Systemizing the Solution of Simulation-Driven Optimization Problems

Marco Enriquez
(joint work with William Symes)

Center for Advanced Aviation Systems Development (CAASD)
menriquez@mitre.org

November 21, 2013

National Institute of Standards and Technology (NIST)

Outline

Definition and Problem Statement

- Simulation-Driven Optimization [SDO] Problems
- Solution of the SDO Problem
  - The Adjoint State Method

`TSOpt` ("Time Stepping for Optimization") Framework

- Generic Approach to Solving SDO Problems
- `TSOpt` Components and Features
  - Implementation variants of the Adjoint State Method

Numerical Results

- Solving the "Optimal Well Rate Allocation Problem" with `TSOpt`

Conclusion

Definition

A Simulation-Driven Optimization Problem:

$$
\begin{aligned}
[SD] \quad \min_c f(c) &= \int_0^T J(u(t),c)dt \\
\text{where } (u(t),c) \text{ solves:} & \\
\frac{d}{dt}u(t) &= H(u(t),c), \qquad t \in [0,T] \\
H &\equiv 0 \text{ for } t < 0
\end{aligned}
$$

Where H is a dynamic operator, and:

$c \in \mathbb{R}^n$,

$u \in C^1([0,T] \times \mathbb{R}^n, U)$, for a state Hilbert space U ,

$J : C^1([0,T] \times \mathbb{R}^n, U) \to \mathbb{R}$

Hence $f : \mathbb{R}^n \to \mathbb{R}$

Definition

A Simulation-Driven Optimization Problem:

$$[SD] \quad \min_c f(c) = \int_0^T J(u(t),c)dt$$

where $(u(t),c)$ solves:

$$\frac{d}{dt}u(t) = H(u(t),c), \qquad t \in [0,T]$$
$$H \equiv 0 \text{ for } t < 0$$

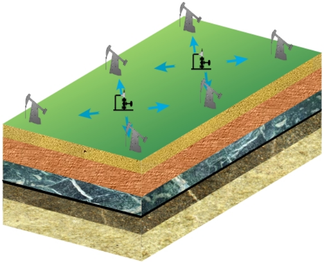
Where H is a dynamic operator, and:

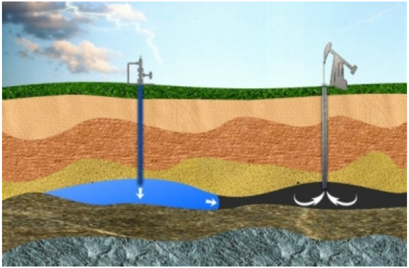
$c \in \mathbb{R}^n$,
$u \in C^1([0,T] \times \mathbb{R}^n, U)$, for a state Hilbert space U ,
$J : C^1([0,T] \times \mathbb{R}^n, U) \to \mathbb{R}$

Hence $f : \mathbb{R}^n \to \mathbb{R}$

Motivating Examples: Optimal Well Rate Allocation

Given a reservoir model, along with location of injection and production wells, find the optimal well rates to maximize revenue





[1] Images courtesy of www.amerexco.com/recovery

Motivating Examples: Optimal Trajectories

Find the optimal aircraft trajectory that minimizes fuel and/or time cost, given path constraints

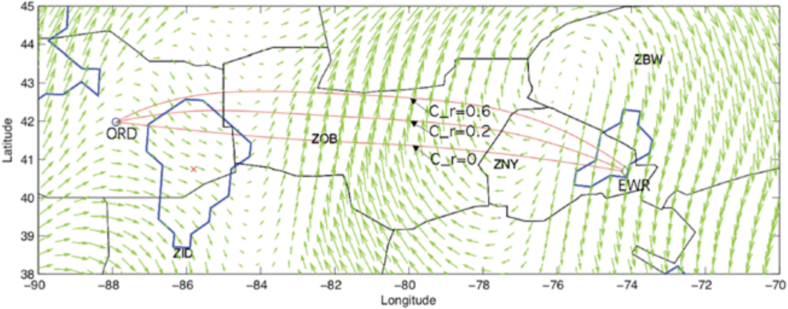


Figure 3. Optimal trajectories at 34,000 feet from ORD to EWR with different design parameters.

