which is exactly the number of realizations in the ensemble. This allows all algorithms to use the computational budget equally. All algorithms executed in parallel over a total of 10 cores (same number of ensemble members) with a computational budget of 1000 simulations.

The objective function evolution (expected NPV) per iteration for all the algorithms is presented in Fig. 9. In this context, for a fair comparison of the algorithms, we consider one iteration to be the simulation of all realizations in the ensemble for a certain candidate solution within a robust well control optimization procedure.

In this plot, it can be seen that the DFTR algorithm clearly outperforms both PSO and APPS. It not only achieves the best NPV at the end, but also presents improved convergence performance, being able to obtain a solution with higher NPV than the best solutions obtained with APPS and PSO within fewer than 30 ensemble simulations. PSO starts from a solution with higher NPV due to its stochastic nature, but it is surpassed by DFTR nearly over 20 ensemble simulations. APPS showed a performance



(a) Producer 1 (b) Producer 2 (c) Producer 3 (d) Producer 4

(e) Injector 1 (f) Injector 2 (g) Injector 3 (h) Injector 4

Fig. 8 Optimal well controls for producers and injectors corresponding to best runs of APPS, PSO and DFTR. The best runs of each algorithm are identified with APPS-S150-P1, PSO-W40-P1, and DFTR-R75-P1 in the simulation analysis. Orange dashed lines are the controls obtained with APPS, blue dotted lines are the PSO controls, and solid purple lines are the controls with DFTR

considerably inferior to both DFTR and PSO from the first iteration until the end of the execution.

Another interesting behavior that can be observed in Fig. 9 is the search pattern presented by the algorithms. PSO exploits the feasible space stochastically through the spread of particles spanning a large search area (blue circles). On the other hand, APPS and DFTR conduct a more localized search, exploring the space in surrounding regions of trial solutions. Among the algorithms, DFTR presented much faster convergence arguably because of the model support in the search for new trial solutions. The model, i.e. the trust-region, provides additional information with respect to the curvature of the response surface in the neighborhood of trial solutions. This helps the algorithm in identifying ascent regions in the feasible space, which lead to larger improvements in the objective per iteration.

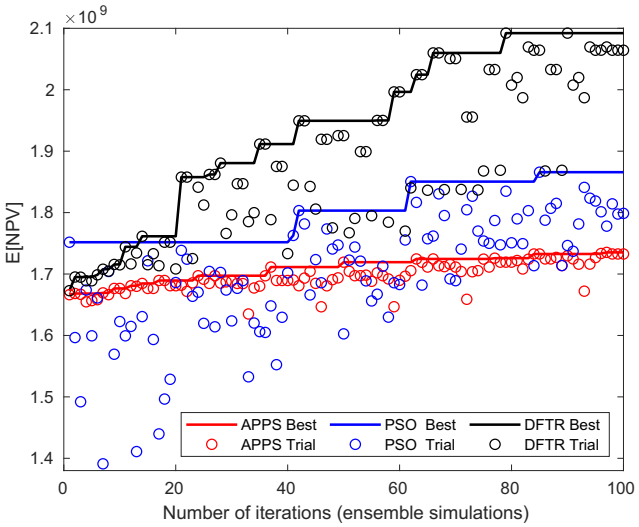


Fig. 9 Performance comparison of the algorithms in the robust well control optimization problem. The figure shows the trial (circles) and best solutions (solid line) found by the algorithms by the number of iterations, where an iteration corresponds to the execution of one batch of simulations of all ensemble members. Each circle in the plot represents one candidate solution by the corresponding algorithm, whereas the solid lines represent the best solution found by the algorithm among the cases evaluated

Good convergence properties are particularly relevant when optimizing the performance of high-fidelity models of real-world fields. In such scenarios, each cost function evaluation requires considerable amounts of computational resources as it requires the simulation of a set of fully-fledged models corresponding to the ensemble of possible realizations of the real reservoir. Because of the curse of dimensionality in these applications, even with a large computational budget, it is wise to have informed decisions on where to search for potential solutions rather than exploiting randomly the entire feasible space.

