



Derivative-free search approaches for optimization of well inflow control valves and controls

Mathias C. Bellout¹ · Thiago L. Silva^{1,2} · Jan Øystein Haavig Bakke³ · Carl Fredrik Berg¹

Received: 27 April 2023 / Accepted: 3 January 2024
© The Author(s) 2024

Abstract

Decisions regarding problem conceptualization, search approach, and how best to parametrize optimization methods for practical application are key to successful implementation of optimization approaches within georesources field development projects. This work provides decision support regarding the application of derivative-free search approaches for concurrent optimization of inflow control valves (ICVs) and well controls. A set of state-of-the-art approaches possessing different search features is implemented over two reference cases, and their performance, resource requirements, and specific method configurations are compared across multiple problem formulations for completion design. In this study, problem formulations to optimize completion design comprise fixed ICVs and piecewise-constant well controls. The design is optimized by several derivative-free methodologies relying on parallel pattern-search (APPS), population-based stochastic sampling (PSO) and trust-region interpolation-based models (DFTR). These methodologies are tested on a heterogeneous two-dimensional case and on a realistic case based on a section of the Olympus benchmark model. Three problem formulations are applied in both cases, i.e., one formulation optimizes ICV settings only, while two joint configurations also treat producer and injector controls as variables. Various method parametrizations across the range of cases and problem formulations exploit the different search features to improve convergence, achieve final objectives and infer response surface features. The scope of this particular study treats only deterministic problem formulations. Results outline performance trade-offs between parallelizable algorithms (APPS, PSO) with high total runtime search efficiency and the local-search trust-region approach (DFTR) providing effective objective gains for a low number of cost function evaluations. APPS demonstrates robust performance across different problem formulations that can support exploration efforts, e.g., during a pre-drill design phase while multiple independent DFTR runs can provide local tuning capability around established solutions in a time-constrained post-drill setting. Additional remarks regarding joint completion design optimization, comparison metrics, and relative algorithm performance given the varying problem formulations are also made.

Keywords Completion design · Inflow control valves · Derivative-free optimization · Pattern-search methods · Model-based trust-region · Production optimization

1 Introduction

Field development of georesources increasingly relies on efficient drilling of horizontal and multilateral sections using geosteering technology [1] and the successful deployment of advanced inflow control technology [2]. Designing lower completion programs including inflow control valves is challenging due to geological uncertainty and the need to run time-consuming reservoir simulations to test possible config-

urations. A general approach to deal with uncertainty during model-update is to assimilate data from multiple sources, e.g., seismic data, log measurements, and, if available, production data. More efficient measurement and assimilation capability in turn enable the ability to update subsurface models also while drilling. These model-processing improvements and the availability of modern completion technology, e.g., adjustable completion strings at the wellsite [3], allow for a possible re-evaluation of the completion program before running the strings into hole. Within this context, procedures to optimize completion design and production strategy can be introduced to help adjust pre-drill solutions according to updated subsurface models.

✉ Carl Fredrik Berg
carl.fredrik.berg@ntnu.no

Extended author information available on the last page of the article

If practical, automatic search procedures can serve both as quality checks and to provide novel input to improve established designs. This work studies three derivative-free methodologies with different search characteristics for various configurations of fixed completion design and well control optimization. The main contributions of this work are the experimental setup and application results offering a comparison of the different methodologies over increasingly complex problem formulations and cases. The different formulations are tested on two cases with realistic properties, i.e., a 2D grid and a section of the Olympus model. Only deterministic properties are used to focus the study on algorithm performance for different problem configurations and increasing model complexity (a scope treating uncertainty is deferred to subsequent work). The remainder of this section introduces the three derivative-free search methodologies and provides the context and motivation for joint completion and control optimization.

1.1 Optimization methodologies

In general, beyond time limitations for well planning and reconfiguration imposed by the drilling context, optimization procedures for well planning workflows must perform efficiently since objective functions involve computationally costly reservoir simulations. Moreover, iterative procedures not only require practical search algorithms with problem-specific tuning but also fit-for-purpose formulations and parametrizations to maintain low-dimensional problems. This is particularly relevant for the efficiency of derivative-free methodologies, since the computational resources required to drive the search of such methods typically depends on the number of variables.

Three derivative-free optimization methods that rely on very different search characteristics have been implemented and tested in this study: (1) Asynchronous Parallel Pattern Search (APPS) direct-search algorithm; (2) Particle Swarm Optimization (PSO) population-based algorithm, and (3) Derivative-Free Trust-Region model-based (DFTR) algorithm. The procedures aim to complement existing workflows for pre- and post-drill selection and screening of completion configurations. Their performance is therefore studied within the described pre- and post-drill contexts. The range of optimization procedures is tested and detailed tuning performed to solve a simple two-dimensional case and a realistic three-dimensional production scenario based on the Olympus model. Specifically, comparisons are presented using equivalent performance measures based on total runtime. This should help further application within the existing workflows by meaningfully comparing different procedures even though they possess fundamentally different search properties and capabilities, i.e., intrinsic parallel and serial execution modes.

This work focuses on derivative-free methodologies because, even though they typically require a larger number of cost function evaluations, the completion problems presented in this work treat only a low number of variables (8 to 36). In general, derivative-free algorithms are typically straightforward to implement for applications relying on reservoir simulations, e.g., they do not require extensive access to simulator code often necessary for adjoint-gradient use [4], and can be more robust to numerical inconsistencies, e.g., due to adaptive simulator time-stepping for different candidate solutions [5]. Moreover, compared to gradient-based methods, these algorithms are less dependent on good initial points to perform a successful search, i.e., they are less prone to being caught in local minima at early stages during optimization. Finally, these algorithms can be applied to extended formulations of field development problems that involve significantly different variable types, e.g., concurrent solution approaches including well placement coordinates, well controls such as bottomhole pressure (BHP) and/or rates, and completion parameters (e.g., cross-section areas).

1.2 Completion optimization

Compared to passive flow controls, devices with valves can be useful to deal with underlying geological uncertainty since they enable flow adjustments after installation. (See an extensive comparison between valve and passive control devices in [6], where properties and functionality of these controls across a range of applications are discussed.) Essentially, the option to reconfigure the completion during field lifetime creates a buffer with respect to uncertainty in reservoir properties and variations in fluid conditions. However, a modern drilling workflow enhanced with fast model update capability provides a crucial opportunity to deal with such uncertainty also when installing passive completions, i.e., specifying a particular configuration along the wellbore. Passive control devices are often a more reasonable option since these can be installed at a significantly lower cost and are more robust in terms of maintenance. Moreover, even in wellbore configurations with only fixed equal-strength devices, the completions can regulate zones with overproduction since their pressure drop is commonly proportional to the squared flow rate [7].

Furthermore, the capability to operate well controls, i.e., BHP and/or rates, over reasonable ranges provides increased robustness with respect to unexpected conditions. Introducing the capability of dynamically managing sweep by varying well control settings provides an additional means to counter reservoir and fluid uncertainties that complements the functionality of the fixed ICD configuration. This aligns with results in [7] where additional controls to the optimization problem, e.g., optimizing ICDs within an integrated system including network and facility variables, will in some cases

mitigate the effects of wellbore pressure drops and in general will influence the solution in terms of design and predicted benefit. Thus, the completion optimization process should consider well controls [6, 8] to ensure optimality with respect to, at least, these two closely located, highly interdependent variable types. Altogether, this provides a context and lends support for performing joint completion design optimization involving fixed nozzle-based completions and well controls.

1.3 Paper structure

This paper is structured as follows: Section 2, discusses inflow control technology, the completion design process within a drilling workflow perspective and approaches for completion redesign. Section 3, describes the core problem setup, the objective and the iterative procedures used for optimization. Section 4, describes the two models and provides the configuration for the different cases tested in this work. Optimization performance is discussed and design solutions presented in Section 5. Finally, Section 6, summarizes the main points from the experiments and suggests topics for further work.

2 Background

This section first introduces inflow control technology and a general drilling and model-updating workflow. This serves as context to describe completion design approaches and the development of redesign workflows.

2.1 Inflow control technology

The promise of advanced wells [9], i.e., long horizontal, possibly multilateral, wellbores with smart well technology such as sophisticated sensors and inflow controls, is higher productivity due to larger coverage of the reservoir and greater flexibility for production. Such wellbores can increase field recovery not only by joining isolated regions within the reservoir but also by enabling connection to satellite formations. Moreover, the greater penetration also provides more information about the surrounding geology, which aids reservoir characterization [10]. In terms of production flexibility, longer wellbores with inflow controls allow shutting down certain regions of the reservoir, e.g., to avoid uneven production profiles and high water cuts due to wellbore friction and formation heterogeneity.

Deploying wells with advanced inflow controls can incur substantial additional drilling and completion costs that must be justified by a corresponding increase in revenue [9]. Establishing automatic procedures that can complement manual search for improved designs is therefore important to both

increase economic productivity and ensure robustness, e.g., with respect to geological and fluid contact uncertainty. Note that in this work the focus is on the first aspect, i.e., the development and testing of a set of core methodologies to find optimal control configurations with respect to economic recovery. As mentioned, this work is performed within a deterministic setting. Subsequent work will focus on the second part by extending these methodologies to apply to the uncertain case, e.g., using a random sampling strategy [11] to deal with increased computational cost.

Wells with inflow controls are typically implemented within reservoir simulators using multisegmented wellbore models [12]. Multisegment well models provide a refined representation of the well that makes it simpler to associate fluid flow to a range of control devices in various configurations [13]. General operation for wells with inflow controls is that formation fluids enter the well annulus and then flow from the annulus into the tubing by passing through restrictions to flow, e.g., nozzles, for the particular completion. This completion design imposes an additional pressure drop between the formation and the tubing that is used to balance the drawdown along the length of the wellbore. A given pressure drop depends on the velocity and density of the fluids and on the geometry of the restriction [3]. Several types of completions with different flow-through features and operational capabilities can be installed; broadly [14], inflow control completions can be passive inflow control devices (ICDs), active inflow control valves (ICVs) and autonomous inflow control devices (AICDs) [15]. ICDs are installations with preset flow restrictions, while ICV settings can be adjusted after installation from the surface. Autonomous completions can self-adjust based on the viscosities of the fluids, essentially choking back undesired fluids such as water and gas while letting oil through with relatively little obstruction.

Within a single compartment, the pressure-drop mechanism restricting the flow from the formation into the well can be adjusted in terms of shape, number and flow-through features. Multiple flow-restriction mechanisms exist with specific features [6], e.g., nozzles, long helical tubes or labyrinthine channels. This work targets the problem of configuring ICDs with fixed openings (nozzles) towards the formation, i.e., passive devices. Note, however, that the methodologies presented in this work can be extended to more sophisticated production scenarios, e.g., installations involving ICVs where individual well sections are controlled over the lifetime of the field [16, 17]. Passive devices generally require a lower up-front cost because they involve no moving parts, have straightforward run-in-hole application [8], and typically do not require intervention [10]. Due to the lack of opportunities for modification of wells with fixed-sized devices, it is important that they have an optimal design at installation [18]. Because of their fixed nature, it is important that the performance of passive devices is

explored through time, i.e., that sufficient predictive capability provided by, e.g., reservoir simulation, is embedded in the design process. Furthermore, the decision process itself, or crucial aspects of it, should be able to be repeated efficiently once new information is available, e.g., through fast model update. Overall, control devices cannot be universally implemented, but require specific characterization of static reservoir conditions and thorough study of long-term reservoir behavior [18]. This further emphasizes the importance of supporting the (re-)design process by efficiently updating subsurface representations before installation and of using dynamic models within a systematic search approach, i.e., optimization. Strategically, drilling operators seek to integrate these work components into efficient workflows to evaluate possible alternatives at the rig before finally having to run the completions into hole.

2.2 Drilling workflow and model-updating context

Subsurface models are crucial tools for decision-making and optimization. However, inherent uncertainty can make model predictions vary substantially from actual reservoir conditions and production results. Frequent model updating, at least semi-automatically on a continuous basis [19], preferably performed by assimilating information using various data sources across different scales [20], helps ensure accurate production forecasts and ultimately successful reservoir management. Keeping an evergreen model, e.g., through continuous data integration and reservoir characterization and modeling work [21], is important for optimized drilling and field recovery. Furthermore, the ability to differentiate and emphasize specific model components to support particular operational needs, i.e., fit-for-purpose modeling, is an important capability in modern field development workflows. Ultimately, the vision is to have automated workflows that continuously upload data coupled with autonomous intelligence for assimilation of particular properties necessary to produce highly relevant and useful representations of reality according to operational needs.

The various data types accumulated during drilling, e.g., actual well path, logging-while-drilling (LWD) logs and formation pressure measurements [22], are crucial to update models and decrease uncertainty in forecasts. This model-updating process within modern drilling workflows is a precursor for subsequent re-evaluation of the completion program, since the planned completion design may no longer be appropriate to current reservoir conditions. To operate efficiently within a drilling workflow, the re-evaluation of the completion program can be performed using dedicated optimization techniques. "Pre-" and "post-"drill descriptors are used in this work to refer to the general time limitations imposed on the completion design process during planning

and the real-time drilling environment, respectively. More broadly, this distinction corresponds to the development and drilling phases described in [23] in the context of well location planning and optimization.

While computationally demanding, pre-drill completion optimization efforts have the advantage of a relatively large time budget of months and even years to conduct a number of optimization runs. This time frame allows starting from multiple initial points while using different settings and testing across multiple realizations. Post-drill completion optimization efforts, on the other hand, are required to arrive at a solution in a limited time frame of hours [8, 10], ranging from 12 to 24 hours, which is the time it takes for pulling up the drillstring before running the completion into hole. Thus, within this time frame, there is the opportunity to improve the existing completion program, developed using the now-outdated pre-drill model. More specifically, a new optimization effort can be launched using an updated model of the reservoir. Importantly, this enables the development of an ICD configuration that is optimal with respect to the most recent subsurface information. In this work, the performance of the selected optimization procedures is therefore studied within two different operational contexts: (1) a pre-drill state without a specific time condition on the optimization effort and (2) a heavily time-constrained post-drill state delimited by the time from pulling-out-of-hole to laying the completion string.

2.3 Completion redesign

Time is thus a crucial aspect for drilling workflows integrating model updating and potential completion redesign. A major goal at the wellsite is to perform an efficient re-evaluation of the completion program following the acquisition of new information about the subsurface. At this point, the completion program has been developed through early-stage feasibility studies including full-field simulation and detailed single-well pre-drill completion design work. Clearly, improvements in dynamic simulation tools, e.g., more time-efficient single-well modeling techniques [24], aid the completion re-assessment before running it into the hole. These design steps are followed by post-drill evaluation and history-matching [8] to finalize the workflow.

Workflows for redesign and tuning of lower completions at the platform not only rely on engineering experience but also on expert analysis and efficient integration of updated information. Model-updating is enabled by efficient LWD data collection tools [25] and improved real-time inversion methodology [26]. In [27], a real-field case is described where formation log data are used to adjust completion settings before running these into the hole. In that case, a model is updated with the actual well path and log data, and various

simulation scenarios are run to select appropriate configurations for the completion.

In this work, a range of optimization techniques is tested to support the above redesign process. Potentially, this re-evaluation, using dedicated optimization techniques, can be started already during drilling as LWD data are retrieved and analyzed. For example, program re-evaluation can be performed at specific times during drilling. The particular timing of these steps can be determined a priori or be dependent on the state of information during drilling. Along these lines, possibly one or several trigger functions, e.g., depending on how much geological and geophysical knowledge has evolved compared to a previous state, can be defined and used to initiate the completion optimization routine.

2.4 Flow control applications

An early comparison of two flow control devices to delay gas breakthrough, one fixed at installation time and another that could be modified from the surface during production, is given by [12]. Another early real-field application is presented in [28], where a number of ICDs are installed in horizontal sections to balance inflow along the wellbore and gain better control of the gas-oil ratio. Several simulation cases and two real-field cases are presented in [29] to demonstrate key benefits of increased sweep, reduced flow variation and compensated wellbore friction. Similar benefits from real-field installations of nozzle-based passive devices in one sandstone and one carbonate reservoir are described in [30]. In [31], installation of flow control devices is studied to avoid early gas and water breakthrough in a production scenario with varying permeability profile caused by a horizontal well traversing a set of dipping multistacked reservoirs. Other authors, e.g., [16, 18], describe the challenges of balancing inflow with control devices as increasingly longer horizontal sections are being installed, while [3] describes specific production and injection challenges for horizontal wellbores in naturally fractured carbonate reservoirs. Overall, the main targets are to control and reduce water coning and/or gas cresting, to eliminate uneven lateral influx due to abrupt changes in permeability and to avoid sand production.

2.5 Completion design approaches

Various approaches for how to combine predictive tools with engineering experience to evaluate completion options have been presented in the literature. In [32], a decision-tree methodology is presented emphasizing formation stability followed by steps analyzing economic performance and risk in the selection process for completion solutions. In [18], on the other hand, a simulation-based approach is used to assess

the impact of control devices for typical production problems of fluid distribution and breakthrough delay. A general completion design workflow is provided by [3] to change the configuration of nozzle-based completions at wellsite.

In particular, [10] presents a classification for conducting completion design based on the complexity of the reservoir simulation models used. In that work, completion design workflows of increasing complexity are described. The first approach relies on well-centric simplified geological and reservoir simulation models built from logs or pseudo-logs, and is suitable when the time frame is short and only limited data are available. Because of the local nature of the model, however, this workflow cannot be used to provide long-term predictions. The second approach relies on a well-centric simplified skeleton grid for the reservoir model with geological properties along the wellbore derived from an existing full-field model. Limiting the updates of the reservoir model to the near-wellbore region enables fast updates of the model using LWD data. The well-centric nature of these models facilitates efficient assimilation of LWD data once these data become available, and supports completion design by matching configurations with surrounding geological features, e.g., log-derived permeability distribution. The third approach is to use a sector model, i.e., to simulate parts of the full-field reservoir simulation model with flux or pressure boundary conditions from the full field model. Furthermore, completions are often modeled with increased resolution from local grid refinement around the well, in particular when interference with other wells is expected. This provides the most accurate results, but is also the most time-consuming approach. Three-dimensional sector simulation models with local grid refinement are used in [8, 33] to tailor completion designs. A combined approach is described in [7] where the ICD settings from a well-centric optimization workflow was used in a full-field model to assess interference between different wells. Other approaches, e.g., [34], use proxies (fluid travel times) in automated routines to reduce recovery variations between compartments, but do not explicitly apply mathematical search routines to iterate over candidate solutions. In contrast, in this work, an approach that explicitly couples mathematical search algorithms with reservoir simulation models to optimize completion design is presented. The overall problem setup and the range of optimization techniques tested in this work are presented next.