[1]B. Sridhar et al., "Aircraft Trajectory Optimization and Contrails Avoidance in the Presence of Winds"

Claim

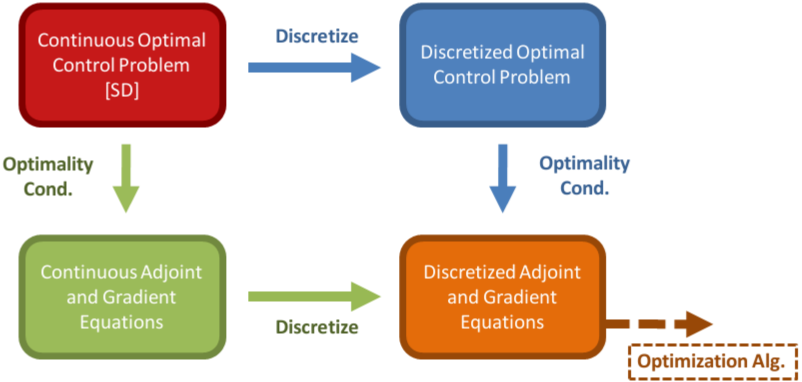
Despite the variety of problems we can pose, the (numerical) approach to solving [SD] requires similar steps, executed by the same components!

Why not "abstract" these steps/interactions, and capture it in a framework?

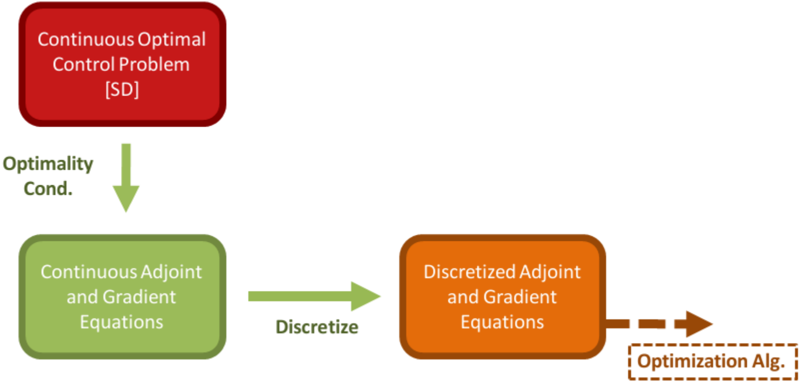
Benefits of "systemization":

- Reduced code base
- Consistency checks
- Easily switch between computational strategies

Solving the [SD]



"Optimize-then-Discretize" (OtD)



The Continuous Adjoint-State Method

Applying the optimality conditions to [SD], for $t \in [0, T]$:

Continuous State Equation:

$$\frac{du}{dt} = H(u(t), c) \qquad u(0) \equiv 0$$

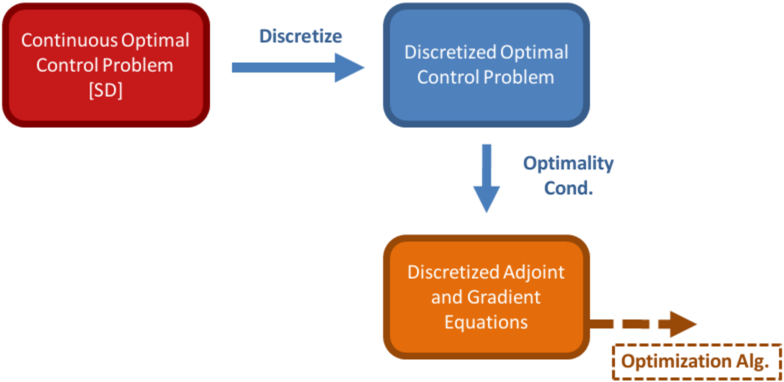
Continuous Adjoint Equation:

$$\frac{dw}{dt} = -D_u H(u(t), c)^* w(t) + J_u(u(t), c) \qquad w(T) \equiv 0$$

Gradient (w.r.t. $\mathcal{L}^2$ inner product):

$$\nabla f(c)(t) = D_c H(u(t), c)^* w(t) + J_c(u(t), c)$$

"Discretize-then-Optimize" (DtO)