4.3.3 Field production analysis

In this section we present the field results corresponding to the best solutions found by APPS, PSO, and DFTR in the robust well control optimization problem tackled in this case study. Figure 10 presents the mean values and the standard deviations for the field rates and the

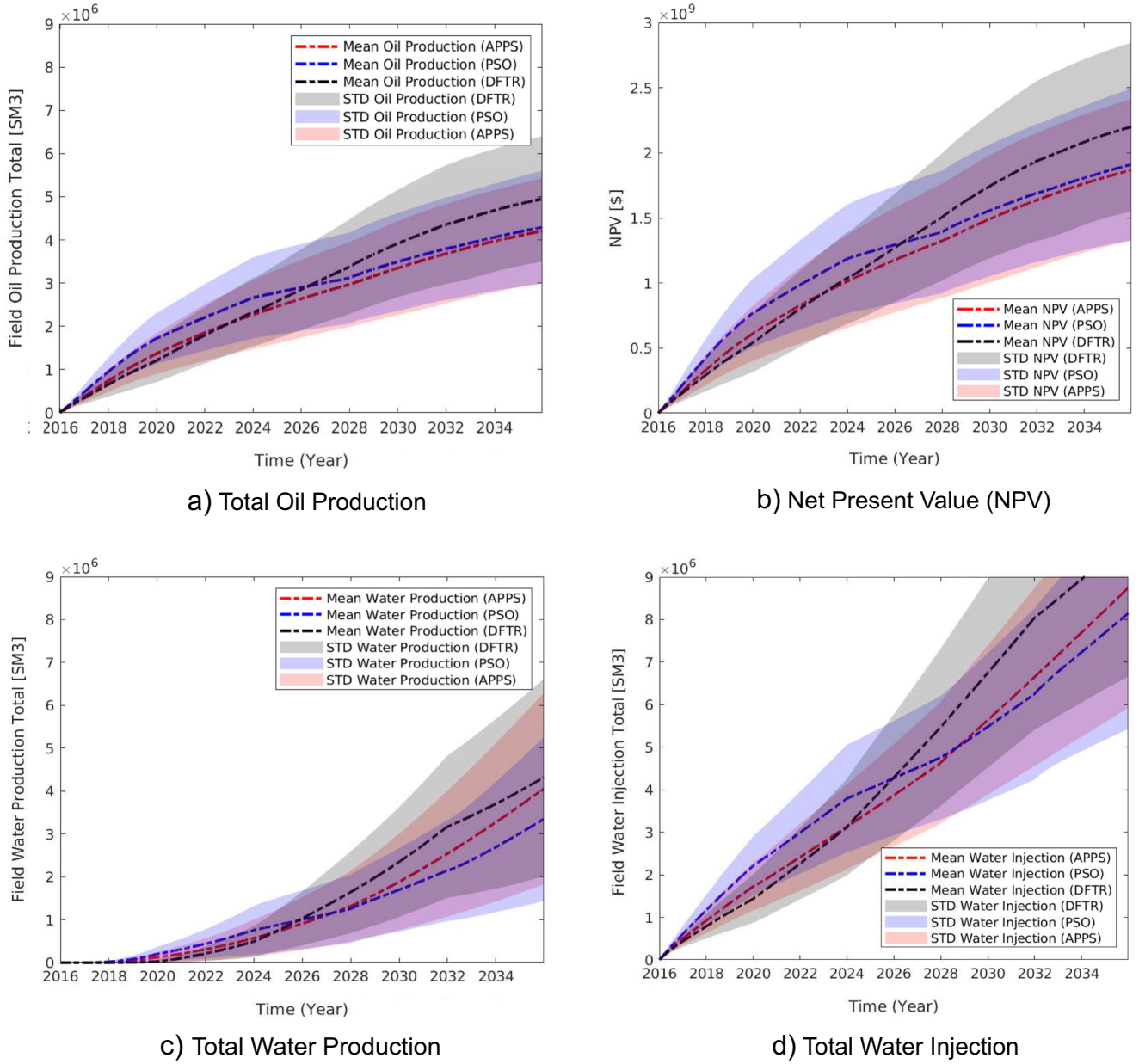


Fig. 10 Total oil production, water production, water injection, and NPV obtained with the best solutions found by APPS, PSO, and DFTR. Red dashed lines and red shaded regions are the mean and standard deviation values obtained with APPS. Blue dashed lines and blue shaded regions are the mean values and corresponding standard deviation obtained with PSO. Black dashed lines and grey shaded regions are the mean and standard deviation values obtained with DFTR