3 Methodology

The start of this section presents the setup and the mathematical formulation for the ICD and control problem being addressed in this work. The problem setup characterizes the

level of abstraction used to define the system components and optimization variables. The mathematical formulation formalizes the problem setup and specifies the main variables of interest, the associated constraints and the objective. The three derivative-free optimization algorithms used to search for optimal solutions are then described. These descriptions focus on the main operations and features of the methods, and present particular applications to the control problem.

3.1 Problem setup

The ability to divide wellbores into segments not only provides improved accuracy for simulation of horizontal wells and multilaterals [35, 36], but also add additional flexibility when defining a problem setup for optimization of ICDs. Similar to [10], the ICD problem setup in this work consists of a fixed number of compartments of equal length. Specifically, the well completion design problem addressed in this work considers the selection of the most suitable cross-sectional flow area associated with an ICD within each compartment. These cross-sectional flow areas can be treated as continuous variables with bound-constraints in the optimization methodologies.

For optimization purposes, in completion designs in which there are possibly multiple ICDs lying within two packers, the ICDs can be combined into an equivalent ICD. This equivalent ICD is an effective cross-sectional area that corresponds to the combined effect of all the nozzles in the compartment. In particular, this abstraction is useful to reduce the number of variables in the optimization problem. Ultimately, in practice, a reverse-engineering step is required to convert an optimal cross-sectional area obtained by optimization methodologies into an actual operational number and (possibly stepwise) size of nozzles and other hardware specifications to realize the corresponding drawdown profile (see, e.g., [33]).

Other common variables for overall completion design are the number and length of compartments (determined by the positions of the packers), which, either directly or indirectly, influence the flow area determining inflow from the formation. Though packers are typically placed according to how permeability is distributed along the wellbore [15], compartment length could also be subject to optimization, e.g., by varying the measured depth of the packers. Potentially, joint optimization of compartment lengths and ICD settings could be performed either simultaneously or possibly in a sequential manner, i.e., iteratively varying packer locations and nozzle sizes. Varying packer location is likely to have significant effect in cases where the wellbore traverses zones with extensive boundaries that are fully or partially sealing. Though such cases are interesting, in this work, compartment lengths are kept constant to both keep problem dimension and

complexity low and emphasize the joint optimization of well controls and nozzle sizes.

3.2 ICD design and well control optimization

The ICD design and well control optimization problem tackled in this work consists of finding the set of cross-sectional flow areas for the ICDs and the well controls (either BHPs or flow targets) that improve the reservoir performance. The reservoir performance is assessed by means of an objective function that assigns some economic criteria to the reservoir in- and outflows over time. Different objective function formulations can influence the optimal completion design [7], e.g., different solutions may be obtained when using objectives emphasizing higher economic return through higher production versus objectives that prioritize equalizing inflow along the wellbore [37]. Though economically optimal, searches based only on economic measures may not yield reasonable nor practical completion and control configurations from a design and operational perspective. Different formulations can also influence the optimization search itself, e.g., discontinuity and smoothness properties can have significant impact on algorithm exploration and convergence performance, even for derivative-free methodologies that do not require well-defined gradient information and are less susceptible to noise.

3.2.1 Problem formulation

This work follows the general formulations presented in [38] and [39] for the optimization problem and objective function. The optimization problem is given as

$$\begin{aligned} \mathbf{u}^* &= \underset{\mathbf{u}}{\operatorname{argmax}} J(\mathbf{x}, \mathbf{u}), & (1) \\ \text{s.t. } g(\mathbf{x}, \mathbf{u}) &= 0, & (2) \\ \mathbf{u} &\in D. & (3) \end{aligned}$$

Here, $J(\mathbf{x}, \mathbf{u})$ represents the objective function, where \mathbf{x} corresponds to the discretized-in-space-and-time state variables of the reservoir system (i.e., pressure and saturation), while \mathbf{u} are the decision variables being optimized, which correspond to specific model parameters of interest. Thus, in this work, \mathbf{u} represents both ICD cross-sectional areas and well control pressure and rate settings, within a feasible space D . The constraint given by Eq. (2) represents the system of reservoir equations solved numerically by the simulator for the unknown state variables \mathbf{x} . This means that, for a given set of model parameters \mathbf{u} , a reservoir simulation will be performed to satisfy constraint Eq. (2), and the calculated state variables of the reservoir \mathbf{x} will also be used to compute the objective function given by Eq. (1).

In this work, optimizations are performed using an undiscounted Net Present Value (NPV) formulation for the objective J . Following [38], J is given as

$$J(\mathbf{x}, \mathbf{u}) = \sum_{n=1}^N \left(\sum_{i=1}^{N_w} \sum_{p=1}^{N_p} C_{p,i} q_{p,i}(\mathbf{x}, \mathbf{u}) \right) \Big|_{t=t_n} \Delta t_n. \quad (4)$$

In this formulation, the outer sum is taken over N simulation time steps t_n , while N_w and N_p correspond to the number of wells and flowing phases, respectively. For well i , the terms $C_{p,i}$ and $q_{p,i}$ represent respectively the price/cost for produced/injected volumes and the production/injection flow rate for phase p .

Tentatively, functions that describe overarching flow patterns and that incorporate specific understanding of the reservoir and of the particular production scenario can be used to formulate new types of objectives that augment purely economic measures [34]. The goal of augmented formulations of this type would be to focus on quantifying essential geology- and geophysics-based phenomena for engineering-based decision-making. Possibly, such formulations would yield solutions that are both a better match to overall engineering considerations and also less susceptible to underlying model uncertainty. Exploring a range of appropriate, possibly more efficient, cost function formulations for the completion design optimization problem is, however, beyond the scope of this paper and will be treated in subsequent work.

3.3 Optimization methodologies

This section presents the three different derivative-free optimization algorithms applied to the ICD and well control problem: A direct-search Asynchronous Parallel Pattern Search algorithm (APPS), a population-based Particle Swarm Optimization algorithm (PSO), and a model-based Derivative-Free Trust-Region algorithm (DFTR).

The algorithms are implemented in the open-source optimization framework for georesources field development problems `FieldOpt` [40] with up-to-date research code available at [41]. General descriptions provided in the next sections focus on core search features and algorithm characteristics. Detailed descriptions of `FieldOpt`'s versions of APPS [40], PSO [42] and DFTR [43, 44] can be found in these publications, while the main references are provided in the corresponding sections given below.

As a general rule, the computational expense of derivative-free search procedures typically increases with the number of variables [45]. It is therefore crucial and common to take advantage of search features within the different procedures that can be effectively parallelized, e.g., concurrent computation of the cost function sampling that involves

time-consuming reservoir simulations. Given the pre-/post-drilling time perspective discussed earlier, a main assumption for later analysis is that the procedures presented in this work have available all computational resources necessary to fully exploit their intrinsic capabilities for distributed computing. In practice, this means, e.g., having available during optimization an equal number of cores as the total number of stencil points for the APPS algorithm and an equal number of cores as the number of particles for the PSO runs. Asserting this assumption is important because it predicates the performance analysis presented in Section 5 where optimization runs are compared not only in terms of computational cost, i.e., total number of function evaluations, but also with respect to actual elapsed runtimes.

Parallel computation of a set of sampling points is straightforward for direct search and population-based procedures that, in a very broad sense, iteratively construct and test candidate solutions in batches. This is not the case, however, for the DFTR procedure that conducts its search based on explicit model building. In essence, DFTR performs a sequential sampling of the search space, since finding a new point is the result of a minimization of the current model approximating the objective. Moreover, at each iteration, the procedure relies on various interacting model-monitoring, maintenance and rebuilding operations, e.g., a verification of whether the overall spread of points in the sample set satisfies geometric conditions, a check on point acceptance criteria, and a test of model accuracy against expected performance measures. Descriptions and pseudocode for APPS, PSO and DFTR procedures are provided next.

3.3.1 APPS

APPS [46] is an advanced iterative procedure based on a pattern search derivative-free algorithm, i.e., its search relies on straightforward comparisons of cost function values computed across a set of points specified by a stencil or pattern. Pattern search has global convergence properties [47, 48] in the sense that, given algorithmic conditions are met, convergence to a local optimum is assured irrespective of the initial point. Various optimization procedures with different features, e.g., Mesh Adaptive Direct Search (MADS) [49, 50], are based on the core pattern search operation. These core features of the pattern search algorithm are presented next, followed by a brief description of the asynchronous implementation applied in this work.

The basic pattern search algorithm relies on local polling of the cost function space around a current best solution point. The algorithm uses a stencil centered at this point to determine search directions and distance (step size) away from the current solution. A stencil corresponding to the coordinate

axes is commonly used (although other stencils are possible, any stencil must span the search space according to certain geometric properties to ensure at least one descent direction and algorithm convergence [48]). The stencil step size can be increased or decreased by specified factors depending on whether the polling operation finds points that improve the objective, or not. If one of the stencil end points yields an improvement, then the algorithm uses this point as the center of the stencil at the next iteration. Moreover, the stencil step size may be increased to further explore this promising area. If no improvement is found, then the stencil step size is decreased, and the polling continues until a specified minimum step size threshold is reached. A pseudocode for the APPS implementation used in this work is provided in Algorithm 1.

The independent nature of the set of stencil points enables a concurrent evaluation of the corresponding objective values over a number of computing cores. However, parallel implementations of pattern-search methodologies can be inefficient for simulation-based optimization where cost function evaluation runtimes may vary significantly for different solution candidates. In an oilfield development context, this is more likely to be the case for well placement problems than for well control problems, since varying well configurations may yield very different simulator runtimes due to their parametrization within reservoir grids and geological heterogeneity. Still, it is important for distributed implementations dealing with simulation-based problems to take advantage of possible idle cores during optimization. In particular, the APPS procedure applied in this work enables an asynchronous computation of sampling points, i.e., the polling previously described is no longer dependent on a single comparison of all points in the stencil before proposing new stencil points. Rather, during the iteration, if the evaluation at certain points has been completed, then new sampling points can be proposed and their evaluation started at the cores that are now available, i.e., without waiting for the complete evaluation of the original set. Implementations are found in [46], where further uncoupling polling directions from the stencil enable sophisticated strategies to determine standalone properties, agency and interaction between the individual stencil directions. For example, in that work, polling directions may be treated as independent agents with individual step size properties that pass the current best point and step size information dynamically before making a new polling decision along their particular direction.

The notation used in Algorithm 1 is as follows. The function being optimized is f , step sizes are denoted by Δ , and the initial point is denoted by \mathbf{x} with corresponding lower and upper bounds \mathbf{x}_{lb} and \mathbf{x}_{ub} . The sets of trial and evaluated points are denoted respectively by \mathcal{T} and \mathcal{E} , while the search directions \mathbf{d}_i are p -dimensional vectors in the set

\mathcal{D} . In each iteration of the algorithm, some additional information of the new trial point \mathbf{y} is stored to ensure proper asynchronous execution of the algorithm. Thus, tags $\text{TAG}(\mathbf{y})$ and $\text{PARENT_TAG}(\mathbf{y})$ correspond to the current point and the parent point, i.e., the point that generated the current point, while $\text{DIRECTION_INDEX}(\mathbf{y})$ and $\text{STEP}(\mathbf{y})$ label the trial-point direction index and the associated step size, respectively.

Algorithm 1 Pseudocode for APPS [46, 47, 50].

Inputs: Initial point \mathbf{x} , variable bounds \mathbf{x}_{lb} , \mathbf{x}_{ub}

Parameters: Step size tolerances Δ_{\min} and Δ_{tol}

Initialization:

Specify search directions: $\mathcal{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_p\}$

Set initial step sizes: $\Delta_i = \Delta_{\text{init}}$

Generate Trial Points:

Let $\mathcal{I} := \{i : \Delta_i \geq \Delta_{\text{tol}} \ \& \ \tau_i = -1\}$ be the tentative iterates

Let $\tilde{\Delta}_i \leq \Delta_i$ be the length of the longest feasible step size from \mathbf{x} along direction \mathbf{d}_i

Create new trial point $\mathbf{y} := \mathbf{x} + \tilde{\Delta}_i \mathbf{d}_i$

Save the following information from \mathbf{y} :

– $\text{PARENT_TAG}(\mathbf{y}) := \text{TAG}(\mathbf{x})$

– $\text{DIRECTION_INDEX}(\mathbf{y}) := i$

– $\text{STEP}(\mathbf{y}) = \Delta_i$

Set $\tau_i := \text{TAG}(\mathbf{y})$ as the trial point tag

Add new trial point \mathbf{y} to the set \mathcal{T}

Exchange Trial Points:

Send new trial points in \mathcal{T} for evaluation

Collect set of evaluated trial points in \mathcal{E}

Process Evaluated Trial Points:

Let $\mathbf{z} \in \mathcal{E}$ be the point that $f(\mathbf{z}) > f(\mathbf{y})$, $\forall \mathbf{y} \setminus \mathbf{z} \in \mathcal{E}$

if $f(\mathbf{z}) > f(\mathbf{x})$ **then**

Replace \mathbf{x} with the new best point \mathbf{z}

Check convergence based on function value

Set $\Delta_i := \max\{\text{STEP}(\mathbf{z}), \Delta_{\min}\}$

Set $\tau_i := -1 \ \forall i = 1, \dots, p$

Prune the evaluation queue \mathcal{T}

Go back to **Generate Trial Points**

else

for all $\mathbf{y} \in \mathcal{E}$ **do**

if $\text{PARENT_TAG}(\mathbf{y}) = \text{TAG}(\mathbf{x})$ **then**

$i := \text{DIRECTION_INDEX}(\mathbf{y})$

$\Delta_i := \frac{1}{2} \Delta_i$

$\tau_i := -1$

end if

end for

Delete all points in \mathcal{E}

Check convergence based on step length

Go back to **Generate Trial Points**

end if

3.3.2 PSO

The general description of the PSO applied in this work follows [51]. PSO is a stochastic search algorithm that has been

applied for both standalone and joint optimization of well placement and controls [23, 45]. Search is conducted using a fixed number of particles that move as a collective through the solution space, mimicking a swarm or flock of birds. Overall operation is driven by information sharing and cooperation between the particles using a series of position and velocity updates at each iteration. The search is initiated using a random population; thereafter, each particle is moved based on local information obtained from neighboring particles, while memory of the best performance so far is maintained. The performance and location of any one particle is communicated to other particles within a specified neighborhood (see [51] for a wider discussion of different neighborhood types, e.g., geographical or social). Finally, the introduction of stochastic components within each PSO velocity update enables a global search feature to avoid local optima and facilitate wide exploration.

Parallel implementation of the PSO search procedure is straightforward and, in this work, all PSO runs performed are provided the number of computational cores needed to fully parallelize the evaluation of all candidate solutions in a given population.

Based on the recent review paper [52], a pseudocode for the PSO algorithm used in this work is provided in Algorithm 2. While the main operations and conditions are described in the pseudocode, the following set of Eqs. (5)–(8) form the basis of the algorithm.

$$\mathbf{p}_i^{*,t} := \{\mathbf{x}_i^k \mid f(\mathbf{x}_i^k) \geq f(\mathbf{x}_i^l) \forall l \in [1, t]\} \tag{5}$$

$$\mathbf{g}^{*,t} := \{\mathbf{p}_i^{*,t} \mid f(\mathbf{p}_i^{*,t}) \geq f(\mathbf{p}_j^{*,t}) \forall j \in [1, N]\} \tag{6}$$

$$\mathbf{v}_i^{t+1} = \omega \mathbf{v}_i^t + c_1 r_1 (\mathbf{p}_i^{*,t} - \mathbf{x}_i^t) + c_2 r_2 (\mathbf{g}^{*,t} - \mathbf{x}_i^t) \tag{7}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{8}$$

Here, i denotes the particle’s index, t is the iteration, f is the objective being optimized, \mathbf{x}_i^t is the position vector of particle i at iteration t , and N is the swarm size. Moreover, $\mathbf{p}_i^{*,t}$ is the best position for particle i at iteration t , while $\mathbf{g}^{*,t}$ is the global best position at iteration t , \mathbf{v}_i^t is the velocity vector and ω is the inertia weight. Finally, r_1 and r_2 are two random numbers independently generated within the range of $[0, 1]$, while c_1 and c_2 are typically called acceleration coefficients or learning factors (in this work, these factors are abbreviated as Learning{Fac.1, Fac.2} in Tables 4 and 5). Note that we only need an iterative update to keep track of the particle’s best position, as shown in Algorithm 2, and thus we do not need to calculate Eq. (5).

Algorithm 2 Pseudocode for PSO [52].