The Discrete Optimal Control Problem

Using a fixed time-stepping algorithm, the discretized analogue of [SD]:

$$
\begin{aligned}
[DSD] \qquad \min_{c} \qquad & \sum_{i=0}^{N} J[u^{(i)}, c]\Delta t \\
s.t. \qquad & u^{(n+1)} = u^{(n)} + \Delta t \, \bar{H}^{(n)}[u^{(n)}, c] \qquad n = 0, \ldots, N \\
& u^{(0)} \equiv 0 \, ,
\end{aligned}
$$

where $\bar{H}^{(n)}$ is a (time-dependent) discrete dynamic operator. Note that each $u^{(n)} \simeq u(t_n)$ is called a *(simulation/forward) state*

MITRE

Adjoint-State Methods (ASM)

Applying the optimality conditions to [DSD]:

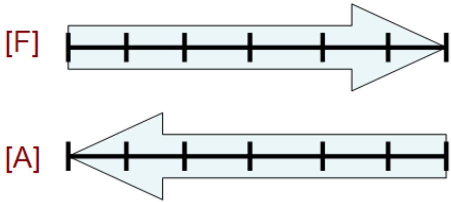
$$
\begin{array}{llcl}
[A] & w^{(n)} & = & w^{(n+1)} + \Delta t \left(D_u \bar{H}^{(n)}[u^{(n)}, c]^* w^{(n+1)} + J_u[u^{(n)}, c] \right) \\
 & w^{(N)} & \equiv & 0
\end{array}
$$

The gradient can then be obtained through the following accumulation

$$
\Delta t \sum_n D_c \bar{H}^{(n)}[u^{(n)}, c]^* w^{(n)} + J_c[u^{(n)}, c]
$$

Visualizing the Adjoint-State Method

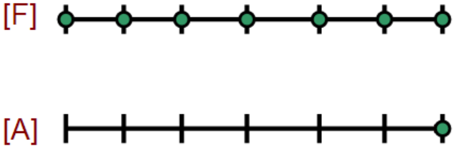
Introduce the forward grid (the grid used by the forward evolution) and the adjoint grid (which will be used by the adjoint state method)



Visualizing the Adjoint-State Method

Once we complete the reference simulation, we can begin the adjoint simulation. Start with the adjoint state's (given) final value:

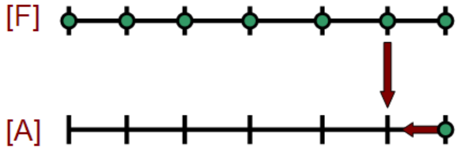
$$w^{(N)} = 0$$



Visualizing the Adjoint-State Method

Then start the backward evolution by using the *last* adjoint state and a corresponding forward state:

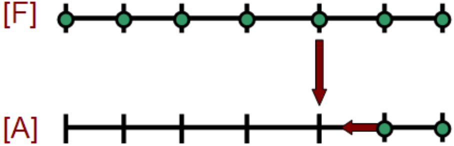
$$w^{(N-1)} \quad = \quad w^{(N)} + \Delta t \left(D_u \bar{H}^{(N-1)}[u^{(N-1)}, c]^* w^{(N)} + J_u[u^{(N-1)}, c] \right)$$



Visualizing the Adjoint-State Method

Iterate through this process ...

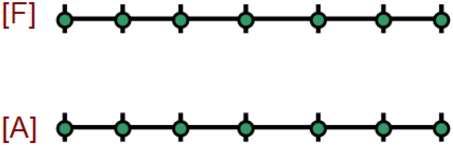
$$w^{(N-2)} \quad = \quad w^{(N-1)} + \Delta t \left(D_u \bar{H}^{(N-2)}[u^{(N-2)}, c]^* w^{(N-1)} + J_u[u^{(N-2)}, c] \right)$$



Visualizing the Adjoint-State Method

... to eventually generate all the adjoint states

$$w^{(0)} \quad = \quad w^{(1)} + \Delta t \left(D_u \bar{H}^{(0)}[u^{(0)}, c]^* w^{(1)} + J_u[u^{(0)}, c] \right)$$



TSOpt ("Time Stepping For Optimization")

TSOpt is "middle-ware" written in C++, designed to aid solution of simulation-driven optimization problems