net present values corresponding to the best solutions found by the algorithms. From the plots, it is possible to conclude that the total oil production obtained by the DFTR solution is the highest among the algorithms, whereas PSO and APPS yield similar production rates with a slightly higher production from PSO. On the other hand, the DFTR solution also achieves the highest total water production and injection, while the PSO solution achieves lowest production and injection of water. The water production and injection obtained with APPS is average compared to DFTR and PSO solutions. Even with a control strategy that produces and injects more water, DFTR achieves a considerably higher NPV compared to PSO and APPS, mainly because of the considerably higher oil production.

The best well control strategies for producers and injectors obtained with the APPS, PSO and DFTR, corresponding to the field results shown Fig. 10, are depicted in Fig. 11. As it can be seen in the figure, the optimal control strategies computed by each algorithm

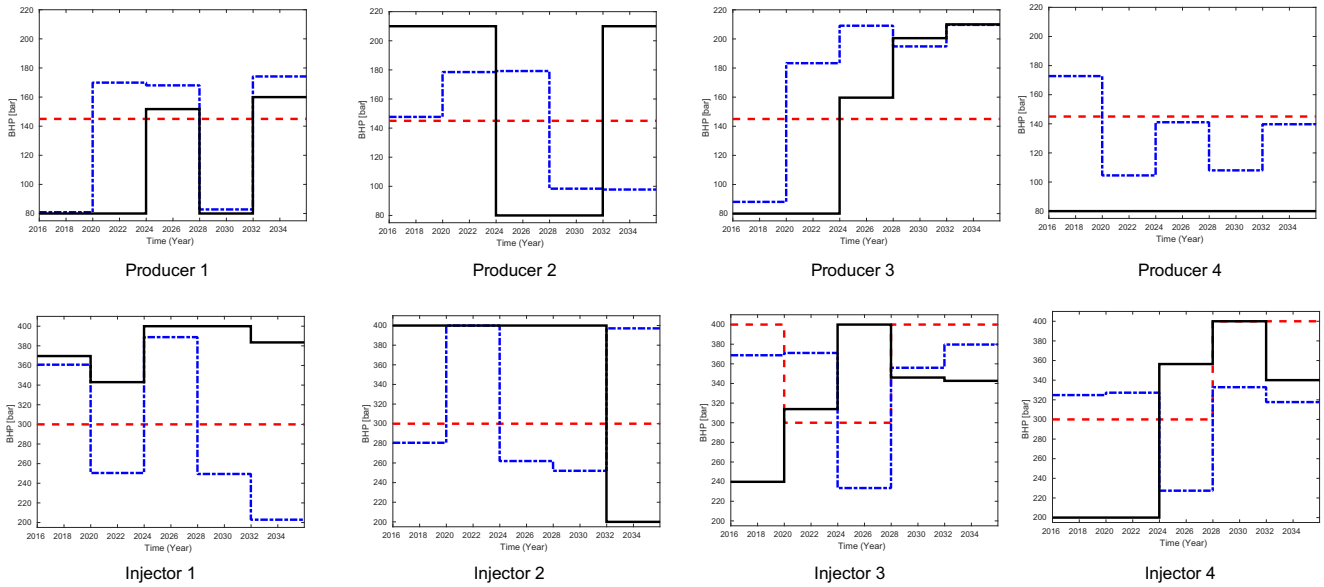


Fig. 11 Optimal well controls for the robust optimization problem. Red dashed lines are the controls obtained with APPS, blue dotted lines are the PSO controls, and solid black line are the controls with DFTR

follow similar trends in some cases (see for instance the production well p1) but can also be quite different (see injection well i3). In the case of the producer p1, the controls are switched between an average and low value of bottom-hole pressure at similar times, whereas with APPS the controls are kept constant at the average until the end of the field life cycle. On the other hand, in the case of the injector i3, the controls suggested by the optimizers switch quite often in different time periods and sometimes in opposite directions, e.g., around the eighth year, DFTR increases the control to a high value while PSO switches the control to a low value at the same time.