Inputs:
 Swarm size N
 Problem dimensionality D
 Maximum number of iterations T
 Lower bound of the search space LB
 Upper bound of the search space UB

Main Iteration:
 Initialize the swarm randomly
 Iterate through the swarm:
for $i = 1$ to N **do**
 Set velocities $\mathbf{v}_i^0 :=$ random vector $\in [LB, UB]^D$
 Set positions $\mathbf{x}_i^0 :=$ random vector $\in [LB, UB]^D$
 Set particles $\mathbf{p}_i^{*,0} := \mathbf{x}_i^0$
end for
 Set initial global best position $\mathbf{g}^{*,0}$ using Eq. (6)
 $t := 1$
while $t \leq T$ **do**
 for $i := 1$ to N **do**
 $r_1, r_2 :=$ random numbers $\in [0, 1]$
 Update velocities \mathbf{v}_i^{t+1} using Eq. (7)
 Update positions \mathbf{x}_i^{t+1} using Eq. (8)
 if $f(\mathbf{x}_i^{t+1}) < f(\mathbf{p}_i^{*,t})$ **then**
 $f(\mathbf{p}_i^{*,t+1}) := f(\mathbf{x}_i^{t+1})$
 else
 $\mathbf{p}_i^{*,t+1} = \mathbf{p}_i^{*,t}$
 end if
 Update best particles $\mathbf{g}^{*,t+1}$ using Eq. (6)
 $t := t + 1$
end for
end while

3.3.3 DFTR

Below follows a general presentation of the DFTR algorithm and thereafter descriptions of each of its main steps and operations. The overall presentation follows [48] while for a comprehensive description of the main steps and operations see [44]. Finally, Algorithm 3 at the end of this section assembles the various DFTR steps and operations to present the core functionality of the method.

The DFTR algorithm DFTR is a sophisticated derivative-free procedure that relies on local quadratic approximations of the objective to guide its search. This modeling of response surfaces is particularly useful when dealing with black-box functions, e.g., from simulated data points. The method builds polynomial-based interpolation models to capture cost function curvature, which enables a robust search. Moreover, the method enables building underdetermined approximation models, which is a feature that provides faster convergence with fewer function evaluations. (In this regard, note that similar methodologies have been applied that use a fixed set of sample points, e.g., the BOBYQA package used by [53] for well location and completion optimization.) Finally, notice

that the DFTR is essentially a sequential algorithm (although for some particularly structured problems, the procedure may be decomposed into a series of subproblems that can be solved in a distributed manner, as discussed in [54]).

General operation The cost function is denoted by $f(\mathbf{x})$ for a given vector of variables $\mathbf{x} = (x_1, x_2, \dots, x_n)$ for an n -dimensional problem. During its execution, the DFTR algorithm samples the cost function f , and the function values are then used to build a model m_k . At each iteration k , the algorithm optimizes the current model m_k inside a trust region (a ball centered around the current iterate \mathbf{x}_k) to compute a new point \mathbf{x}_{k+1} . The cost function f is then evaluated at the new point, and the point is either accepted or rejected as the new iterate. The model is constantly maintained throughout the optimization by various operations that use recent function evaluations. These operations are discussed in more detail next.

Generation of models A core DFTR feature is its reliance on interpolation models to find new iterates. At each iteration k , the incumbent model m_k interpolates the objective f in a set of evaluated points \mathcal{Y}_k using quadratic polynomials. The interpolation equations for an n -dimensional problem uses the basis ϕ of all monomials of degree at most 2. The model is a linear combination of the terms of the basis:

$$m(\mathbf{x}) = \sum_{i=0}^L \alpha_i \phi_i(\mathbf{x}) \quad (9)$$

where α_i are the coefficients of the linear combination and L is the total number of terms of the basis, i.e., $L = (n + 1)(n + 2)/2$. The coefficients α_i can be obtained by solving the interpolation condition that $m(y_i) = f(y_i)$ for every point $y_i \in \mathcal{Y}$, which can be denoted in matrix form by the following system of linear equations:

$$M(\phi, \mathcal{Y})\alpha = \mathbf{f}(\mathcal{Y}) \quad (10)$$

where $M(\phi, \mathcal{Y})$ is a matrix with the linear combinations representing the m_i models for all points $i \in \mathcal{Y}$, and $\mathbf{f}(\mathcal{Y})$ is a vector with the function values $f(y_i)$.

Underdetermined models An exact solution to this system of equations would require at least the same number of interpolation points as the number of terms of the basis. For many applications, including reservoir engineering problems, it is impractical to require this number of simulations before being able to start the iterative sequence. In fact, the use of underdetermined models is key to efficiently implementing quadratic interpolation in derivative-free optimization.

Thus, the first models are built using just a few points and are gradually improved as more simulations are performed.

From the initialization up to n points, the models are built using coarse finite differences with only the linear terms of the basis. From $n + 1$ to L points, the interpolation condition is not sufficient to compute a fully determined quadratic model. However, the additional degrees of freedom (i.e., the number of evaluated points minus the number of linear terms of the basis) allow the construction of more elaborate models, e.g., a model with minimum Frobenius norm Hessian around the incumbent solution \mathbf{x}_k :

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

Here, $\nabla^2 m(\mathbf{x}_k)$ is the Hessian matrix with minimum Frobenius norm while \mathbf{s} is the step.

Even with some curvature captured by a minimum Frobenius norm Hessian, the model might still yield inaccurate approximations. The quality of the model is typically measured in terms of its modeling errors. A model considered to be Fully-Linear (FL) has at least $n + 1$ points with bounded modeling errors as given by the following equations:

$$|f(\mathbf{x} + \mathbf{s}) - m(\mathbf{x} + \mathbf{s})| \leq \kappa_{ef} \Delta^2, \|\mathbf{s}\| < \Delta, \quad (11a)$$

$$\|\nabla f(\mathbf{x} + \mathbf{s}) - \nabla m(\mathbf{x} + \mathbf{s})\| \leq \kappa_{eg} \Delta, \|\mathbf{s}\| < \Delta, \quad (11b)$$

where κ_{ef} and κ_{eg} are function-dependent parameters, \mathbf{s} is the step, and Δ is the radius of the trust region.

Calculation of new points At the beginning of each iteration k , the DFTR algorithm calculates a new point by optimizing the model m_k inside the trust region which is a ball of radius Δ_k centered around the iterate \mathbf{x}_k :

$$\underset{\mathbf{s}}{\text{maximize}} \quad m_k(\mathbf{x}_k + \mathbf{s}) \quad (12)$$

$$\text{s.t. } \|\mathbf{s}\|_\infty \leq \Delta_k \quad (13)$$

$$\mathbf{x}_{\text{lb}} \leq \mathbf{x}_k + \mathbf{s} \leq \mathbf{x}_{\text{ub}} \quad (14)$$

where the step \mathbf{s} is limited by the trust-region radius Δ_k , and the new point must honor the lower (\mathbf{x}_{lb}) and upper (\mathbf{x}_{ub}) bounds. This procedure is called the *trust-region subproblem*. DFTR convergence can be obtained by finding a step \mathbf{s} that simply provides sufficient ascent, also known as Cauchy or eigen-step, as discussed in [48, 55]. (However, achieving optimally in the sub-problem is preferred to improve convergence and save costly reservoir simulations in the main optimization loop.)

Trial point acceptance A tentative new point \mathbf{x}_k^+ is obtained by solving the trust-region subproblem (the DFTR implementation in this work solves the quadratic optimization subproblem using the standard solver SNOPT [56]). As

mentioned, to improve performance, the trial point \mathbf{x}_k^+ is accepted (or rejected) to become the next iterate \mathbf{x}_{k+1} based on whether it yields a sufficient ascent and the incumbent model is deemed sufficiently accurate. Improvement in the objective function and mismatch between the model and the actual function are both measured with the following relationship [44]:

$$\rho_k(\mathbf{x}_k, \mathbf{x}_k^+) = \frac{f(\mathbf{x}_k^+) - f(\mathbf{x}_k)}{m_k(\mathbf{x}_k^+) - m_k(\mathbf{x}_k)}. \quad (15)$$

Here, $\rho_k(\mathbf{x}_k, \mathbf{x}_k^+)$ is the ratio of the actual ascent and the predicted ascent. Notice that the denominator is positive since \mathbf{x}_k^+ is the position where the model m_k is maximal within the search radius Δ_k around \mathbf{x}_k . A $\rho_k(\mathbf{x}_k, \mathbf{x}_k^+)$ close to 1 indicates good model accuracy whereas less accurate models present either too small or overly large values.

For given tuning parameters η_1 and η_0 , if ρ_k is sufficiently large (i.e., $\rho_k \geq \eta_1 > 0$), or acceptably large (i.e., $\rho_k \geq \eta_0 > 0$, and the model is FL), then the point is accepted as the next iterate ($\mathbf{x}_{k+1} = \mathbf{x}_k^+$). If the point is accepted, then the model is updated and the trust-region center is moved towards the new iterate. On the other hand, a small ρ_k indicates that either the trust region is overly large or that the model is inaccurate (and thus not FL). Furthermore, if the model is FL and ρ_k is small, then reducing the radius will reduce the error [48]. However, if the model is not FL, then it must be improved by model maintenance operations [57]. In that case, the rejected point will be included in the interpolation set if it contributes to reducing the error (since, even though rejected, the point contains valuable information obtained through simulation). If the rejected point does not improve the model, it will not be included in the interpolation set, the trust-region radius will be adjusted and another point will be computed subsequently. For brevity, we omit details on polynomial interpolation and model maintenance procedures here, but these concepts are discussed extensively in [48].

Radius update The factor ρ_k also influences the size of the trust-region radius. If $\rho_k \geq 1$, then the radius is increased as follows:

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

This equation prevents the radius from growing excessively in each step by bounding it to twice the step size $\|\mathbf{s}_k\|$ and to the radius size upper bound Δ_{\max} .

Finally, if $\rho_k < \eta_1$, then the radius is reduced if the model m_k is FL (as the error will be reduced according to Eqs. (11a)–(11b)). However, if the model is not FL, then the radius remains the same as there is no guarantee the model will be improved by reducing the radius. In this case, appropriate

model maintenance procedures are employed to improve the model accordingly [58].

Criticality step As algorithm convergence depends both on sufficient objective improvement and model accuracy, a special procedure checks the so-called *criticality measure* to avoid early convergence to non-optimal solutions. For unconstrained problems, the criticality measure applies a simple gradient norm, while for bounded problems, it relies on a projected gradient equation given as follows:

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

This definition represents the distance between \mathbf{x}_k and the projected point $\mathbf{x}_k + \nabla m_k(\mathbf{x}_k)$ onto the feasible set [59].

Equations (11a) and (11b) certify bounded-error models provided they are FL. Ideally, when convergence is near ($\sigma[m_k, \mathbf{x}_k] < \epsilon_c$, for a small $\epsilon_c > 0$), both the gradients of the model $\nabla m(\mathbf{x}_k)$ and of the actual objective $\nabla f(\mathbf{x}_k)$ become small. However, if the radius Δ_k is too large, the gradients might differ in magnitude and direction, which can lead to slow convergence of the algorithm. The *criticality* step is a solution to this issue as was addressed in [60] and [48, Chapter 10]. It ensures that the radius of the trust region is comparable to the criticality measure, reducing it when necessary. The reduction should occur iteratively as the criticality measure $\sigma[m_k, \mathbf{x}_k]$ depends on the model m_k , and thus also on radius Δ_k . The radius is reduced by a factor $\omega \in (0, 1)$, so $\Delta^{(1)} = \omega\Delta_k$, and model maintenance procedures make a new model $m^{(1)}$ which is FL within the new trust region radius $\Delta^{(1)}$. The procedure is repeated until the criticality measure is sufficiently high ($\sigma[m^{(1)}, \mathbf{x}_k] \geq \mu\Delta^{(1)}$, for $\mu > 0$).

The various DFTR steps and operations discussed above are collected into the full DFTR algorithm as presented in Algorithm 3. The next section describes the two optimization cases and the experimental setup used in this work, while the section after presents and analyzes the results.

4 Experimental cases

Two cases, Case 1 and Case 2, are presented for optimization of inflow and well control settings. Case 1, shown in Fig. 1, comprises a 2D reservoir model with a 60×60 grid that is a cut-out of layer 21 from the SPE 10 model [61]. Case 2, shown in Fig. 2, is a $86 \times 38 \times 7$ section of the Olympus benchmark grid [62]. This sub-section includes two major faults that run through the model. Both cases involve a single long deviated producer and two injectors. The scope of this

Algorithm 3 Pseudocode for DFTR [44]. Note $\|\cdot\|_\infty$ denotes the Frobenius norm while abbreviation *bct* stands for "before criticality test".

Inputs: Initial point \mathbf{x}_1 , variable bounds $\mathbf{x}_{lb}, \mathbf{x}_{ub}$
Parameters: initial radius (Δ_1^{bct}), maximum radius (Δ_{max}), thresholds for acceptance of steps ($1 > \eta_1 > 0, \eta_0 \geq 0$), criticality step threshold (ϵ_c)

Compute initial model m_1^{bct}
 Add function values from points in m_1^{bct} to \mathcal{Y}
for $k = 1, 2, \dots$ **do**
 if $\sigma[m_k^{bct}, \mathbf{x}_k] < \epsilon_c$ **then**
 $(m_k, \Delta_k) \leftarrow \text{CriticalityStep}(m_k^{bct}, \Delta_k^{bct})$
 else
 Set $m_k = m_k^{bct}$ and radius $\Delta_k = \Delta_k^{bct}$
 end if
 if $\Delta_k < \Delta_{tol}$ **then**
 stop and return solution \mathbf{x}_k and $m_k(\mathbf{x}_k)$
 end if
 Compute step \mathbf{s}_k by solving Eqs. (12)–(14)
 Set $\mathbf{x}_k^+ = \mathbf{x}_k + \mathbf{s}_k$
 Evaluate $f(\mathbf{x}_k^+)$ and $\rho_k(\mathbf{x}_k, \mathbf{x}_k^+)$ (using Eq. 15)
 if $\rho_k \geq \eta_1$ or ($\rho_k > \eta_0$ and the model is FL) **then**
 Set $\mathbf{x}_{k+1} = \mathbf{x}_k^+$
 Add \mathbf{x}_k^+ to interpolation set \mathcal{Y}
 Compute an updated model m_{k+1}^{bct}
 else
 Set $\mathbf{x}_{k+1} = \mathbf{x}_k$ (keep the incumbent point)
 if \mathbf{x}_k improves the model m_{k+1}^{bct} **then**
 Add \mathbf{x}_k^+ to interpolation set \mathcal{Y}
 else
 Generate improved model m_{k+1}^{bct}
 using model maintenance procedures
 Compute another trial point \mathbf{x}_k^+
 end if
 end if
 if $\rho_k \geq \eta_1$ **then**
 Increase radius:
 $\Delta_{k+1}^{bct} = \min[\max(\Delta_k, 2\|\mathbf{s}_k\|), \Delta_{max}]$
 else if Model was FL **then**
 Decrease radius: $\Delta_{k+1}^{bct} = \frac{1}{2} \Delta_k$.
 else
 Keep the same radius: $\Delta_{k+1}^{bct} = \Delta_k$.
 end if
end for

work is to study the resulting fluid flow patterns along and towards the production wellbore due to optimized control settings. The following case descriptions therefore emphasize initial production scenarios for these two cases and the optimization setup.

4.1 Case 1 production scenario

Case 1 is based on a single-layer reservoir model that provides a horizontal two-phase waterflooding production scenario. The injection is driven by two vertical injectors (INJ1 and INJ2) in the northern part of the reservoir. Produc-

tion is from one horizontal producer (PROD1) in the southern part, starting with its heel in the west and ending with its toe in the east.

Figure 1 shows the porosity and permeability fields for Case 1, in addition to the relative positioning of the wells. The case has a clear high-permeability region that extends diagonally from northwest to southeast. The southeastern part of the high-permeability region is traversed by approximately the last two-thirds of the PROD1 trajectory. This particular configuration is expected to lead to early water breakthrough starting close to the toe and cause bypassed oil around the lower-permeability area close to the heel.

The defined objective drives the optimization both to increase oil recovery while decreasing water production and injection. In the search for an optimal control configuration, the push to reduce water overproduction due to high permeability is balanced by the revenue increase caused by higher hydrocarbon production from the same zone. The implementation of ICDs along the well enables zone-specific flow restrictions that facilitate the development of a stable water front towards the wellbore. For Case 1, an improved configuration of fixed-sized valves will try to avoid water overproduction by restricting fluid flow from the part of the well near the high-permeability zone.

The Case 1 producer, PROD1, has a length of 1200 m and is compartmentalized into eight segments of equal length. The optimization setup in this work models each of the PROD1 compartments as being deployed with a single ICD with fixed valve size throughout the production timeframe. Each ICD is represented using a single continuous optimization variable that accounts for the total perforation cross-sectional flow-through area of the device in the corresponding compartment. As previously mentioned, this parametrization is practical for optimization purposes because it associates a single variable to each compartment, thus keeping the total number of variables representing ICDs low. Moreover, this straightforward parametrization enables potential scaling of the problem, e.g., to treat joint configurations where not only production but also well placement variables are involved, and/or to deal with completions with valve settings that vary in time. (Following the above parametrization, the dimension of a problem involving devices with variable inflow settings would increase by a factor equal to the number of control periods defined for the devices.)