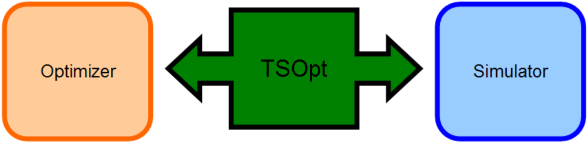
TSOpt:

- abstracts commonalities among time-stepping methods
- provides a way for a simulation package to inter-operate with optimization algorithms
- supports use of the adjoint-state method

Motivating observation: for every simulation driven optimization problem, the solution process is (mostly) the same:

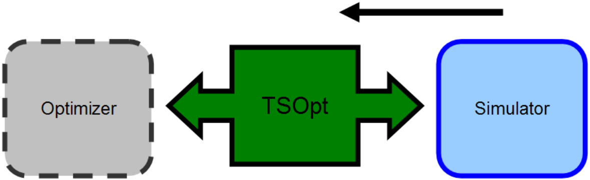
- reference, linearized and adjoint simulation execution order
- constructing needed data structures for optimization

TSOpt ("Time Stepping For Optimization")



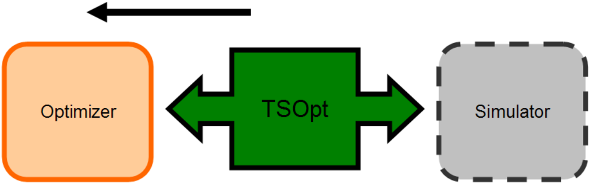
TSOpt ("Time Stepping For Optimization")

$$u^{+} = u + \Delta t H(u, c, t)$$

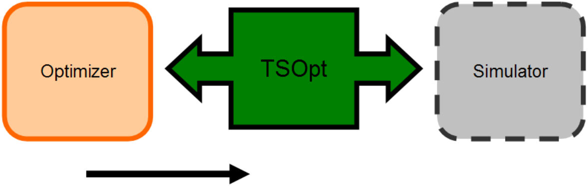


TSOpt ("Time Stepping For Optimization")

$$\nabla J(c), J(c), \ldots$$

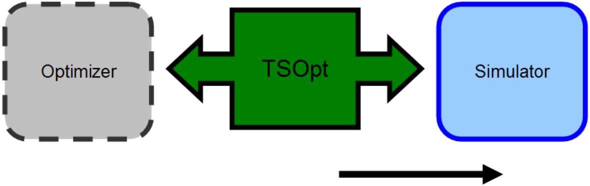


TSOpt ("Time Stepping For Optimization")



$$s = -B(c)^{-1}\nabla J(c)$$
$$c^{+} = c + \alpha s$$

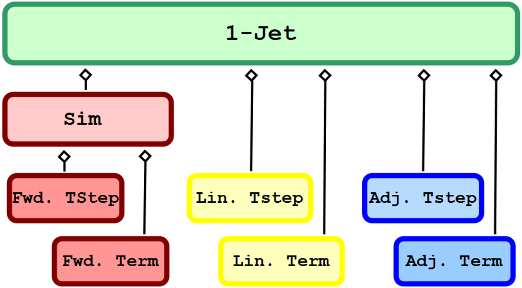
TSOpt ("Time Stepping For Optimization")



$$c = c^{+}$$

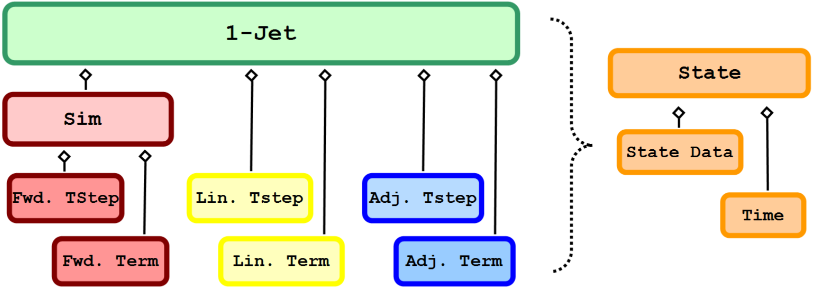
TSOpt's Components

In TSOpt, we use Jet objects to perform various simulations.



TSOpt's Components

In TSOpt, we use Jet objects to perform various simulations.