5 Conclusions

This work proposes a derivative-free trust-region optimization algorithm for well control optimization. The algorithm is assessed in a synthetic field developed for benchmarking field development optimization methodologies. DFTR performance is compared against two other derivative-free methods, APPS and PSO. Current results show DFTR has promising convergence properties compared to the other derivative-free methods. In particular, the method presented improved convergence properties, being capable to reach fairly good solutions requiring comparatively fewer iterations. This feature can be particularly attractive for practitioners who seek ways to improve production strategies while using fully-fledged models, which can be quite demanding.

In the robust well control optimization problem, the competitiveness of the algorithm is much improved against parallel methods because of its ability to perform a batch of ensemble simulations in parallel at each iteration. In a case study involving an ensemble of realistic models, the proposed method presented superior convergence performance compared to the other algorithms, being able of achieving a final solution with considerably higher expected NPV. Good convergence is particularly important when optimizing the performance of an ensemble of high-fidelity models of real-world fields. In such scenarios, each cost function evaluation requires considerable amounts of computational resources to simulate the ensemble of models at each iteration. In such cases, even with a generous computational budget, it is highly beneficial to utilize an algorithm such as the DFTR that can smartly search for candidate solutions rather than randomly exploiting the feasible space.

Notice that we do not claim that the proposed algorithm is superior to the existing derivative-free optimization methods available in the literature. In fact, a comprehensive analysis of the existing related methods applied to well control and well placement problems is a relevant topic for future research. Instead, our work focuses on the description of a novel derivative-free trust-region algorithm for robust well control optimization, which to the best of our knowledge, is a new type of algorithm for such problems. From the results obtained in representative well control optimization problems, we conclude that the DFTR method offers practitioners an alternative approach when derivatives

are not readily available and the computational budget is limited.

We acknowledge that problem size may be the most important limitation of any derivative-free optimization method, including the DFTR algorithm. The literature unanimously advises using gradient based methods whenever possible, even if this costs more hours in the implementation side. For instance, employing the DFTR algorithm to solve the original Olympus case with $n = 1\,440$ variables could require more than a 1 million simulations (estimate based on the number of terms of the basis, $L = (n+1)(n = 2)/2$). Still, even though the main advantage of the DFTR algorithm is that it enables search based on descent information from underdetermined models, and thus the algorithm requires far less cost function evaluations to advance, solving for problems with thousands of variables still requires a large number of simulations just to obtain linear descent information and is a major challenge and current research topic. In fact, [10] further describes the general DFTR methodology as practical for problems with number of variables in the range 50–100. Thus, as a first application of the DFTR algorithm to a realistic case, this work has tested the algorithm using the Olympus model using a control optimization problem definition involving fewer variables than those posed in the original challenge.

Future work involves extending the proposed methodology in terms of constraint-handling. As the trust-region method builds a model of the objective function based on sampled points, this capability can be extended to also treat output constraints, which are typically difficult to deal with for both direct-search and gradient-based methods.

Another potential research topic involves conducting a more extensive comparison of the proposed method with other types of algorithms. In this work, we assume gradients can not be relied or are unavailable due to the lack of sensitivities provided by the simulator. Therefore, we only compare the proposed method with other types of derivative-free methods which are typically employed in well control optimization problems. On the other hand, further comparison of the proposed method with adjoint and other gradient-based methods, e.g. SQP, could demonstrate the robustness of the derivative-free trust-region method in the presence of discontinuities in the nonlinear variable profiles. Further, an extended comparison with other derivative-free algorithms, e.g. experimental design with response surfaces and meta model assisted memetic algorithms, could provide readers with a good overview of the existing derivative-free algorithms for production optimization. Though a more extensive computational analysis involving several algorithms is an interesting line of study, such study would require substantial standalone effort and thus is left for future research.