More to the point, the level of abstraction of the above parametrization is considered sufficient to treat the reservoir-grid scale fluid distribution for the type of cases presented in this work. That is, a higher ICD and compartment density can provide higher resolution but is not expected to substantially affect the overall flow pattern evolution to warrant the increase in problem dimension. This perspective and chosen level of refinement fits within the larger field development context described for this work, where the aim of the partic-

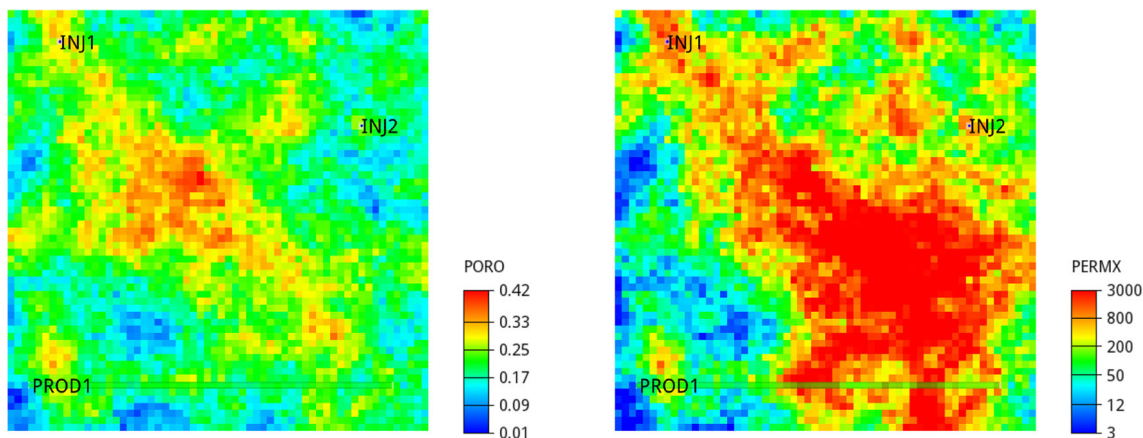


Fig. 1 Porosity (left) and permeability (right) fields for Case 1. Horizontal producer PROD1 is located at the southern part of the reservoir, while vertical injectors INJ1 and INJ2 are located in the northern part

ular optimization effort is to arrive at solution configurations that may guide and support the overall well planning and design workflow. Within such workflow, the team of drilling and reservoir engineers and completion specialists will in turn translate the obtained ICD and control solutions into specialized hardware configurations and possible operational plans for the well.

4.2 Case 2 production scenario

The Case 2 Olympus section model presents a more complex and realistic production scenario with not only water influx laterally from injectors but also contribution from an aquifer. The production scenario comprises two deviated injectors (INJD15 and INJD16) located in the eastern region of the reservoir, and a long (approximately 2500 m) deviated producer (PRODX2) that traverses the southern part. PRODX2 is placed in the uppermost layers of the reservoir, while both INJD15 and INJD16 penetrate all seven layers. PRODX2's

trajectory traverses 47 grid blocks and two main non-sealing faults while slightly sloping downwards from heel to toe with heel and toe depths at 2034 m and 2058 m, respectively. The toe of the PRODX2 wellbore is located slightly less than 12 m from the oil-water contact. As with Case 1, the multi-segmented well model for the producer in Case 2 consists of eight segments and is constructed such that each segment connects to approximately the same number of grid blocks. Thus, the same ICD parametrization as described for Case 1 applies for Case 2. Permeability, porosity, fault multipliers and other static model data are taken from the upper channelized formation of the Olympus case (realization 37). Figure 2 shows the channelized permeability field and the relative positioning of all the wells.

4.3 Optimization setup for Case 1 and Case 2

In this work, wells are controlled using bottomhole pressure (BHP) without specified target rate limits. Default BHP set-

Fig. 2 Relative positioning of wells (top) and permeability field (bottom) corresponding to Case 2. Grid and static properties taken from Olympus benchmark case, realization 37. The figure shows the fourth layer from top

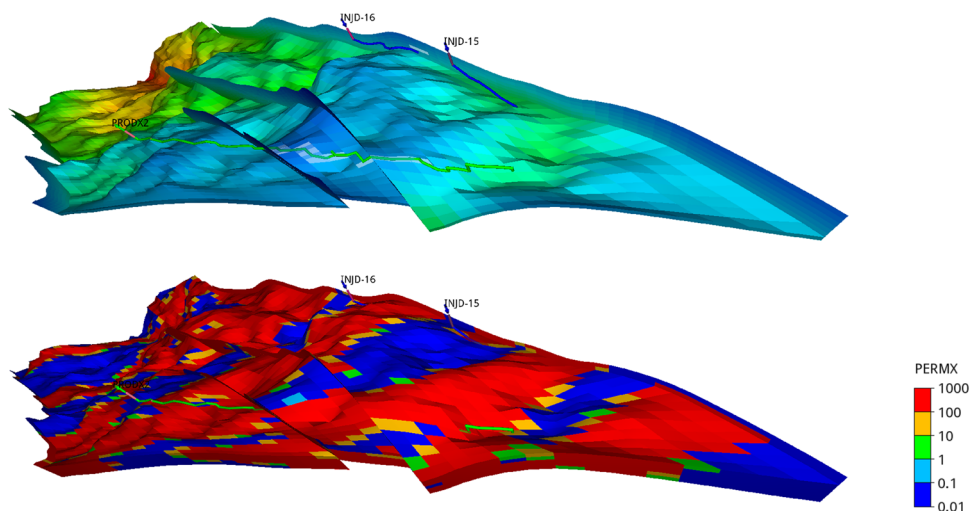


Table 1 Simulation parameters and variable settings for all optimization runs based on Case 1

| | |
|-----------------------------------|---|
| Case 1 simulation parameters | |
| Grid block size | $24\text{ m} \times 24\text{ m} \times 24\text{ m}$ |
| Total # of blocks | 3600 |
| Simulation timeframe | 2190 days |
| Default BHP (PROD1) | 90 bar |
| Default BHP (INJ1, INJ2) | 230 bar |
| # of PROD1 segments | 8 |
| ICD A_x | $3.927E - 5\text{ m}^2$ |
| Optimization settings | |
| # of BHP variables-per-well | 10 |
| BHP bounds (PROD1) [bar] | [70, 150] |
| BHP bounds (INJ-) [bar] | [170, 230] |
| # of ICD variables | 8 |
| ICD A_x bounds [m^2] | $[0, 7.353 \times 10^{-3}]$ |

Total ICD cross-sectional area is denoted as A_x . "INJ-" signifies both injectors in this case

tings, i.e., for those optimization runs where well controls are not treated as variables, are set to constant for the entire production timeframe. For Case 1, default BHP settings are 230 bar for injectors INJ1 and INJ2, and 90 bar for producer PROD1. For Case 2, default BHP settings are 205 bar for injectors INJD15 and INJD16, and 155 bar for producer PRODX2. Whenever well controls are defined as optimization variables, BHPs are parametrized as piecewise-constant functions with 10 control periods, i.e., during optimization each well is associated with 10 control variables. Production timeframes for Case 1 and Case 2 are set to 2190 days and 5470 days, respectively. All simulations are performed using

Table 2 Simulation parameters and variable settings for all optimization runs based on Case 2

| | |
|---|--|
| Case 2 simulation parameters | |
| Grid block size (approx. mean DX, DY, DZ) | $57\text{ m} \times 87\text{ m} \times 3\text{ m}$ |
| Total # of active blocks | 22876 |
| Simulation timeframe | 5470 days |
| Default BHP (PRODX2) | 155 bar |
| Default BHP (INJD15, INJD16) | 205 bar |
| # of PRODX2 segments | 8 |
| ICD A_x | $3.927E - 5\text{ m}^2$ |
| Optimization settings | |
| # of BHP variables-per-well | 10 |
| BHP bounds (PRODX2) [bar] | [105, 175] |
| BHP bounds (INJ-) [bar] | [195, 255] |
| # of ICD variables | 8 |
| ICD A_x bounds [m^2] | $[0, 7.353 \times 10^{-3}]$ |

Total ICD cross-sectional area is denoted as A_x . "INJ-" signifies both injectors in this case

a commercial reservoir simulator. Tables 1 and 2 summarize the relevant simulation settings and optimization parameters in addition to the bounds for the different variable types.

Three problem formulations denoted OptFm.0, OptFm.1 and OptFm.2 are used to perform optimization runs. Case 1 and Case 2 are each solved using these three problem formulations. For each case, OptFm.0 entails optimizing only the eight ICD variables of the producer while keeping all BHP well controls at default levels. OptFm.1 involves both ICD variables and the BHP controls of the producer while keeping constant injector BHPs at default values. Finally, OptFm.2 entails optimizing the eight ICD variables of the producer and the BHPs of both the producer and injectors. Thus, OptFm.1 and OptFm.2 are associated with a total of 18 and 38 control variables, respectively. Table 3 summarizes the types of variables involved in these different problem formulations.

These formulations are used to compare the performance of the different derivative-free search procedures over a range of reasonable problem sizes and possible variable types. Taken together, these formulations study the effect on the ICD optimization when the number of well controls is allowed to increase and provide, to a certain extent, data on the impact of jointly optimizing ICDs and BHP controls.

To expand the performance analysis presented in Section 5, this work introduces a dual presentation of cost function evolution curves. In that section, the cost function data are plotted both with respect to the number of cost function evaluations (n_{fevals}), and also in terms of *runtime-equivalent cost function evaluations*, ($n_{\text{eq.fevals}}$). The total number of $n_{\text{eq.fevals}}$'s performed by a particular optimization run is obtained by dividing its runtime with the time it takes to perform one simulation run (of the corresponding case) using default control values. The complementary $n_{\text{eq.fevals}}$ plots enable additional visual comparison and analysis of cost function evolution for the different procedures in terms proportional to actual runtime. (An analog definition using number of cores instead of simulation runtime was applied in [63] to compare parallelizable procedures like APPS and PSO against inherently serial algorithms like DFTR.) The logic behind the selection of the number of cores for the parallel algorithms is as follows. For PSO, the number of cores equals the number of particles or the swarm size while the number of cores used

Table 3 Variables involved in the problem formulations OptFm.0, OptFm.1 and OptFm.2 applied to Case 1 and Case 2

| Formulation | # of variables (type) | Wells |
|-------------|-----------------------|-------------|
| OptFm.0 | 8 (ICD) | PROD· |
| OptFm.1 | 8 (ICD) + 10 (BHP) | PROD· |
| OptFm.2 | 8 (ICD) + 30 (BHP) | PROD·, INJ· |

"INJ·" signifies both injectors, while "PROD·" signifies producers

Table 4 Settings for optimization procedures APPS, PSO and DFTR applied in OptFm.0 through OptFm.3 runs

| General | | DFTR | |
|---------------------------|------------------------------|------------------------------|----------------------|
| ObjectiveScalingCoeff. | 1.0×10^9 | <i>Max#FunctionEvals.</i> | 200 / 3600 / 3600 |
| APPS | | InitialRadius | 0.3 / 0.075 / 0.165 |
| <i>Max#FunctionEvals.</i> | 1000 / 3600 / 10 800 | <i>CostFunctionTolerance</i> | 1.0×10^{-6} |
| InitialStepLength | 0.25 / 0.35 / 0.25 | EpsilonCriticality | 1.0×10^{-5} |
| <i>MinimumStepLength</i> | 0.005 | {Epsilon0, Epsilon1} | {0, 0.05} |
| ContractionFactor | 0.618 | PivotThreshold | 0.0625 |
| ExpansionFactor | 1.618 | AddThreshold | 100 |
| PSO | | ExchangeThreshold | 1000 |
| <i>#SwarmIterations</i> | 170 / 250 / 308 | RadiusMax | 1.5 |
| <i>SwarmSize</i> | 12 / 27 / 57 | RadiusFactor | 6 |
| Learning{Fac.1, Fac.2} | {2, 2} / {1, 1} / {1.5, 1.5} | <i>RadiusTolerance</i> | 1.0×10^{-5} |
| VelocityScale | 1 | Gamma{Inc., Dec.} | {2, 0.5} |
| | | Crit.{Mu, Beta, Omega} | {100, 10, 0.5} |

Triplet settings correspond to parameters used at each problem configuration, i.e., OptFm.0 / OptFm.1 / OptFm.2. In this work, each function evaluation corresponds to a reservoir simulation. Parameters in *italic* serve as stopping criteria for the particular algorithms. The abbreviations Evals., Coeff., Fac., Inc., Dec., and Crit. stand for Evaluations, Coefficient, Factor, Increase, Decrease, and Criticality, respectively

for APPS is twice the number of unknown variables. The upper bound of 38 cores is set to all the parallel runs as this is the number of cores available in the computer used for the simulations.

For each case, Case 1 and Case 2, and for the different problem formulations, OptFm.0, OptFm.1 and OptFm.2, the APPS, PSO and DFTR search procedures are applied using different settings. These settings have been manually tuned in a systematic manner for each case and formulation. Table 4 lists the settings used in Case 1, and Table 5 lists the settings used in Case 2. Among these settings are scaling coefficients

and the stopping criteria for the three algorithms used in this paper. Notice that these methods have rather different search characteristics, thus they rely on different convergence criteria beyond a general maximum number of function evaluations that apply to all. Note that all variables for all problem formulations are scaled to the order of unity, i.e., all variables lie within the domain $[-1, 1]$ during optimization. This scaling was performed by using a linear transformation of each vector value with its corresponding variable bound (see [64], p.274 for details). Thus, all settings provided in Tables 4 and 5 apply to the scaled variable domain.

Table 5 Settings for optimization procedures APPS, PSO and DFTR applied in OptFm.0 through OptFm.3 runs

| General | | DFTR | |
|---------------------------|--------------------------|------------------------------|--------------------------|
| ObjectiveScalingCoeff. | 1.0×10^9 | <i>Max#FunctionEvals.</i> | 15 000 / 15 000 / 15 000 |
| APPS | | InitialRadius | 0.105 / 0.09 / 0.15 |
| <i>Max#FunctionEvals.</i> | 10 800 / 10 800 / 10 800 | <i>CostFunctionTolerance</i> | 1.0×10^{-6} |
| InitialStepLength: | 0.025 / 0.35 / 0.075 | EpsilonCriticality | 1.0×10^{-5} |
| <i>MinimumStepLength:</i> | 0.001 / 0.005 / 0.005 | {Epsilon0, Epsilon1} | {0, 0.05} |
| ContractionFactor: | 0.618 | PivotThreshold | 0.00625 |
| ExpansionFactor: | 1.618 | AddThreshold | 100 |
| PSO | | ExchangeThreshold | 1000 |
| <i># of iterations</i> | 170 / 250 / 350 | RadiusMax | 1.5 |
| <i>SwarmSize</i> | 12 / 27 / 38 | RadiusFactor | 6 |
| Learning{Fac.1, Fac.2} | {2, 2} / {2, 2} / {2, 2} | <i>RadiusTolerance</i> | 1.0×10^{-5} |
| VelocityScale | 1 | Gamma{Inc., Dec.} | {2, 0.5} |
| | | Crit.{Mu, Beta, Omega} | {100, 10, 0.5} |

Triplet settings correspond to parameters used at each problem configuration, i.e., OptFm.0 / OptFm.1 / OptFm.2

5 Results

Results corresponding to Case 1 and Case 2 are discussed next. Results for the different approaches with tuned algorithm settings are presented for the simple model of Case 1 before expanding model complexity in Case 2. For each case, results and analysis are organized according to the specified problem formulations (OptFm.0, OptFm.1 and OptFm.2; see Table 3). Solving for these various formulations enables analysis regarding problem dimension and variable types.

5.1 Case 1 OptFm.0 results

Figure 3 and Table 6 summarize search performance for Case 1 OptFm.0. Figure 3 presents the corresponding cost function performance profiles, and Table 6 provides optimal cost function values, relative percentage increases compared

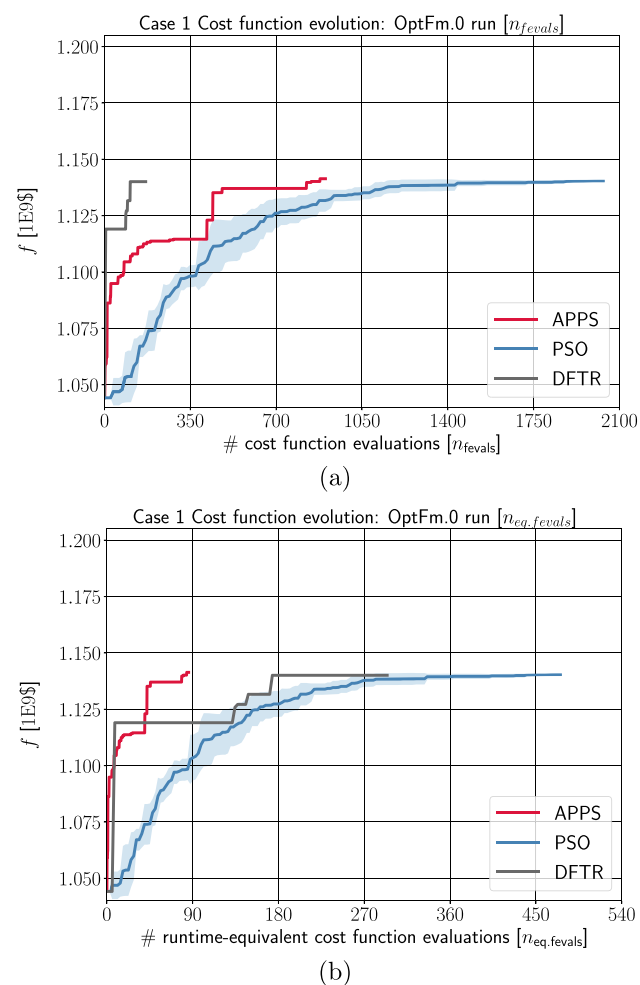


Fig. 3 Case 1 OptFm.0 cost function evolution curves for APPS, PSO and DFTR runs; plotted in terms of total number of cost function evaluations, n_{fevals} , (above) and runtime-equivalent cost function evaluations, $n_{eq.fevals}$, (below)

Table 6 Case 1 OptFm.0 results

| Proc. | $f^*[1 \times 10^9\$]$ | $\Delta[\%]$ | n_{fevals} | $t_{opt}[s]$ | n_c |
|-------|------------------------|--------------|--------------|--------------|-------|
| APPS | 1.141 | 9.3 | 901 | 945 | 16 |
| PSO | 1.140 (± 0.001) | 9.2 | 2040 | 5280 | 12 |
| DFTR | 1.140 | 9.2 | 168 | 3265 | 1 |

f^* and $\Delta[\%]$ represent best cost function values and percentage increase with respect to default control configuration for this case, respectively. For each optimization run, n_{fevals} := total number of cost function evaluations; t_{opt} := total runtime in seconds; and n_c := number of cores used in parallel by each procedure. The f^* value for PSO is the mean over six optimizations runs (standard deviation given in parenthesis). Best PSO run yields $f_{best}^* = 1.141$

to base case objective value (obtained using default initial controls), as well as the total number of cost function evaluations performed, final runtimes and the number of cores used for parallelization. Additionally, Fig. 4 shows the control solution for each algorithm (for PSO, the best solution from its set of runs is used), as well as resulting oil and water saturation maps at end of production time.