Inversion Software Construction

`TSOpt`'s modular structure minimizes the amount of code needed to perform an inversion

User:

- provides `TSOpt` with a forward, linearized, and adjoint "step"
- provide a "State" class

`TSOpt`:

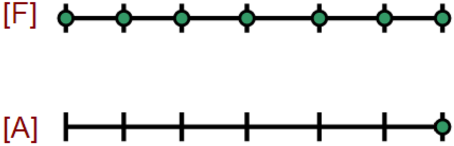
- arranges proper execution forward, linearized and adjoint simulation
- implements the Adjoint-State method to form gradients

Output can be passed to optimization software

TSOpt and the Adjoint-State (AS) Method

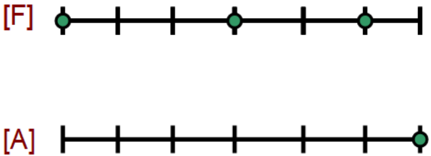
Recall the ASM requires access to the reference simulation state history. TSOpt implements the following to manage state storage:

- **save all**: save states as you forward simulate, access as needed
  - Cost: A typical 3D RTM, $O(\text{TB})$

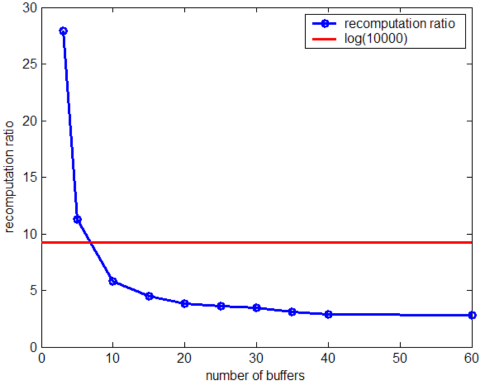


TSOpt and the Adjoint-State (AS) Method

- **Griewank's optimal checkpointing**: rely on forward simulations, *and* use stored simulation states as a starting point for evolution
  - Recomputation Ratio = Total Number of Forward Traversals $/N$
  - Cost: $O(log(N))$ recomputation, given a special distribution of the states and a small amount of buffers



Checkpointing, $N = 10000$



buffers	3	5	10	15	20	25	30	35	40	60
ratio	27.9	11.3	5.8	4.5	3.8	3.6	3.4	3.1	2.9	2.8

Simulation Verification

In order to obtain meaningful results from inversion, one must guarantee that the gradient is accurate

Gradient quality depends on the adjoint states, which depends on:

- linearization of the reference equations
- adjoint of the linearization

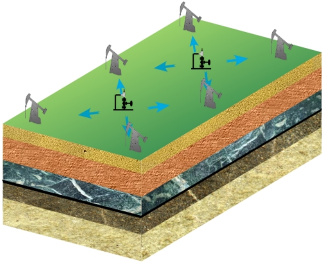
`TSOpt` is capable of the following simulation verification (**unit**) tests:

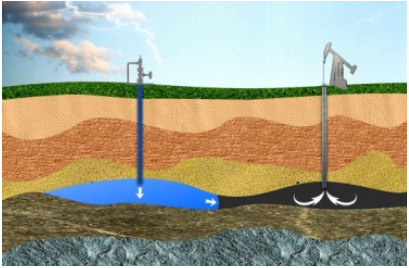
- **derivative test**: compare linearized simulation to finite difference approximation (using reference simulation)
- **dot product test**: give the linearized simulation operator A, adjoint simulation operator A^* and random control x and random state y, check $\langle Ax, y\rangle - \langle x, A^*y\rangle$ (Fixed timestep only)

MITRE

Revisiting the Optimal Well Rate Allocation Problem

Given a reservoir model, along with location of injection and production wells, find the optimal well rates to maximize revenue