Acknowledgements This research is a part of BRU21 – NTNU Research and Innovation Program on Digital and Automation Solutions for the Oil and Gas Industry (www.ntnu.edu/bru21). The authors acknowledge funding from the Research Council of Norway (INTPART – Brazil-Norway Production Optimization Consortium - Phase 2, project number 286801).

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital).

References

1. Baumann, E.J., Dale, S.I., Bellout, M.C.: FieldOpt: A powerful and effective programming framework tailored for field development optimization. Comput. Geosci. **135**(104), 379 (2020)
2. Brouwer, D.R., Jansen, J.D.: Dynamic optimization of waterflooding with smart wells using optimal control theory. SPE J. **9**(04), 391–402 (2004)
3. Bukshtynov, V., Volkov, O., Durlofsky, L.J., Aziz, K.: Comprehensive framework for gradient-based optimization in closed-loop reservoir management. Comput. Geosci. **19**(4), 877–897 (2015)
4. Capolei, A., Suwartadi, E., Foss, B., Jørgensen, J.B.: Waterflooding optimization in uncertain geological scenarios. Comput. Geosci. **17**(6), 991–1013 (2013)
5. Chen, C., Li, G., Reynolds, A.: Robust constrained optimization of short- and long-term net present value for closed-loop reservoir management. SPE J. **17**(3) (2012)
6. Codas, A., Foss, B., Camponogara, E.: Output-constraint handling and parallelization for oil-reservoir control optimization by means of multiple shooting. SPE J. **20**(4), 856–871 (2015)
7. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. Society for Industrial and Applied Mathematics (SIAM) and Mathematical Programming Society (MPS) (2000)
8. Conn, A.R., Scheinberg, K., Toint, P.L.: Recent progress in unconstrained nonlinear optimization without derivatives. Math. Program. **79**(1-3), 397–414 (1997)
9. Conn, A.R., Scheinberg, K., Vicente, L.N.: Geometry of interpolation sets in derivative free optimization. Math. Program. **111**(1-2), 141–172 (2008)
10. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. Society for Industrial and Applied Mathematics (SIAM) and Mathematical Programming Society (MPS) (2009)
11. Datta-Gupta, A., Alhuthali, A.H.H., Yuen, B., Fontanilla, J.: Field applications of waterflood optimization via optimal rate control with smart wells. SPE Reserv. Eval. Eng. **13**(03), 406–422 (2010)