Figure 3(a) and (b) show APPS, PSO and DFTR cost function evolution curves plotted with respect to n_{fevals} and $n_{eq.fevals}$, respectively. These curves arrive at practically the same final value and yield close-to the same relative increase in the objective, as can be confirmed by the first two columns in Table 6. Figure 3(a) shows a very efficient DFTR progression in terms of n_{fevals} (< 200), while both APPS and PSO require substantially more sampling (~ 500 , > 1500) before approaching convergence. Table 6 shows APPS requires in total 901 function evaluations compared to 168 by DFTR. However, Fig. 3(b) shows APPS converging more than three times faster than DFTR if APPS is deployed with full parallelization over 16 cores (APPS has a 16-point sampling stencil in this eight-variable formulation).

PSO runs for this case spend more than double the number of cost function evaluations compared to APPS and 60% longer time to converge compared to DFTR. Note, however, that PSO runs in this work converge only due to a pre-set maximum number of swarm iterations. Multiple combinations of swarm size and total number of iterations were tested for this specific case and the particular combination presented in Table 4 consistently provided best performance over all runs, e.g., at least a swarm size 1.5 times larger than problem dimension was needed for none of the swarms to collapse during optimization. Thus, Case 1 OptFm.0 is solved using a swarm size of 12 over 170 iterations, which yields a total number of function evaluations of 2040 (see Table 4 for additional parameters). A total of six runs were performed because of the stochastic nature of the algorithm, and all runs were fully parallelized, i.e., the number of cores used equals the swarm size. Result columns $PSO:n_{fevals}$ and $PSO:t_{opt}$ in Table 6 confirm the swarm-based search of PSO requires

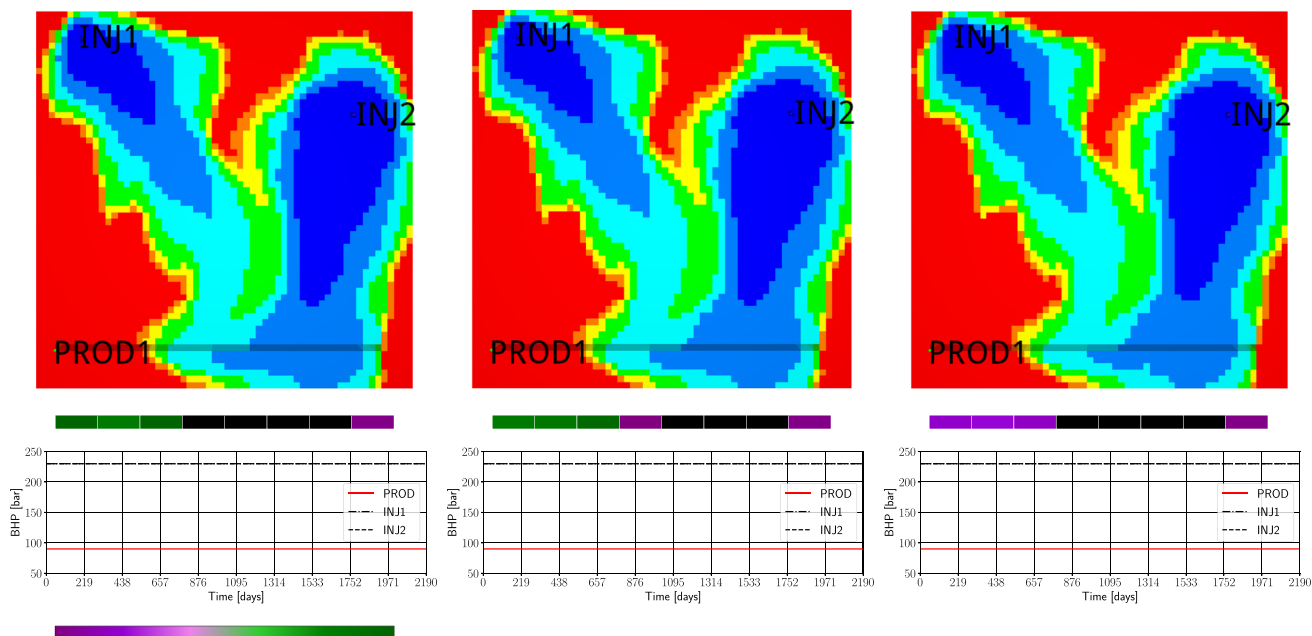


Fig. 4 Optimal controls and corresponding oil-water saturation maps at end production time for Case 1 OptFm.0. Results: APPS (left), PSO (center) and DFTR (right). Oil is indicated by red and water by blue in the saturation map. The PSO results shown correspond to the run with highest f^* from the PSO set. Colorbar representation of indi-

vidual ICD openings: dark green (fully open, i.e., upper bound), dark purple (close to shut, i.e., almost lower bound), black (fully closed, i.e., lower bound); optimization bounds given in Table 1. The same colorbar applies to Figs. 6 and 8. Fixed BHP settings are provided for reference

substantial computational resources. The runtime efficiency of APPS compared to DFTR and PSO for this particular case is reflected in Fig. 3(b).

A comment regarding algorithm- and case-specific convergence criteria is appropriate at this point. Note that while all methods operate under a standard limit on maximum number of function evaluations (explicitly defined by `Max#FunctionEvaluations` for APPS and DFTR, and equal to `#SwarmIterations` times `SwarmSize` for PSO, see Table 4), additional algorithm-specific criteria apply for APPS and DFTR. In this work, APPS stops once its stencil size is less than the `MinimumStepLength` ($\propto 1.0 \times 10^{-3}$) while DFTR operation is subject to both a `RadiusTolerance` (1.0×10^{-5}) and a `CostFunctionTolerance` (1.0×10^{-6}).

Though these algorithm-specific convergence criteria may make comparisons more difficult and nuanced, their enforcement reflects the search mode and accuracy required by the respective approaches. As an example, recall that while DFTR explicitly approximates the objective through interpolation, APPS relies on direct comparison of cost function sample points, thus only implicitly, and in a comparatively coarser manner, capturing cost function shape. Given the different modes of operation by the different algorithms, the implemented tolerances should indeed be tuned according to what is strictly necessary and sufficient for the particular approach to conduct an efficient search in the given case.

Based on an extensive set of additional runs (equal to those reported here, for both cases), we found that convergence criteria other than the ones reported in Table 4 for DFTR, APPS and PSO algorithms resulted in less efficient searches, e.g., premature convergence or long plateaus. Thus, for all methods in all cases, the algorithm parameters and results reported in this work are those corresponding to the best-performing runs in terms of the highest final objective and sufficient number of iterations. This configuration enables a comparison based primarily on the core search properties of the algorithms, since each run is close to the best result possibly achieved given that particular algorithm for that particular case.

With this in mind, we further discuss a previous result to show that reported core search performance remains largely consistent despite some relative influence algorithm-specific criteria can have on convergence. As presented earlier, Table 6 shows $\text{APPS}:t_{\text{opt}}$ less than three times $\text{DFTR}:t_{\text{opt}}$. Clearly, this factor would decrease if APPS was encouraged to produce a similar convergence plateau as the other runs. However, even assuming the APPS plateau continues for, say, 200 more function evaluations under a tighter step size tolerance, and that this yields a proportional time increase of roughly $\text{APPS}:t_{\text{opt}} \approx 1200\text{ s}$, the speed-up difference, i.e., $\text{APPS}:t_{\text{opt}} \approx 1200\text{ s}$ versus $\text{DFTR}:t_{\text{opt}} \approx 3200\text{ s}$ would still remain substantial. We will further consider differing search

properties and convergence criteria when analyzing subsequent results in the following sections.

The search efficiency demonstrated by DFTR for this simple case shows the value of creating accurate models through careful sampling. However, it can also be seen as a reflection that the objective function is easier to approximate for a low-dimensional problem with only one type of variable. The inclusion of other variable types with different degree of influence on the objective (further discussed below) will likely make the cost function more challenging to approximate. (The next section, Case 1 OptFm.1 results, further explores this search feature contingent on problem size increases and the inclusion of different variable types.)

Finally, Fig. 4 shows the actual ICD solutions and corresponding saturation maps at the end of the production timeframe. The well segments are represented by eight horizontal rectangles placed below the saturation maps and PROD1. The coloring of each of the rectangles represents the total cross-sectional value of the ICD in that particular segment. The coloring varies from dark-green (fully open, i.e., upper bound) to light green then light purple to dark purple (almost closed, i.e., close to lower bound) to black (fully closed). All three solutions yield a similar completion design that corresponds to the permeability field of the case, i.e., high-permeability zones are completely shut down, while segments near the PROD1 heel located in low-permeability zones, in addition to the toe segment, both remain open to varying degree. The final saturation maps are thus also very similar with only slight variation in the shape of the remaining oil located between the injectors. Overall, these visualizations confirm the expectation of what the applications should produce, i.e., a closing of high water-cut areas.

5.2 Case 1 OptFm.1 results

The Case 1 OptFm.1 formulation not only optimizes ICD settings, but also involves BHP variables for PROD1, i.e., the problem is a joint optimization problem consisting of fixed-in-time ICDs and piecewise-constant time-varying BHP variables. Note we use the "joint" term here to refer to problem formulations comprising multiple variable types, e.g., well placement, production controls and/or configuration of inflow control devices. Joint problems may be solved using various strategies, e.g., simultaneous, sequential or embedded. This work implements a simultaneous solution approach, i.e., at each iteration, each method operates on all components of the variable vector.

A particular feature of the Case 1 OptFm.1 formulation is that the two variable types affect the magnitude of the objective to different degree. Given their physical implementation, a change in a particular ICD variable affects the entire production timeframe, while a change in any particular BHP variable

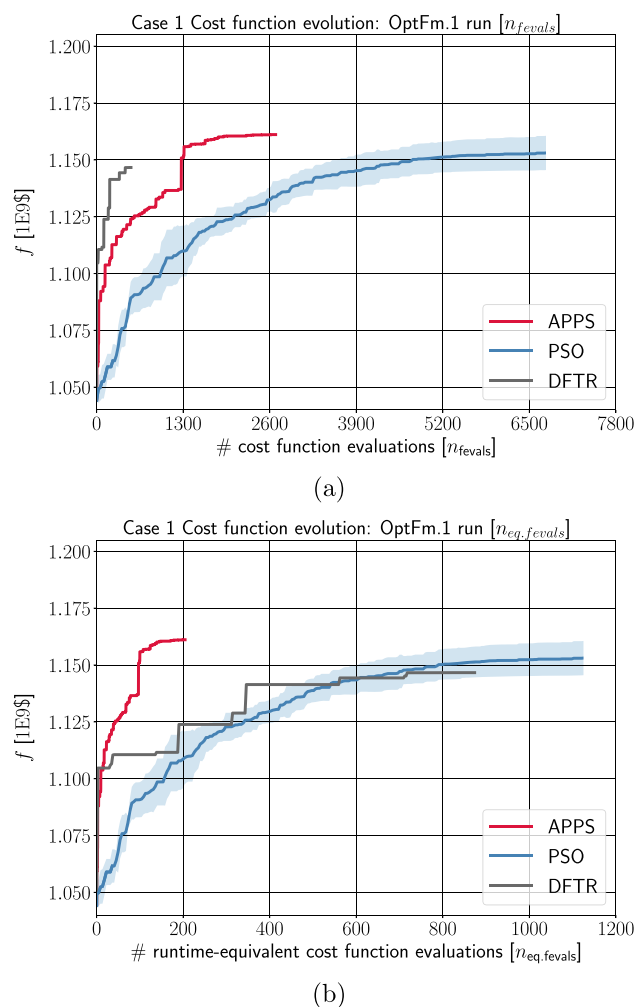


Fig. 5 Case 1 OptFm.1 cost function evolution curves for APPS, PSO and DFTR runs

influences the objective primarily through production during the corresponding control period. (An analog relationship is similarly found in other joint problems, e.g., when optimizing both well trajectories and controls.) Due to the problem structure, this feature remains despite the different variable types being scaled onto the $[-1, 1]$ domain using their corresponding bounds (see the end of Section 4.3).

Complementary performance results are provided by Fig. 5 and Table 7, while Fig. 6 shows saturation maps and

Table 7 Results for OptFm.1

| Proc. | $f^*[1 \times 10^9\$]$ | Δ [%] | n_{fevals} | t_{opt} [s] | n_c |
|-------|------------------------|--------------|--------------|---------------|-------|
| APPS | 1.161 | 11.2 | 2689 | 2250 | 36 |
| PSO | 1.153 (± 0.008) | 10.4 | 6750 | 12 426 | 27 |
| DFTR | 1.147 | 9.9 | 516 | 9673 | 1 |

Column descriptions provided in Table 6. Best PSO run yields $f_{best}^* = 1.158$

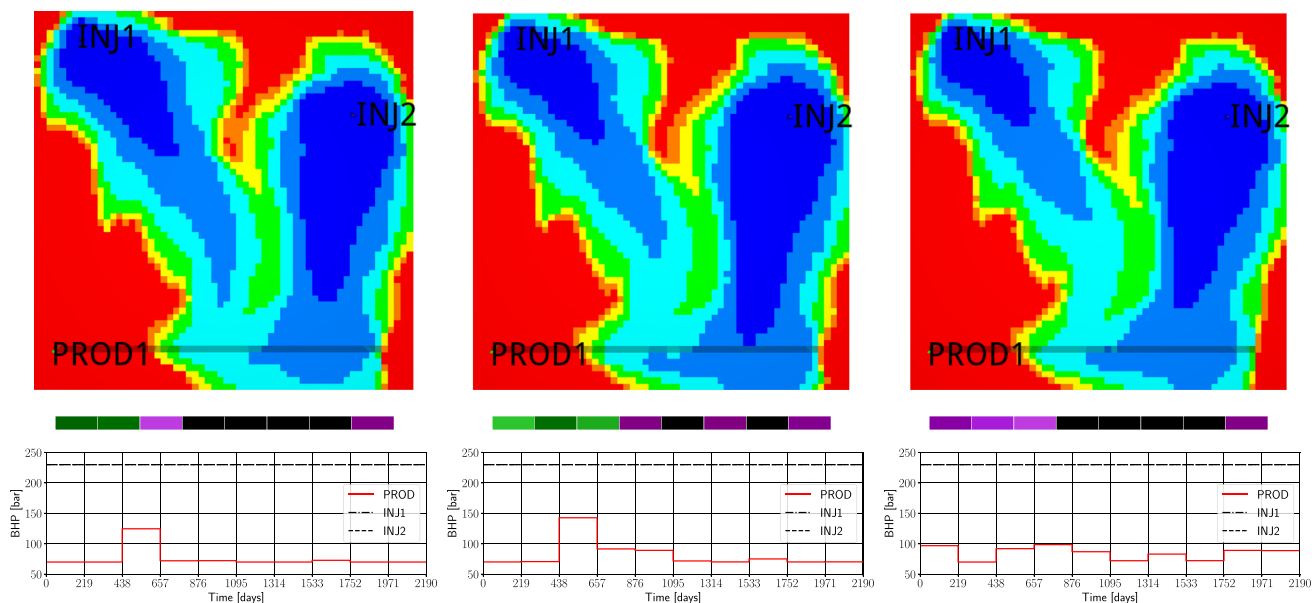


Fig. 6 Optimal controls and corresponding oil-water saturation maps at end production time for Case 1 OptFm.1. Results: APPS (left), PSO (center) and DFTR (right). The PSO results shown correspond to the run with highest f^* from the PSO set

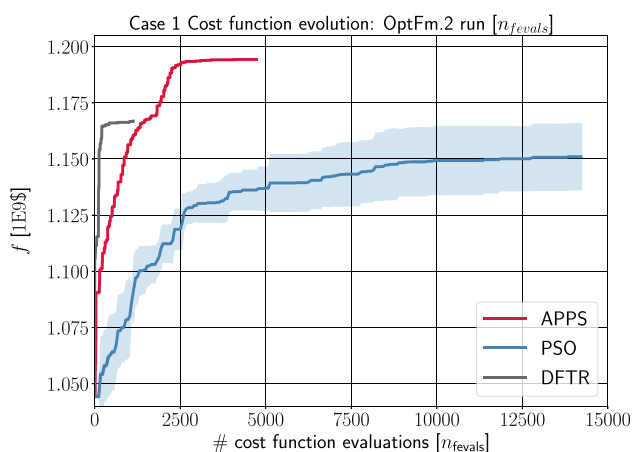
final ICD and BHP solutions for the corresponding search approaches. Results from Case 1 OptFm.1 are compared with the previous OptFm.0 results to study changes in algorithm performance due to increased problem dimension and limited expansion of variable structure. In particular, derivative-free performance is directly related to problem size and close-to discontinuous cost function sensitivity may impact individual search characteristics. Finally, overall performance is studied in terms of solution and objective value, e.g., differences in final configuration and cost function gains due to solving the problem jointly with well controls are explored.

Comparing f -increase columns (Δ) from Tables 6 and 7 shows individual APPS, PSO and DFTR increments of 1.9, 1.2 and 0.7%, respectively. For each algorithm, these increments correspond to relative objective function gains of 20, 13 and 8% attributed to the greater flexibility enabled by the OptFm.0 to OptFm.1 problem expansion. Notably, irrespective of the algorithm, the OptFm.1 formulation requires approximately three times the number of cost function evaluations used to solve for OptFm.0. In a similar manner, both APPS and PSO total runtimes are 2.4 times longer while the DFTR runtime is three times longer compared to the OptFm.0 runs. In terms of the number of cost function evaluations, Fig. 5(a) shows a very efficient local search by DFTR even though both APPS and PSO eventually reach higher values. If considering performance in terms relative to runtime, however, Fig. 5(b) shows an inverse situation where APPS conducts a very efficient search compared to DFTR.