[1]Images courtesy of www.amerexco.com/recovery

The Optimal Well Rate Allocation Problem

The optimal well rate allocation problem can be written as:

$$\max_{q_i \quad i \in I \cup P} \quad f(q) = \int_0^T dt \underbrace{\sum_{i \in P} \alpha s_o q_i(t)}_{\text{profit, oil produced}} - \underbrace{\sum_{i \in P} \frac{\beta}{2} s_a q_i^2(t)}_{\text{water production penalty}} - \underbrace{\sum_{i \in I} \gamma q_i(t)}_{\text{cost to inject}},$$

where α, β and γ are scalar variables and the aqueous pressure p and aqueous saturation s_a solve the Black Oil equations:

$$0 = B\left(s_a(t), \frac{ds_a}{dt}, p(t), q\right)$$

MITRE

The Black Oil Equations

- DE system that captures two-fluid inteactions in a porous medium
  - Appropriate for low to moderate fluid flow
- Assumption: incompressible fluid and rock

$$-\nabla \cdot (\underbrace{K(x)}_{\text{abs. perm.}} \lambda_{tot}(s_a(x,t))\nabla p(x,t)) =$$

$$\sum_{i \in P}(1 - s_a)q_i(t)\delta(x - x_i) + \sum_{i \in P \cup I} s_a q_i(t)\delta(x - x_i)$$

$$\underbrace{\phi(x)}_{\text{rock por.}} \frac{\partial}{\partial t} s_a(x,t) - \nabla \cdot (K(x) \underbrace{\lambda_a}_{\text{phase mob.}} (s_a(x,t))\nabla p(x,t)) =$$

$$\sum_{i \in P \cup I} s_a q_i(t)\delta(x - x_i)$$

Fully Discretized Problem

After using a Finite Volume method in space and Bwd. Euler in time:

$$\begin{aligned} \min \qquad & \bar{f}(q) = \sum_{k=1}^{N} \Delta t\, \bar{J}(k, s_a^{(k)}, q) \\ \text{s.t.} \qquad & e^T q = 0 \\ & q_{min} \leq q_i \leq q_{max} \end{aligned}$$

where $s_a^{(k+1)}$ and $p^{(k+1)}$ solve:

$$\begin{bmatrix} F(\ldots^{(k+1)}, q) \\ G(\ldots^{(k+1)}, q) \end{bmatrix} := \begin{bmatrix} q^{(k+1)} - A^{(t)} p^{(k+1)} \\ D^{-1}(q^{(k+1)} - A^{(a)} p^{(k+1)}) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{s_a^{(k+1)} - s_a^{(k)}}{k} \end{bmatrix}$$

where the matrices $A^{(\theta)}$ and D are defined as:

$$D_{i,i} = \phi_i \cdot |\Omega_i| \qquad T_{i,j} = cK_{i,j}$$

$$A_{i,j}^{(\theta)} = -T_{i,j} \lambda_{\theta_{i,j}} \qquad A_{i,i} = \sum_j T_{i,j} \lambda_{\theta_{i,j}}$$

MITRE

The Adjoint Equations

Simultaneously solve for the adjoint variables $w_s^{(k)}$ and $w_p^{(k)}$ in the following equation:

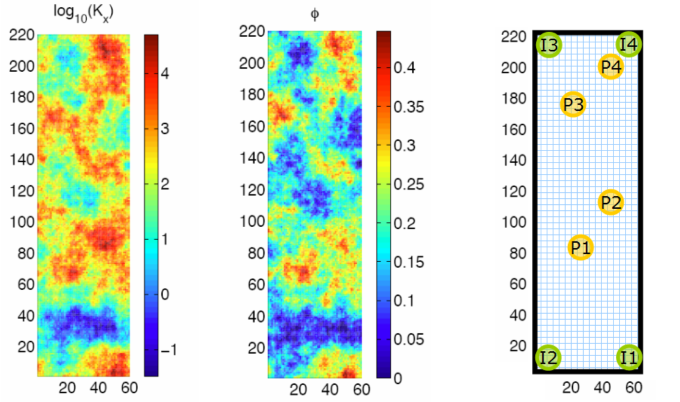
$$
\begin{aligned}
-\frac{w_s^{(k+1)} - w_s^{(k)}}{k} &= D_sF(\ldots^{(k)})^T w_s^{(k)} - D_sG(\ldots^{(k)})^T w_p^{(k)} - \nabla_s \bar{J}(\ldots^{(k)}) \\
0 &= -D_pF(\ldots^{(k)})^T w_s^{(k)} + D_pG(\ldots^{(k)})^T w_p^{(k)}
\end{aligned}
$$

The directional derivative can then be obtained from the following expression:

$$
\nabla J(q)(k) = \nabla_q \bar{J}(\cdot^{(k)}) - D_qF(\ldots^{(k)})^T w_s^{(k)} + D_qG(\ldots^{(k)})^T w_p^{(k)}
$$