12. Dehdari, V., Oliver, D.S.: Sequential quadratic programming for solving constrained production optimization–case study from brugge field. SPE J. **17**(03), 874–884 (2012)
13. Echeverría Ciaurri, D., Isebor, O., Durlofsky, L.: Application of derivative-free methodologies to generally constrained oil production optimization problems. Procedia Comput. Sci. **1**(1), 1301–1310 (2010)
14. van Essen, G., Van den Hof, P., Jansen, J.D.: Hierarchical long-term and short-term production optimization. SPE J. **16**(01), 191–199 (2011)
15. Fonseca, R., Rossa, E.D., Emerick, A., Hanea, R., Jansen, J.: Overview of the olympus field development optimization challenge. In: ECMOR XVI - 16th European Conference on the Mathematics of Oil Recovery, EAGE, pp. 1–10 (2018)
16. G., K.T., Michael, L.R., Virginia, T.: Optimization by direct search: New perspectives on some classical and modern methods. SIAM Rev. **45**(3), 385–482 (2003)
17. GeoQuest, S.: Eclipse reservoir simulator. Man. Tech. Descr. Houston TX (2014)
18. Giuliani, C.M.: Contributions to derivative-free optimization: an exact-penalty method and decompositions for distributed control. Ph.D. thesis, Universidade Federal de Santa Catarina. https://bu.ufsc.br/teses/PEAS0331-T.pdf (2019)
19. Giuliani, C.M., Camponogara, E., Conn, A.R.: A derivative-free exact penalty algorithm: Basic ideas, convergence theory and computational studies. To appear in Computational and Applied Mathematics (2021)
20. Hasan, A., Gunnerud, V., Foss, B., Teixeira, A.F., Krogstad, S.: Decision analysis for long-term and short-term production optimization applied to the voador field. In: Proc. of SPE Reservoir Characterization and Simulation Conference and Exhibition, pp. 16–18 (2013). Society of Petroleum Engineers
21. Hough, P.D., Kolda, T.G., Torczon, V.J.: Asynchronous parallel pattern search for nonlinear optimization. SIAM J. Sci. Comput. **23**(1), 134–156 (2001)
22. Isebor, O.J., Echeverría Ciaurri, D., Durlofsky, L.J.: Generalized field-development optimization with derivative-free procedures. SPE J. **19**(5) (2014)
23. Jansen, J.D.: Adjoint-based optimization of multi-phase flow through porous media - a review. Comput. Fluids **46**(1), 40–51 (2011)
24. Jansen, J.D., Brouwer, R., Douma, S.G.: Closed loop reservoir management. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers (2009)
25. Kourounis, D., Durlofsky, L.J., Jansen, J.D., Aziz, K.: Adjoint formulation and constraint handling for gradient-based optimization of compositional reservoir flow. Comput. Geosci. **18**(2), 117–137 (2014)
26. Kraaijevanger, J.F.B.M., Egberts, P.J.P., Valstar, J.R., Buurman, H.W.: Optimal waterflood design using the adjoint method. SPE Reservoir Simulation Symposium p. 15. SPE-105764-MS (2007)
27. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006). https://doi.org/10.1007/978-0-387-40065-5
28. Nwankwor, E., Nagar, A.K., Reid, D.: Hybrid differential evolution and particle swarm optimization for optimal well placement. Comput. Geosci. **17**(2), 249–268 (2013)
29. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Glob. Optim. **1**(1), 15–22 (1991)
30. Powell, M.J.D.: Least frobenius norm updating of quadratic models that satisfy interpolation conditions. Math. Program. **100**(1), 183–215 (2004)
31. Sampaio, P.R., Toint, P.L.: Numerical experience with a derivative-free trust-funnel method for nonlinear optimization problems with general nonlinear constraints. Optim. Methods Softw. **31**(3), 511–534 (2016)
32. Sarma, P., Durlofsky, L.J., Aziz, K., Chen, W.H.: Efficient real-time reservoir management using adjoint-based optimal control and model updating. Comput. Geosci. **10**(1), 3–36 (2006)
33. Scheinberg, K., Toint, P.L.: Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization. SIAM J. Optim. **20**(6), 3512–3532 (2010)
34. Silva, T.L., Codas, A., Stanko, M., Camponogara, E., Foss, B., et al: Network-constrained production optimization by means of multiple shooting. SPE Reserv. Eval. Eng. **22**(2), 709–733 (2019)
35. Suwartadi, E., Krogstad, S., Foss, B.: Nonlinear output constraints handling for production optimization of oil reservoirs. Comput. Geosci. **16**(2), 499–517 (2011)
36. Volkov, O., Bellout, M.C.: Gradient-based production optimization with simulation-based economic constraints. Comput. Geosci. **21**(5), 1385–1402 (2017)
37. Volkov, O., Voskov, D.V.: Effect of time stepping strategy on adjoint-based production optimization. Comput. Geosci. **20**(3), 707–722 (2016)
38. Wang, C., Li, G., Reynolds, A.C.: Production optimization in closed-loop reservoir management. SPE J. **14**(3), 506–523 (2010)
39. Yan, X., Reynolds, A.C.: Optimization algorithms based on combining FD approximations and stochastic gradients compared with methods based only on a stochastic gradient. SPE J. **19**(5) (2014)

Affiliations

Thiago L. Silva[1,2] · Mathias C. Bellout[1] · Caio Giuliani[3] · Eduardo Camponogara[3] · Alexey Pavlov[1]

Mathias C. Bellout
mathias.bellout@ntnu.no

Caio Giuliani
caio.giuliani@gmail.com

Eduardo Camponogara
eduardo.camponogara@ufsc.br

Alexey Pavlov
alexey.pavlov@ntnu.no

1 Department of Geoscience and Petroleum, NTNU, Trondheim, Norway

2 Department of Sustainable Energy Technology, SINTEF Industry, Trondheim, Norway

3 Department of Automation and Systems Engineering, UFSC, Florianópolis, Brazil