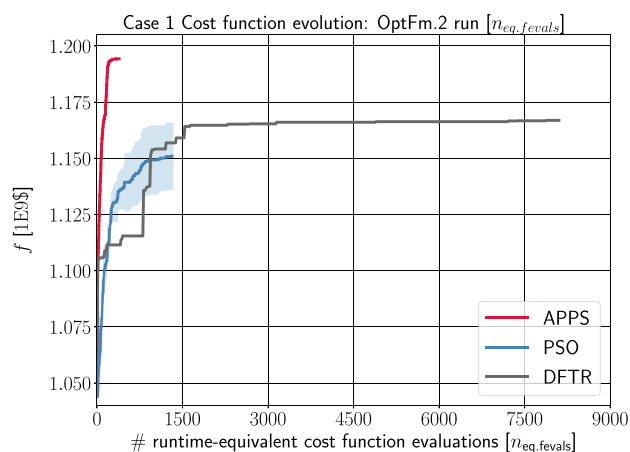
The expansion from OptFm.0 to OptFm.1 entitles increases in both problem dimension and cost function complexity. Increased complexity may complicate DFTR's characteris-

tic model-building routine by forcing reduced model sizes, thus placing greater importance on the initial point and likely leading to less effective searches. Compared to Fig. 3, Fig. 5 shows a markedly poorer f -evolution by DFTR relative to the performance obtained by the other search approaches. In fact, as shown in Table 4, DFTR for OptFm.1 provided the best result when started with a lower "InitialRadius" setting of 0.075 compared to using 0.3 when solving OptFm.0. (Note that for each formulation, DFTR is tested using 12 different InitialRadius settings; thereafter, the run with the highest f value is selected for comparison, and its setting reported in Tables 4 and 5.) Taken together, the OptFm.0–OptFm.1 problem expansion demonstrates DFTR maintains its high local search efficiency while using only very few sampling points. However, its overall exploration capacity is likely to further degrade as search space complexity increases. Subsequent sections will further discuss this point as results from more advanced cases are analyzed.

PSO runs for both OptFm.0 and OptFm.1 problem formulations yield gradually increasing mean f -evolutions with low standard deviation throughout the optimization (Figs. 3 and 5, respectively). Note that for each formulation, the various PSO search parameters are tuned such that the combination requiring lowest total n_{fevals} while consistently providing successful progression with high f^* is chosen. The tuning entitles finding a sufficient number of iterations and population size to both ensure convergence and avoid population collapse for all runs in each formulation. The chosen parameter combinations are reported in Table 4. In relative terms and despite specific tuning, the PSO applications require substantial computational resources both in terms



(a)



(b)

Fig. 7 Case 1 OptFm.2 cost function evolution curves for APPS, PSO and DFTR runs

of n_{fevals} and $n_{\text{eq.fevals}}$ to solve either problem formulation. Finally, for these applications, not even full parallelization (# of particles = # of parallel computing cores) is able to reduce runtime sufficiently to make PSO performance comparable to the most efficient approach, i.e., APPS.

Table 7 (second column) provides f^* performance of the different algorithms for OptFm.1 while Fig. 4 shows corresponding saturation maps and control solutions. PSO solution results in Fig. 4 (center) correspond to the run with highest f^* ($f^* = 1.158$) from the PSO set. Figure 4 shows the PSO ICD and BHP controls are very similar to the APPS controls that yield a $f^* = 1.161$. Associated saturation maps for PSO and APPS are therefore very similar. Comparatively, however, the DFTR map shows substantially less water intrusion from both injectors even though the DFTR BHP profile is somewhat more aggressive compared to the other two solutions (especially after 438 days). In this case, the APPS and PSO ICD solutions enable a higher degree of water encroachment

Table 8 Results for OptFm.2

| Proc. | $f^*[1 \times 10^9\$]$ | $\Delta[\%]$ | n_{fevals} | $t_{\text{opt}}[\text{s}]$ | n_c |
|-------|------------------------|--------------|---------------------|----------------------------|-------|
| APPS | 1.194 | 14.4 | 4733 | 4276 | 38 |
| PSO | 1.151 (± 0.016) | 10.2 | 14 250 | 14 178 | 38 |
| DFTR | 1.167 | 11.8 | 1129 | 89 407 | 1 |

Column descriptions provided in Table 6. Best PSO run yields $f_{\text{best}}^* = 1.167$

by deploying configurations with significantly larger valve openings (from dark to light green).

5.3 Case 1 OptFm.2 results

Case 1 OptFm.2 engages all controls in the problem scope by also treating the injectors' (INJ-) BHP settings as variables. This more than doubles the dimension of the previous formulation to a total of 38 (variable numbers and types for all problem formulations are provided in Table 3). Compared to previous formulations, the additional flexibility enabled by OptFm.2 is expected to facilitate higher final objective values for all derivative-free approaches. However, due to their reliance on different search characteristics, the higher number of variables in this particular formulation can significantly affect the relative computational performance of the approaches. (Compared to gradient-based methods, the performance of derivative-free approaches is much more dependent on the total number of variables.) Finally, the inclusion of additional BHP controls may serve to balance cost function sensitivity associated with the different variable types, and thereby indirectly benefit computational performance by providing a smoothing effect on the objective. Thus, compared to OptFm.1 comprising 8 ICD and 10 BHP variables, the OptFm.2 formulation now involves 8 ICD and 30 BHP controls, which increases the relative influence BHP controls can exert on the overall cost function. In this sense, the optimization may reach higher objective values not only because a greater number of possible configurations are available but also because the search process itself is less dominated by a few (ICD) dimensions.

Figure 7 and Table 8 show the search results for Case 1 OptFm.2 while Fig. 8 shows the saturation maps and optimal ICD and BHP controls corresponding to each approach. It is useful to compare the saturation maps of OptFm.2 (Fig. 8) with those of OptFm.1 (Fig. 6) to illustrate the effect of involving INJ- BHP controls in the optimization. For all solutions, the dark-blue water area from INJ2 has a more compact encroachment on the toe half of PROD1 when using OptFm.2 controls compared to OptFm.1 solutions. On the other hand, the INJ1 injection when using OptFm.2 controls extends the dark-blue areas closer towards the middle of PROD1 while also further pushing water onto the curved pocket of oil

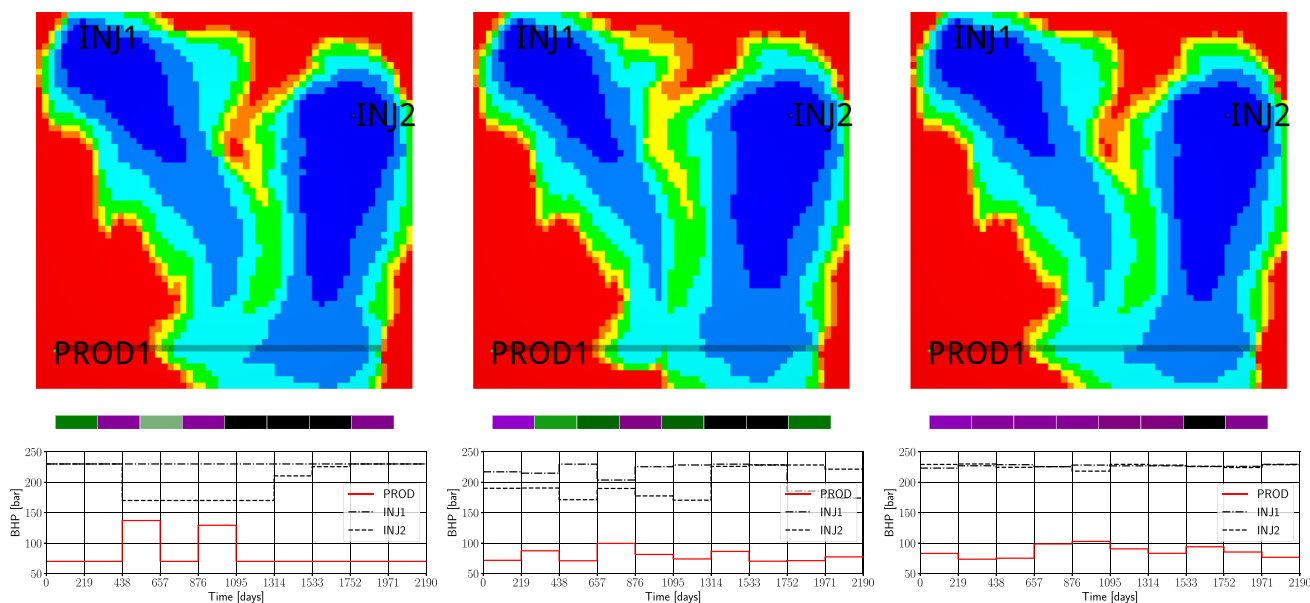


Fig. 8 Optimal controls and corresponding oil-water saturation maps at end production time for Case 1 OptFm.2. Results: APPS (left), PSO (center) and DFTR (right). The PSO results shown correspond to the run with highest f^* from the PSO set

located north of the reservoir. Figure 9 illustrates this redistribution by showing the saturation difference for the APPS solution with increases in oil and water saturation drawn as black and white areas, respectively. Finally, it should be noted that each OptFm.2 solution has an ICD configuration with one more open compartment (to a varying degree) compared to its corresponding OptFm.1 solution. Thus, compared to previous formulations, it appears optimizing using additional (INJ-) controls enables wells with extended reservoir interfaces, and that the higher number of controls provides a measure of flexibility to optimally adjust the resulting fluid displacement.

The corresponding performance data for the different search approaches are shown in Table 8. For this formulation, APPS requires 4733 function evaluations compared to

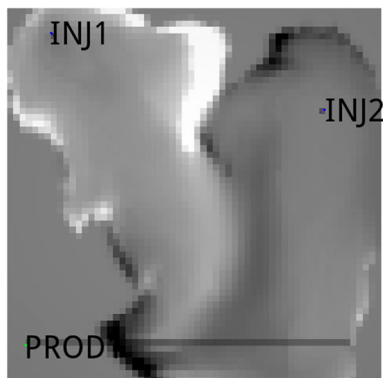


Fig. 9 Oil (black) - water (white) saturation map difference between Case 1 OptFm.1 and OptFm.2 APPS solutions at end production time

1129 by DFTR and 14250 by PSO. However, as Fig. 7(b) also shows, APPS finished more than 3 times faster than PSO and 20 times faster than DFTR. Figure 7(a), on the other hand, presents an efficient DFTR progression in terms of sheer number of cost function evaluations. In particular, for the DFTR algorithm, its plateau convergence phase requires a significant amount of time, which may be due to increased algorithm processing, e.g., substantial model rebuilding efforts, added to the actual time used to run the simulations themselves. Recall that the model-building part of DFTR is particularly dependent on problem dimension (see Section 3.3.3). Finally, for this formulation, PSO conducts an inefficient search even though substantial computational resources are provided. Further, there is a large deviation between the individual PSO runs, as visualised in Fig. 7. PSO yields a mean $f^* = 1.151$ (while the best result from the PSO set is $f^* = 1.167$, equal to the DFTR results).

Lastly, an additional note is made regarding the ICD results obtained from the different approaches across the three problem formulations for this case. Comparing ICD solutions from Figs. 4, 6 and 8, we note that, although there is a spread in magnitude among the open ICD settings, there is a tendency of fewer fully closed segments when increasing the number of BHP controls, i.e., when going from formulations OptFm.0 to OptFm.2. (Beyond this note, however, too few data points are available to make further conclusions.)

To conclude, overall results from Tables 6 to 8 demonstrate the gains obtained with more expansive problem formulations while detailing the resources needed to achieve these

gains for the different search approaches. As part of an area or field development project, these data can support a decision situation concerned with finding an effective problem formulation and configuring an application of optimization routines that is likely to meet expectations, i.e., this type of data provides information regarding possible alternatives for an optimization campaign, e.g., with regard to problem formulation, search strategy choice, method parametrization, etc. Moreover, within an established decision structure, these results can be used to assess probabilistic returns in terms of likely gain and performance for the various search approaches, given the particular problem specifications and computational budget. The next section presents the data for the more realistic Case 2 and expands on the analysis of the approaches performed so far for Case 1.

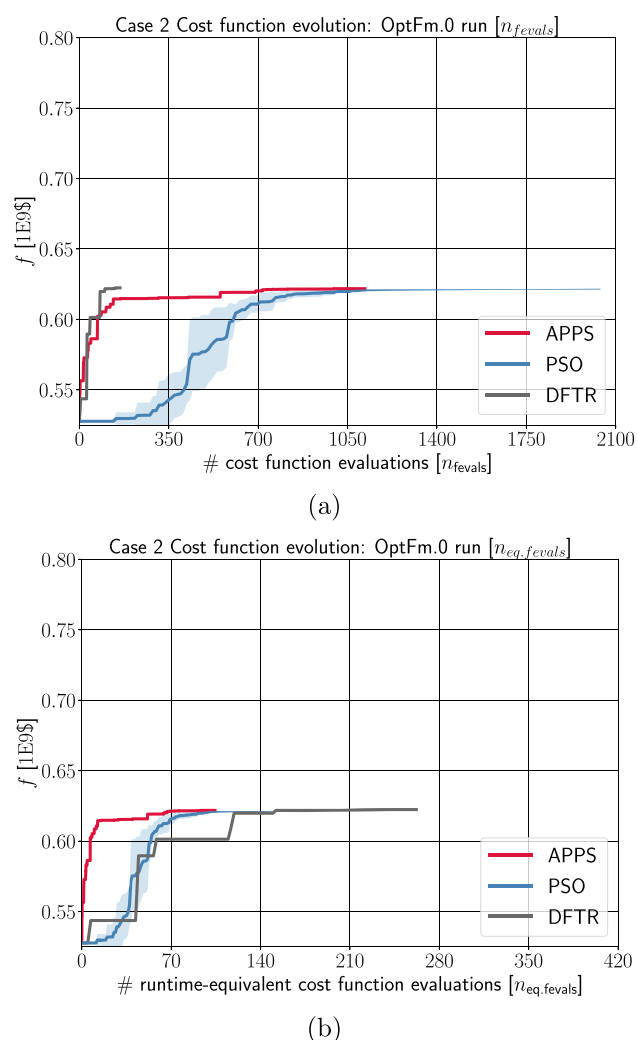


Fig. 10 Case 2 OptFm.0 cost function evolution curves for APPS, PSO and DFTR runs; curves plotted in terms of total number of cost function evaluations, n_{fevals} , (above) and runtime-equivalent cost function evaluations, $n_{eq.fevals}$, (below), analogous to Fig. 3

Table 9 Case 2 OptFm.0 results

| Proc. | $f^*[1 \times 10^9\$]$ | $\Delta[\%]$ | n_{fevals} | $t_{opt}[s]$ | n_c |
|-------|------------------------|--------------|--------------|-------------------|-------|
| APPS | 0.622 | 17.9 | 1118 | 2.3×10^4 | 16 |
| PSO | 0.622 (± 0.000) | 17.9 | 2040 | 4.0×10^4 | 12 |
| DFTR | 0.622 | 17.9 | 159 | 5.8×10^4 | 1 |

f^* and $\Delta[\%]$ represent best cost function values and percentage increase with respect to default control configuration for this case, respectively. Analog column configuration as Table 6. Best PSO run yields $f_{best}^* = 0.622$

5.4 Case 2 OptFm.0 results

The main goal of Case 2 is to extend the performance analysis of the different approaches to a problem with increased model complexity. Figure 10 and Table 9 summarize the search performance for the Case 2 OptFm.0 formulation. Figure 10(a) shows the three approaches reaching the same f^* with different progression paths. For this case, both DFTR and APPS show very efficient initial progressions in terms of n_{fevals} ; however, APPS requires a substantial plateau phase before converging, which leads to a total $n_{fevals} \sim 1100$ compared to $n_{fevals} < 200$ for DFTR. However, Fig. 10(b) shows that using full parallel capacity for this case, i.e., $n_c = \text{stencil size}$, APPS converges close to twice as fast as PSO and ~ 2.5 times faster than DFTR.

Figure 11 shows ICD solutions (fixed BHP settings are included for later reference) and corresponding saturation maps at the end of the production timeframe. All saturation maps for Case 2 are presented with a horizontal slice positioned at layer 4. The saturation maps in general reflect the similarity of final cost function values presented by Table 9. The APPS map, however, shows a somewhat greater water sweep close to the PRODX2 wellbore in the area between the two faults. Similar to Case 1, all ICD solutions show a complete shut-in of segments in the toe-half of PRODX2, which corresponds to the high-permeability channel in this region (shown in Fig. 2).