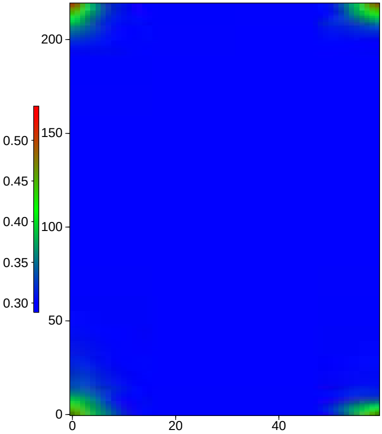
Simulation Information

- SPE10 data for porosity and permeability (left)
- Location of Injecting/Producing Wells (right)
- Grid Cell Size: 10×20 feet



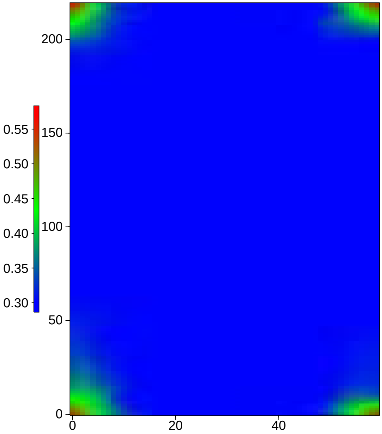
Reference Simulation Results

Saturation plot for $t = 25$ days



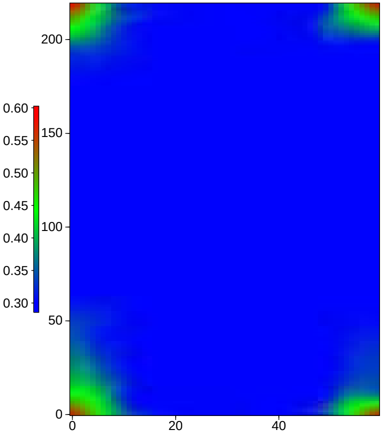
Reference Simulation Results

Saturation plot for $t = 50$ days



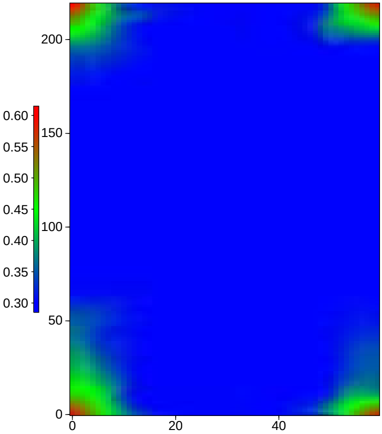
Reference Simulation Results

Saturation plot for $t = 75$ days



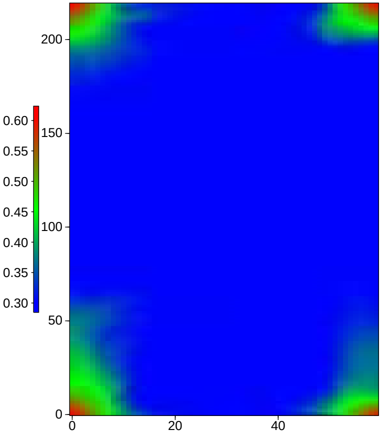
Reference Simulation Results

Saturation plot for $t = 100$ days



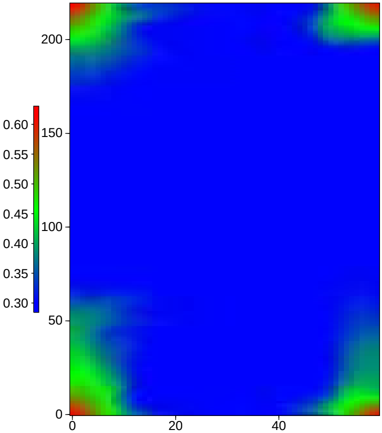
Reference Simulation Results

Saturation plot for $t = 125$ days



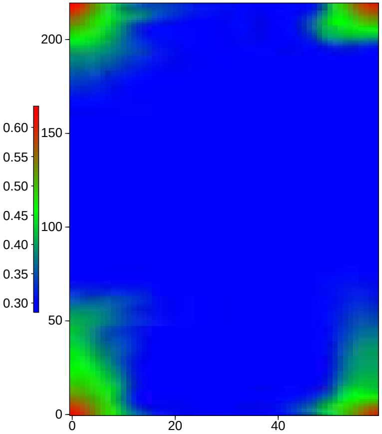
Reference Simulation Results

Saturation plot for $t = 150$ days



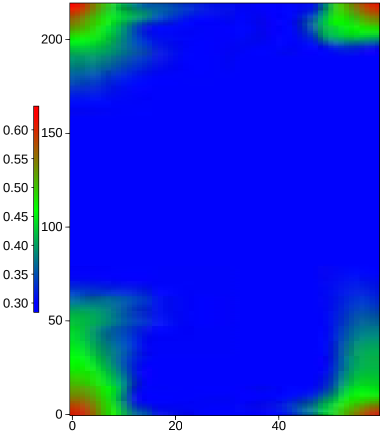
Reference Simulation Results

Saturation plot for $t = 175$ days



Reference Simulation Results

Saturation plot for $t = 200$ days



Inversion Information

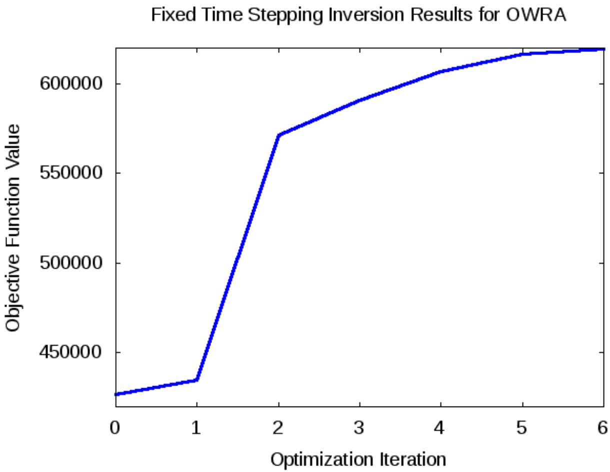
Computational Software:

- Simulation: `BlackOil` simulator
- `TSOpt` to handle simulation execution, gradient construction