5.5 Case 2 OptFm.1 results

Performance results for Case 2 OptFm.1 are presented by Fig. 12 and Table 10, while Fig. 13 shows final ICD and BHP solutions and corresponding saturation maps for the different search approaches. As for Case 1, Case 2 OptFm.1 signifies an expansion in problem formulation to also involve producer BHP controls as optimization variables. For this formulation, Fig. 13 shows that, except for a couple of control periods, both APPS and PSO solutions mainly operate PRODX2 using BHP controls set at the lower bound. Furthermore, compared to Case 2 OptFm.0, the corresponding ICD solutions have several more shut-in segments in the toe-

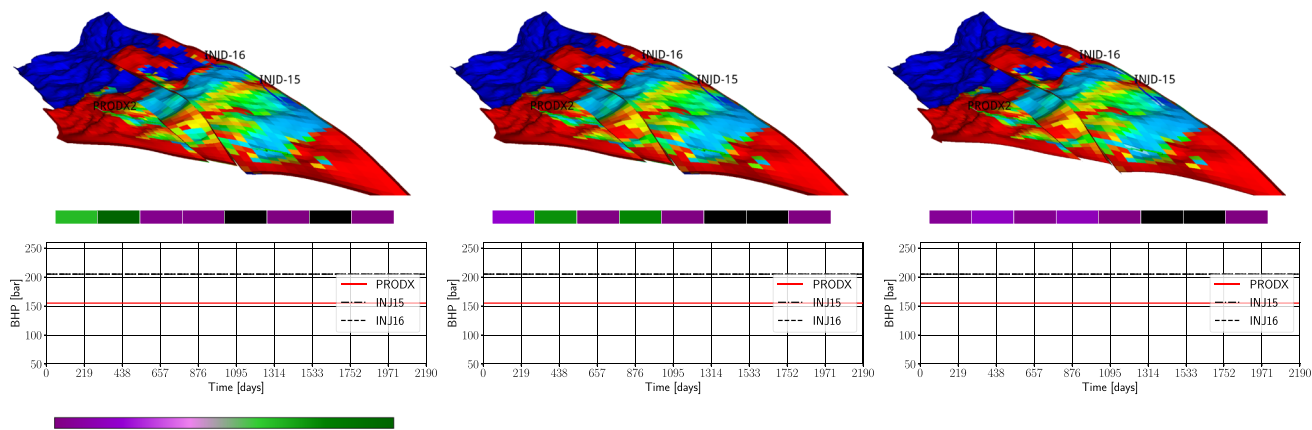


Fig. 11 Optimal ICD controls and corresponding saturation maps for Case 2 OptFm.0. APPS, PSO and DFTR saturation maps showing layer 4 on top, with red indicating oil and blue indicating water. Colorbar representation of individual ICD openings: dark green (fully open, i.e.,

upper bound), dark purple (close to shut, i.e., almost lower bound), black (fully closed, i.e., lower bound); optimization bounds given in Table 2. The same colorbar applies to Figs. 13 and 15. Fixed BHP settings are provided for reference

half of PRODX2. Altogether, this yields a more aggressive control strategy for this particular case that results in significantly higher objective increases for APPS and PSO as seen in Fig. 12. Table 10 confirms these additional increases of 23.3% and 22.1% for APPS and PSO, respectively, compared to the Case 1 OptFm.0 results. These results point to the importance of involving both ICD and BHP variables during optimization.

DFTR solutions for OptFm.0 and OptFm.1 formulations are very alike and OptFm.1 compared to OptFm.0 yields an additional increase of 3% only, as shown by Table 10. (These two solutions are also similar to the OptFm.2 solution discussed later which also yields a relatively small increment of about 8% compared to OptFm.0.) Mirroring these results, Fig. 12 presents a low-performing DFTR search compared to the other approaches. (A similar poor progression is also seen for OptFm.2 in Fig. 14.)

5.6 Case 2 OptFm.2 results

Finally, Fig. 14 and Table 11 summarize the Case 2 OptFm.2 results while Fig. 15 shows the corresponding saturation maps and optimal ICD and BHP controls. For all approaches, Fig. 15 shows significant control activity for both injectors and somewhat stable control settings for PRODX2. However, APPS and PSO solutions present both higher injection pressures in general and practically operate PRODX2 at its lower bound pressure settings, thus ensuring an aggressive production strategy. For APPS, this production setting is countered by fewer ICD segments open for production compared to the OptFm.1 solution. Finally, the APPS saturation map shows a markedly improved waterflooding sweep compared to PSO and DFTR maps.

Of the different methods tested in this work, the DFTR search approach is the most sensitive to local cost function properties. The generally inefficient cost function evolutions and associated DFTR results described above indicate increased search difficulties across the various formulations for this particular case, also despite significant problem-specific tuning of method parameters. Overall, for all formulations, these more demanding search conditions can be reasonably attributed to cost function response surfaces harder to traverse due to Case 2's more realistic physical model.

On a more granular level, DFTR reparametrizations required to improve problem-specific performance may be used to infer some properties and possibly changes in cost function response surface across formulations. (Recall that for each formulation, DFTR has been run with a dozen different initial radii and that the configuration and solution associated with the highest f is reported.) In this regard, the reported "best" DFTR InitialRadius parameters (Table 5) can be compared across the various formulations to probe problem properties. As seen for Case 2 OptFm.0, Fig. 10 and Table 9 show DFTR matching the performance of the other approaches. However, compared to Case 1, this result required a substantial reduction in the initial search radius (to about a third, from 0.3 to 0.105). Because of the simplicity of the OptFm.0 formulation, this re-parametrization can be attributed to less-favorable response surface features caused by the underlying simulation model, as discussed above.

For OptFm.1, the best-performing DFTR runs entitle a reduction in InitialRadius in both Case 1 and Case 2 (to 0.075 and 0.09, respectively). However, for Case 2, this reduction is no longer sufficient to conduct an overall efficient search compared to the other routines. Finally, for OptFm.2, DFTR is again able to successfully initiate its search using a relatively

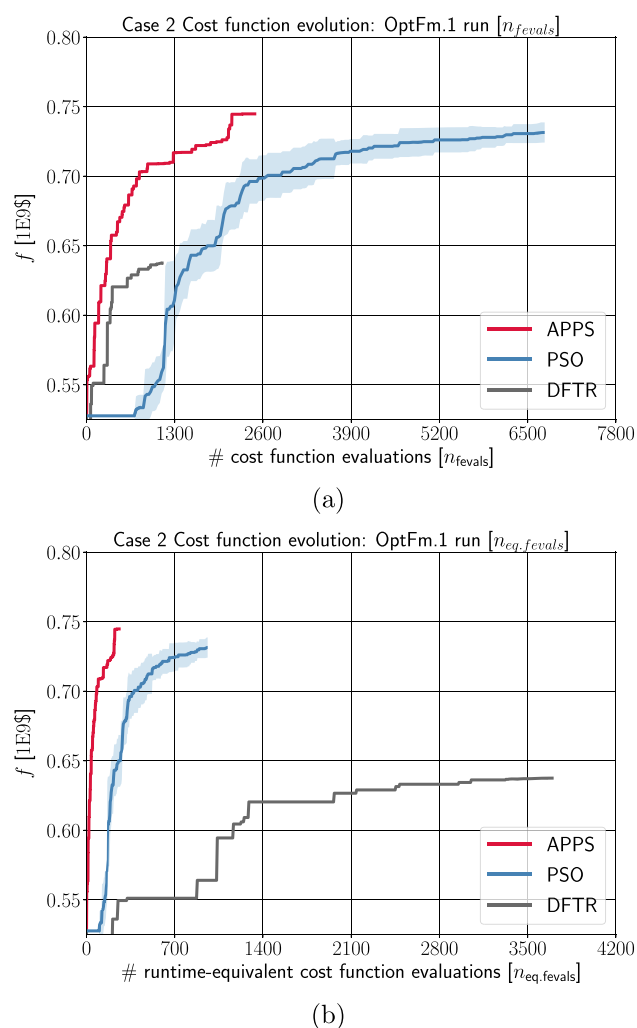


Fig. 12 Case 2 OptFm.1 cost function evolution curves for APPS, PSO and DFTR runs

large initial radius (0.165 and 0.150 for Case 1 and Case 2, respectively), although DFTR's overall Case 2 performance for this formulation yields a poor final result.

Altogether, the occurrence of InitialRadius changes across both cases may be an indication of added challenging conditions presumably caused by the particular variable type configurations of the different formulations. (This function feature was first mentioned at the end of Section 5.2 and fur-

Table 10 Case 2 OptFm.1 results

| Proc. | f^* [1×10^9 \$] | Δ [%] | n_{fevals} | t_{opt} [s] | n_c |
|-------|-----------------------------|--------------|--------------|-------------------|-------|
| APPS | 0.745 | 41.2 | 2479 | 5.6×10^4 | 36 |
| PSO | 0.731 (± 0.008) | 38.5 | 6750 | 2.1×10^5 | 27 |
| DFTR | 0.638 | 20.9 | 1114 | 8.0×10^5 | 1 |

Analog column configuration as Table 7. Best PSO run yields $f_{best}^* = 0.739$

ther discussed at the start of Section 5.3.) If so, the above observation may point to some degree of general smoothening of f caused to a higher ratio of BHP versus ICD variables for OptFm.2 compared to OptFm.1. Such improved conditioning could subsequently enable a local search routine similar to DFTR to start with larger initial radii (or step).

Results from Table 11 show robust improvements in objective (50.7%, 46.9% and 25.5% for APPS, PSO and DFTR, respectively). The corresponding cost function evolution curves presented in Fig. 14 show comparably effective searches for both APPS and PSO in terms of obtaining a similarly high objective value for this problem. Of these two approaches, APPS delivers the most efficient cost function evolution from the start and reaches a convergence plateau using fewer objective evaluations. (Note that PSO for this problem formulation required 350 population iterations to produce a reasonable convergence curve compared to 250 for OptFm.1.)

6 Final remarks

A fundamental challenge in industry algorithm research and development, and subsequent application efforts, is to successfully design and execute fit-for-purpose optimization campaigns. A fit-for-purpose approach is needed because possible area and field development operations to be optimized involve problems with distinct variable types and different requirements for model fidelity. Simply speaking, one particular search approach cannot possibly solve all problem configurations equally well.

The design phase for the development of tailored search methodologies requires scoping of appropriate problem formulations and the definition of practical variable abstractions. In this work, we have tested a range of increasingly complex problem formulations in terms of both problem size and variable type. Given that the tested methods rely on different search characteristics, the study focused on how the increasing number and the composition of different types of variables affect the relative computational performance of the approaches. Finally, an ICD parametrization has been implemented to effectively yield results for large-scale production strategies while maintaining a low problem dimension (the latter particularly useful for the derivative-free optimization methods applied in this work).

Further application requires benchmarking on realistic test cases and an honest attempt at comparing the quite different search features against useful baselines. Multiple metrics in figure and table form have therefore been used in this work to analyze results along relevant dimensions. Furthermore, best-performing tuning has been implemented for the different algorithms to enable comparison based primarily on their core search properties. Thus, each run is close to the

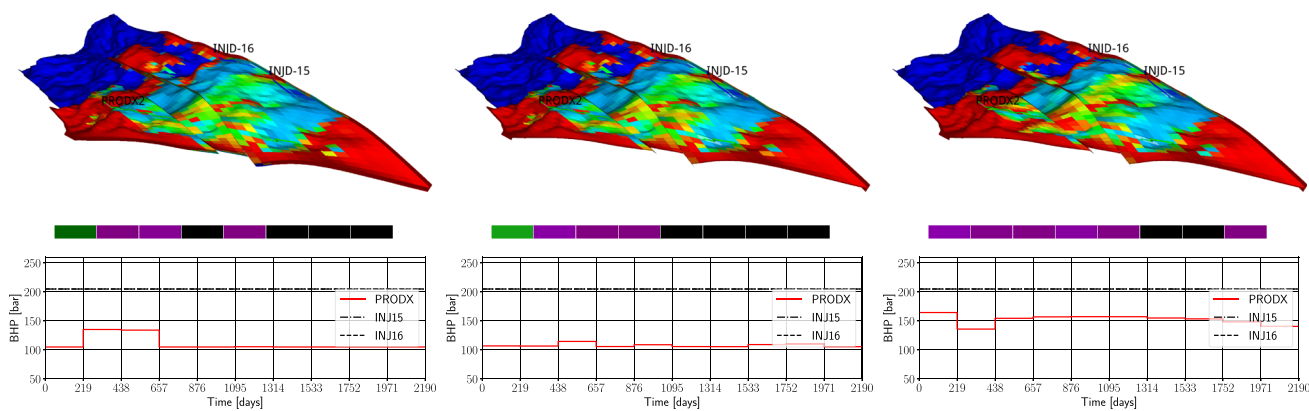


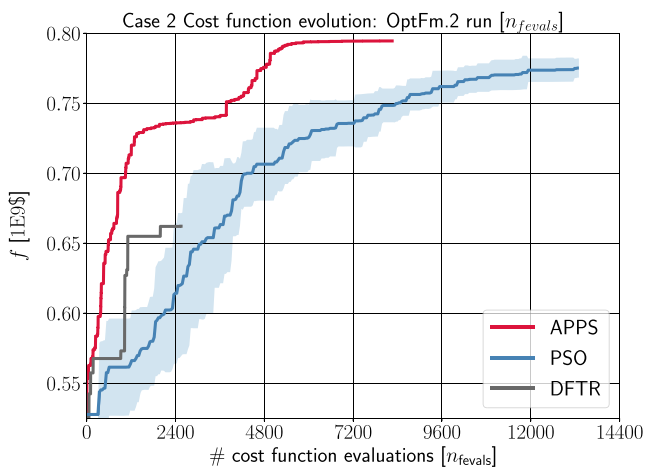
Fig. 13 Optimal ICD and BHP controls and corresponding saturation maps for Case 2 OptFm.1. Analog configuration as Fig. 11

best result and performance possibly achieved (by us) given that particular algorithm for that particular case. Overall, this structuring of the comparison provides nuance to the analy-

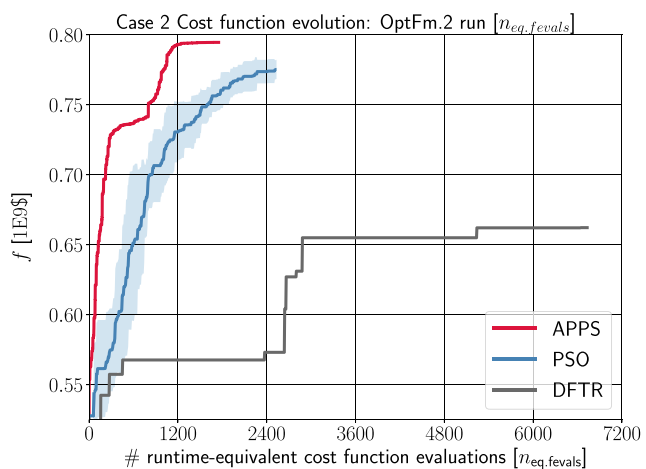
sis and enables highlighting the conditional usefulness of the various approaches given varying circumstances for application, e.g., availability of resources, and for particular cases, e.g., problem dimension and types of variables involved.

A few specific remarks can be made based on the results obtained in this work:

- Introducing the capability of dynamically managing sweep through varying well control settings not only provides higher objective value but can also be seen as a means to counter reservoir and fluid uncertainties and thus complement the functionality of the fixed ICD configuration. Altogether, this provides a context and lends support for performing joint completion design optimization involving fixed nozzle-based completions and well controls.
- APPS demonstrates robust performance across different problem formulations to support different exploration efforts. Comparably, DFTR results demonstrate decreasing performance for formulations involving a higher number of variables and/or comprising multiple variable types.
- Implementation of both APPS and PSO with full parallelization requires substantial computational resources and constrains the optimization effort to a lower number of concurrent trials. DFTR, on the other hand, applies a serial search approach that enables multiple trial runs at



(a)



(b)

Fig. 14 Case 2 OptFm.2 cost function evolution curves for APPS, PSO and DFTR runs

Table 11 Case 2 OptFm.2 results

| Proc. | $f^*[1 \times 10^9\$]$ | $\Delta[\%]$ | n_{fevals} | $t_{opt}[s]$ | n_c |
|-------|------------------------|--------------|--------------|-------------------|-------|
| APPS | 0.795 | 50.7 | 8254 | 3.8×10^5 | 38 |
| PSO | 0.775 (± 0.008) | 46.9 | 9500 | 5.5×10^5 | 38 |
| DFTR | 0.662 | 25.5 | 2550 | 1.5×10^6 | 1 |

Analog column configuration as Table 8. Best PSO run yields $f_{best}^* = 0.783$

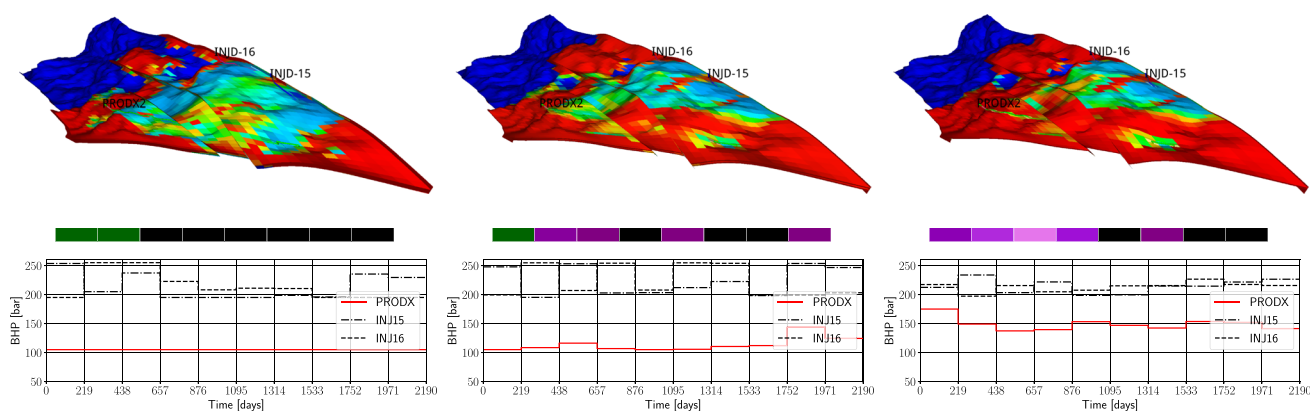


Fig. 15 Optimal ICD and BHP controls and corresponding saturation maps for Case 2 OptFm.2. Analog configuration as Figs. 11 and 13.

one time by requiring only a minimum of computational cores.

- DFTR results show in general a high search efficiency during early stages, but also substantial dependency on the initial position. This characteristic is better suited for local improvement around a particular point and can be an opportunity to improve the performance of a less-effective algorithm during late-stage convergence, e.g., in a hybrid configuration.

Finally, while this work has been scoped to not involve model uncertainty, a robust optimization will typically use the expected objective over a set of realizations covering the probable range of uncertain parameters. Though any final application needs to account for geological uncertainty as part of the design phase, this work has used a deterministic scope to focus on core algorithmic performance against problem types. However, performance over a set of realizations is a crucial dimension and a natural component in future algorithm development.