- Optimization: `IPOpt`, "Interior-Point Optimizer"

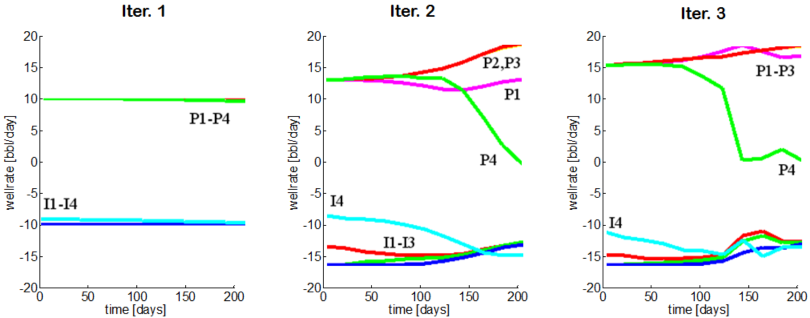
Inversion:

- Find optimal well-rate configuration over 200-day timespan
  - Time step size: $\Delta t = 25d$
- LBFGS Hessian approximation
- Globalization: Linesearch
- Wellrate bounds: $[0, 20]$ bbl/day
- Initial guess: 10 bbl/day for all wells
- Stopping Tolerance: 0.10 (NLP Error)

Objective Function



Control History



Conclusion

`TSOpt`: Modular `C++` framework aiding inversion software construction

- Systemizes process of solving SDO problems by encapsulating/automating common actions
- Reduces code required to successfully perform inversion

`TSOpt` Features:

- Easily switch between strategies for gradient formation
- Supports fixed and adaptive simulations
- Includes "sanity tests": derivative and dot-product test

Conclusion

Open question:

- Are there tests for the components which, if passed, would guarantee the local solution of [SD] will be attained?

Read more about `TSOpt`:

- **Tech Report:** `http://www.caam.rice.edu/tech_reports/2009/TR0933.pdf`
- **Doxy:** `http://www.trip.caam.rice.edu/software/rvl ...`
  `.../tsopt/doc/html/index.html`

Questions?