Acknowledgements The authors would like to thank Jarle Haukås for valuable contributions to the framing of this research, and SLB Stavanger Research (SSR) for hosting the authors during parts of this research.

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

Data Availability The optimization code used in this project is available on the GitHub page of the Petroleum Cybernetics Group NTNU (<https://github.com/PetroleumCyberneticsGroup/FieldOpt-Research-Open>). The data supporting this study's findings are available from the corresponding author upon reasonable request.

Declarations

Competing interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References


1. Constable, M.V., Antonsen, F., Stalheim, S.O., Olsen, P.A., Fjell, Ø.Z., Dray, N., Eikenes, S., Aarflot, H., Haldorsen, K., Digraanes, G., Seydoux, J., Omeragic, D., Thiel, M., Davydychev, A., Denichou, J.M., Salim, D., Frey, M., Homan, D., Tan, S.: Looking ahead of the bit while drilling: From vision to reality. SPWLA 57th Annual Logging Symposium (2016). SPWLA-2016-MMMM
2. Bergmo, P.E.S., Grimstad, A.A.: Water Shutoff Technologies for Reduced Energy Consumption. SPE Norway Subsurface Conference (2022). <https://doi.org/10.2118/209555-MS>. SPE-209555-MS
3. Leung, E., Nukhaev, M., Gottumukkala, V., Samosir, H., El-Fattah, M.A., Ogunsanwo, O., Gonzalez, A.: Horizontal well placement and completion optimisation in carbonate reservoirs. SPE Caspian Carbonates Technology Conference p. 19 (2010). <https://doi.org/10.2118/140048-MS>. SPE-140048-MS
4. Krogstad, S., Nilsen, H.M.: Efficient adjoint-based well-placement optimization using flow diagnostics proxies. ECMOR XVII - 17th European Conference on the Mathematics of Oil Recovery, Online Event (2020). <https://doi.org/10.3997/2214-4609.202035227>
5. Volkov, O., Voskov, D.V.: Effect of time stepping strategy on adjoint-based production optimization. *Comput. Geosci.* **20**(3), 707–722 (2016). <https://doi.org/10.1007/s10596-015-9528-1>
6. Al-Khelaiwi, F.T., Birchenko, V.M., Konopczynski, M.R., Davies, D.R.: Advanced wells: A comprehensive approach to the selection between passive and active inflow-control completions. *SPE Prod.*

- Oper. **25**(03), 305–326 (2010). <https://doi.org/10.2118/132976-PA>. SPE-132976-PA
7. Todman, S., Wood, G., Jackson, M.D.: Modelling and optimizing inflow control devices. SPE Kingdom of Saudi Arabia Annual Technical Symposium and Exhibition p. 19 (2017). SPE-188012-MS
 8. van der Bol, L., McCarty, A., Pritchett, J., Sriporntpraser, Y., Goh, G., Ridho, M., Natapak, R., Sowmyanarayanan, N.M.: ICD design optimisation with single-well dynamic 3D modelling and real-time operation support. International Petroleum Technology Conference p. 16 (2016). <https://doi.org/10.2523/IPTC-18848-MS>. IPTC-18848-MS
 9. Holmes, J.A.: Modeling advanced wells in reservoir simulation. J. Pet. Technol. **53**, 54–66 (2001). <https://doi.org/10.2118/72493-JPT>. SPE-72493-JPT
 10. Gurses, S.F., Vasper, A.C.: Optimized modeling workflows for designing passive flow control devices in horizontal wells. SPE Reservoir Characterization and Simulation Conference and Exhibition p. 10 (2013). <https://doi.org/10.2118/166052-MS>. SPE-166052-MS
 11. Jesmani, M., Jafarpour, B., Bellout, M.C., Foss, B.: A reduced random sampling strategy for fast robust well placement optimization. J. Petrol. Sci. Eng. **184**, 106414 (2020). <https://doi.org/10.1016/j.petrol.2019.106414>
 12. Holmes, J.A., Barkve, T., Lund, O.: Application of a multisegment well model to simulate flow in advanced wells. SPE European Petroleum Conference pp. 171–181 (1998). <https://doi.org/10.2118/50646-MS>. SPE-50646-MS
 13. Youngs, B., Neylon, K.J., Holmes, J.A.: Recent advances in modeling well inflow control devices in reservoir simulation. International Petroleum Technology Conference p. 6 (2009). IPTC-13925-MS
 14. Maas, T.R., Bouts, M.N., Joosten, G.J., Jansen, J.D.: The impact of smart completions on optimal well trajectories. Abu Dhabi International Petroleum Exhibition publisher & Conference (2017). <https://doi.org/10.2118/188368-MS>. SPE-188368-MS
 15. Gurses, S., Chochua, G., Rudic, A., Kumar, A.: Dynamic modeling and design optimization of cyclonic autonomous inflow control devices. SPE Reservoir Simulation Conference p. 15 (2019). <https://doi.org/10.2118/193824-MS>. SPE-193824-MS
 16. Youngs, B., Neylon, K., Holmes, J.: Multisegment well modeling optimizes inflow control devices. World Oil pp. 37–42 (2010)
 17. Agrawal, A., Leeuwen, P.V., Demirbas, B., Hajj, J.: Surface management of gas breakthrough in thin oil rim waxy reservoir: A case study. SPE Intelligent Energy International Conference and Exhibition p. 25 (2016). <https://doi.org/10.2118/181077-MS>. SPE-181077-MS
 18. Fernandes, P., Li, Z., Zhu, D.: Understanding the roles of inflow-control devices in optimizing horizontal-well performance. SPE Annual Technical Conference and Exhibition p. 12 (2009). <https://doi.org/10.2118/124677-MS>. SPE-124677-MS
 19. Webb, S.J., Revus, D., Myhre, A.M., Goodwin, N.H., Dunlop, N., Heritage, J.: Rapid model updating with right-time data - ensuring models remain evergreen for improved reservoir management. Intelligent Energy Conference and Exhibition p. 13 (2008). <https://doi.org/10.2118/112246-MS>. SPE-112246-MS
 20. Antonsen, F., Teixeira De Oliveira, M.E., Petersen, S.A., Metcalfe, R.W., Hermanrud, K., Constable, M.V., Boyle, C.T., Eliassen, H.E., Salim, D., Seydoux, J., Omeragic, D., Thiel, M., Denichou, J.M., Etchebes, M., Nickel, M.: Geosteering in complex mature fields through integration of 3D multi-scale LWD-data, geomodels, surface and time-lapse seismic. SPWLA 59th Annual Logging Symposium p. 16 (2018). SPWLA-2018-Q
 21. Warrlich, G.M.D., Abu-Shiekh, I., Alexander, D.M., Zhu, F., Goossens, P.M., Tull, S.S., Al-Lamki, A.A.: Maintaining an “Evergreen” model to optimise a waterflood development in a carbonate transition zone field. SPE/EAGE Reservoir Characterization and Simulation Conference p. 14 (2009). <https://doi.org/10.2118/125552-MS>. SPE-125552-MS
 22. Salim, D., Couto, P., Alves, J., Freitas, E., Haq, S., Denichou, J.M.: Optimizing recovery by integrating an advanced reservoir simulation approach into the drilling of horizontal wells. Offshore Technology Conference (2015). OTC-26161-MS
 23. Jesmani, M., Bellout, M.C., Hanea, R., Foss, B.: Well placement optimization subject to realistic field development constraints. Comput. Geosci. **20**(6), 1185–1209 (2016). <https://doi.org/10.1007/s10596-016-9584-1>
 24. Goh, G., Tan, T., Zhang, L.M.: A unique ICD’s advance completions design solution with single well dynamic modeling. IADC/SPE Asia Pacific Drilling Technology Conference p. 12 (2016). <https://doi.org/10.2118/180672-MS>. SPE-180672-MS
 25. Dong, C., Dupuis, C., Morriss, C., Legendre, E., Mirto, E., Kutiev, G., Denichou, J.M., Viandante, M., Seydoux, J., Bennett, N., Zhu, Q., Zhong, X.: Application of automatic stochastic inversion for multilayer reservoir mapping while drilling measurements. Abu Dhabi International Petroleum Exhibition and Conference p. 14 (2015). <https://doi.org/10.2118/177883-MS>. SPE-177883-MS
 26. Seydoux, J., Legendre, E., Mirto, E., Dupuis, C., Denichou, J.M., Bennett, N., Kutiev, G., Kuchenbecker, M., Morriss, C., Yang, L.: Full 3D deep directional resistivity measurements optimize well placement and provide reservoir-scale imaging while drilling. SPWLA 55th Annual Logging Symposium p. 14 (2014). SPWLA-2014-LLLL
 27. Maggs, D., Raffn, A.G., Porturas, F., Murison, J., Tay, F., Suwarian, W., Samsudin, N.B., Yusmar, W.Z.A., Yusof, B.W., Imran, T.N.O.M., Abdullah, N.A., Reffin, M.Z.B.M.: Production optimization for second stage field development using ICD and advanced well placement technology. Europec/EAGE Conference and Exhibition p. 11 (2008). <https://doi.org/10.2118/113577-MS>. SPE-113577-MS
 28. Henriksen, K.H., Gule, E.I., Augustine, J.R.: Case study: The application of inflow control devices in the Troll field. SPE Europec/EAGE Annual Conference and Exhibition p. 5 (2006). <https://doi.org/10.2118/100308-MS>. SPE-100308-MS
 29. Montaron, B.A., Bradley, D.C., Cooke, A., Prouvost, L.P., Raffn, A.G., Vidal, A., Wilt, M.: Shapes of flood fronts in heterogeneous reservoirs and oil recovery strategies. SPE/EAGE Reservoir Characterization and Simulation Conference p. 18 (2007). <https://doi.org/10.2118/111147-MS>. SPE-111147-MS
 30. Raffn, A.G., Zeybek, M., Moen, T., Lauritzen, J.E., Sunbul, A.H., Hembling, D.E., Majdpour, A.: Case histories of improved horizontal well cleanup and sweep efficiency with nozzle based inflow control devices (ICD) in sandstone and carbonate reservoirs. Offshore Technology Conference p. 9 (2008). <https://doi.org/10.4043/19172-MS>. OTC-19172-MS
 31. Karim, R.A., Goh, K.F.G., Nuriyadi, M.A., Ahmad, N.A., Leung, E., Murison, J.A.: Horizontal well optimization with inflow control devices (ICDs) application in heterogeneous and dipping gas-capped oil reservoirs. SPE Annual Technical Conference and Exhibition p. 15 (2010). <https://doi.org/10.2118/133336-MS>. SPE-133336-MS
 32. Venkitaraman, A., Manrique, J.F., Poe, B.D.J.: A comprehensive approach to completion optimization. SPE Eastern Regional Meeting p. 11 (2001). SPE-72386-MS
 33. Al Hashemi, M., Bellah, S., Gurses, S., Akhtar, M.N.: ICD completions optimization for an offshore Abu Dhabi well using dynamic modeling. SPE Reservoir Characterization and Simulation Conference and Exhibition p. 9 (2013). <https://doi.org/10.2118/165962-MS>. SPE-165962-MS
 34. Li, D., Alobedli, A., Selvam, B., Azoug, Y., Obeta, C., Nguyen, M., Al-Shehhi, B.H.: A new ICD/ICV well completion design optimizer and well management logic for full field reservoir simulation

- with multiple ICD/ICV wells. Abu Dhabi International Petroleum Exhibition publisher & Conference p. 17 (2017). <https://doi.org/10.2118/188642-MS>. SPE-188642-MS
35. Holmes, J.A., Byer, T.J., Edwards, D.A., Stone, T.W., Pallister, I., Shaw, G.J., Walsh, D.: A unified wellbore model for reservoir simulation. SPE Annual Technical Conference and Exhibition p. 14 (2010). <https://doi.org/10.2118/134928-MS>. SPE-134928-MS
 36. Stone, T.W., Bennett, J., Law, D.H.S., Holmes, J.A.: Thermal simulation with multisegment wells. SPE Reserv. Eval. Eng. **5**(03), 206–218 (2002). <https://doi.org/10.2118/78131-PA>. SPE-78131-PA
 37. Elfeel, M.A., Goh, G., Biniwale, S.: Advanced completion optimization ACO: A comprehensive workflow for flow control devices (2021). <https://doi.org/10.2523/IPTC-21189-MS>. IPTC-21189-MS
 38. Volkov, O., Bellout, M.C.: Gradient-based constrained well placement optimization. J. Petrol. Sci. Eng. **171**, 1052–1066 (2018). <https://doi.org/10.1016/j.petrol.2018.08.033>
 39. Volkov, O., Bellout, M.C.: Gradient-based production optimization with simulation-based economic constraints. Comput. Geosci. **21**(5), 1385–1402 (2017). <https://doi.org/10.1007/s10596-017-9634-3>
 40. Baumann, E.J.M., Dale, S.I., Bellout, M.C.: FieldOpt: A powerful and effective programming framework tailored for field development optimization. Comput. Geosci. **135**, 104379 (2020). <https://doi.org/10.1016/j.cageo.2019.104379>
 41. FieldOpt Research: Open-source repository with up-to-date research code. <https://github.com/PetroleumCyberneticsGroup/FieldOpt-Research-Open> (2022). Accessed 14 Aug 2022
 42. Kristoffersen, B.S., Bellout, M.C., Silva, T.L., Berg, C.F.: An automatic well planner for complex well trajectories. Math. Geosci. **53**(8), 1881–1905 (2021). <https://doi.org/10.1007/s11004-021-09953-x>
 43. Silva, T.L., Bellout, M.C., Giuliani, C., Camponogara, E., Pavlov, A.: A derivative-free trust-region algorithm for well control optimization. ECMOR XVII - 17th European Conference on the Mathematics of Oil Recovery, Online Event (2020). <https://doi.org/10.3997/2214-4609.202035086>
 44. Silva, T.L., Bellout, M.C., Giuliani, C., Camponogara, E., Pavlov, A.: Derivative-free trust region optimization for robust well control under geological uncertainty. Comput. Geosci. **26**(2), 329–349 (2022). <https://doi.org/10.1007/s10596-022-10132-y>
 45. Isebor, O.J., Durlafsky, L.J., Echeverría Ciaurri, D.: A derivative-free methodology with local and global search for the constrained joint optimization of well locations and controls. Comput. Geosci. **18**(3–4), 463–482 (2014). <https://doi.org/10.1007/s10596-013-9383-x>
 46. 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)
 47. Kolda, T.G.: Revisiting asynchronous parallel pattern search for nonlinear optimization. SIAM J. Optim. **16**(2), 563–586 (2005)
 48. Conn, A., Scheinberg, K., Vicente, L.: Introduction to derivative-free optimization. Society for Industrial and Applied Mathematics (2009). <https://doi.org/10.1137/1.9780898718768>
 49. Charles, A., Dennis, J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006). <https://doi.org/10.1137/040603371>
 50. Tamara, K.G., Virginia, T.: Understanding asynchronous parallel pattern search. Tech. rep., Livermore, CA (2002)
 51. Floreano, D., Mattiussi, C.: Bio-Inspired Artificial Intelligence: Theories, Methods and Technologies. MIT Press (2008)
 52. Gad, A.G.: Particle swarm optimization algorithm and its applications: A systematic review. Arch. Comput. Meth. Eng. pp. 1–31 (2022)
 53. Forouzanfar, F., Reynolds, A.C., Li, G.: Optimization of the well locations and completions for vertical and horizontal wells using a derivative-free optimization algorithm. J. Pet. Sci. Eng. **86–87**(Supplement C), 272–288 (2012). <https://doi.org/10.1016/j.petrol.2012.03.014>
 54. Merlini Giuliani, C.: Distributed Derivative Free Optimization. Universidade Federal de Santa Catarina, Presentation PhD Defense (2016)
 55. Audet, C., Hare, W.: Derivative-free and blackbox optimization. Springer (2017)
 56. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005). <https://doi.org/10.1137/S0036144504446096>
 57. Conn, A.R., Scheinberg, K., Vicente, L.N.: Geometry of sample sets in derivative-free optimization: Polynomial regression and underdetermined interpolation. IMA J. Numer. Anal. **28**(4), 721 (2008). <https://doi.org/10.1093/imanum/drn046>
 58. Conn, A.R., Scheinberg, K., Vicente, L.N.: Geometry of interpolation sets in derivative free optimization. Math. Program. **111**(1), 141–172 (2008). <https://doi.org/10.1007/s10107-006-0073-5>
 59. Conn, A., Gould, N., Toint, P.: Trust region methods. Society for Industrial and Applied Mathematics (2000). <https://doi.org/10.1137/1.9780898719857>
 60. 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)
 61. Christie, M.A., Blunt, M.J.: Tenth SPE comparative solution project: A comparison of upscaling techniques. SPE Reserv. Eval. Eng. **4**(4), 308–317 (2001). <https://doi.org/10.2118/72469-PA>
 62. Fonseca, R., Della Rossa, E., Emerick, A., Hanea, R., Jansen, J.: Overview of the Olympus field development optimization challenge. ECMOR XVI - 16th European Conference on the Mathematics of Oil Recovery **2018**(1), 1–10 (2018). <https://doi.org/10.3997/2214-4609.201802246>
 63. Bellout, M.C., Echeverría Ciaurri, D., Durlafsky, L.J., Foss, B., Kleppe, J.: Joint optimization of oil well placement and controls. Comput. Geosci. **16**(4), 1061–1079 (2012). <https://doi.org/10.1007/s10596-012-9303-5>
 64. Gill, P.E., Murray, W., Saunders, M.A.: Practical Optimization. Academic Press, San Diego, CA, USA (1981)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Mathias C. Bellout¹ · Thiago L. Silva^{1,2} · Jan Øystein Haavig Bakke³ · Carl Fredrik Berg¹ 

Mathias C. Bellout
bellout@alumni.ntnu.no

Thiago L. Silva
thiago.silva@sintef.no

Jan Øystein Haavig Bakke
JBakke@slb.com

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

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

³ Schlumberger Norway Technology Center, Stavanger,
Norway