Reservoir Life-Cycle – Integrated Reservoir Management

Reservoir Life-Cycle

Exploration
- Regional Reviews
- Play Definition
- 3D Basin Modeling
- 2D/3D Seismic
- Acquisition
- Processing
- Interpretation
- Drilling
- Prospect Identification
- Strategic Entries
- Data Rooms

Development
- Detailed Geology
- 3D Seismic
- Acquisition
- Processing
- Interpretation
- Drilling
- Petrophysics
- Reservoir Engineering
- Reservoir Simulation
- Well Technology
- Economics
- Conceptual Engineering

Production
- Asset Management
- Reservoir Management
- Reservoir Monitoring
- History Matching
- Formation Damage
- Stimulation
- Production Optimization
- Enhanced Recovery

Storage
- Underground Gas Storage
- CO2 Sequestration

Source: Google Images

Large-Scale Models + Uncertainty
Real-Time Closed-Loop Reservoir Management (CLRM)

- Well Control Optimization
- Well Placement/Well sequence Optimization
- Drilling/Well Trajectory Optimization

Drilling

- Fast Reservoir Simulation
- Data assimilation

Optimizer

Real Oil Reservoir

Reservoir models

Measurements

Update permeability/porosity field

Uncertainty + Multiple Realizations

Source: http://www.imm.dtu.dk/~jbjo/oilproduction.html
Reservoir Simulation – Key Component in CLRM

[Reservoir Simulation Diagram]

[Sorek & Gildin Comp. Geosciences, 2018]

[Reservoir Simulation Diagram]

[Zalavadia & Gildin et al., ATCE, 2019]

[Reservoir Simulation Diagram]

[Florez and Gildin Comp. Geosciences, 2019]
Part 1

Fast Proxies for Reservoir Simulation

By
Eduardo Gildin, Marcelo Dall’Aqua, Emilio Coutinho, Yalchin Efendiev, past students
How to reduce the computational time?

- Fine Scale Model
  (High Complexity)

- Reduced Order Model
  (Minimum number of parameters)

- Data-driven models

---

Inputs

Well Controls (BHP, q_o, q_w, ...)

Fine Model

$N \approx 10^6$

Measurements (BHP, q_o, watercut, ...)

Outputs

Coarse/Reduced Model

$N \approx 10^2$

Approximately Equal

Outputs

Slow, Computationally expensive

Model Fidelity

High accuracy, Needs less data to calibrate
Explanatory, Can often extrapolate

---

Complexity (Physics)

Computational Speed

Data to train
Black box, Limited or no extrapolation

Harold Vance Department of Petroleum Engineering
Numerical Petroleum Reservoir Simulation

• Widely used from the initial stages to the end of the development of an oil field lifecycle;

• Combination:
  - Geological rock properties \((k, \phi)\)
  - Fluid properties \((\rho, \lambda)\)
  - Fluid flow physics

\[
\nabla \left[ \kappa \lambda_j \rho_j \nabla P_j \right] + q_j - \frac{\partial}{\partial t} (\phi S_j \rho_j) = 0
\]

Where \(j\) can be oil and water

*SPE RSC 2020, SPEJ 2015-2019, Comp Geosc 2017-2019*
System Representation

\[ g(x^{n+1}, u^{n+1}) = F(x^{n+1}) + Acc(x^{n+1}, x^n) + Q(x^{n+1}, u^{n+1}) = 0 \]

- Can be written as a linear system:

\[
\begin{align*}
x^{n+1} &= A(t)x^n + B(t)u^n \\
y^{n+1} &= C(t)x^{n+1} + D(t)u^n
\end{align*}
\]

- Building matrices A, B, C and D:
  - Need access to simulator code (to get Jacobians and states)
  - Storage state and Jacobian snapshots (very large memory request for real applications) \(\Rightarrow\) NEED MODEL REDUCTION!


Offline → Training runs
- Run fine scale simulator using a set of training controls
- Store snapshots of state and Jacobian

\[ X = [X_1 \ X_2 \ \cdots \ X_n] \quad X = \begin{bmatrix} X_p \\ X_{Sw} \end{bmatrix} \]

Online → Given \( x^n \) and \( u^n \), would like to calculate \( x^{n+1} \)
- Linearize around timestep \( n \)
- Choose the training run snapshot \( x^i \) that is closest to \( x^n \)
- Use the snapshots and Jacobian stored for \( i \) and \( i+1 \) to calculate \( x^{n+1} \)

\[
x^{n+1} = x^{i+1} - (J^{i+1})^{-1} \left[ F^{i+1} + Acc^{i+1} + \frac{\partial Acc^{i+1}}{\partial x^i} (x^n - x^i) + Q(x^{i+1}, u^{n+1}) \right]
\]

Take \( SVD(X) \) → POD/PCA → Latent/Reduced space: \( z = \Phi x \)

\[
z^{n+1} = z^{i+1} - (J_r^{i+1})^{-1} \left[ F_r^{i+1} + Acc_r^{i+1} + \left( \frac{\partial Acc^{i+1}}{\partial x^i} \right)_r (z^n - z^i) + Q_r(x^{i+1}, u^{n+1}) \right]
\]

Model Reduction → E2CO – Embed to control and observe

\[
\begin{align*}
    x^{n+1} &= A(t)x^n + B(t)u^n \\
y^{n+1} &= C(t)x^{n+1} + D(t)u^n
\end{align*}
\]

Projection

\[
\begin{align*}
    z^{n+1} &= A_r(t)z^n + B_r(t)u^n \\
y^{n+1} &= C_r(t)z^{n+1} + D_r(t)u^n
\end{align*}
\]

POD, POD-DEIM, POD-TPWL, DMD

Our Hypotheses → system ID

Use a “proxy” model to calculate these matrices to predict model output based on dynamical evolution of the states

- Proposed for modeling learn and control of a non-linear dynamical system using as input raw pixel images.
- It uses a convolutional autoencoder coupled with a control linear system approach in order to be able to predict the time evolution of the system state.

Embed to Control – E2C – How to Train?

Input: State Snapshots
• Pressure
• Water Saturation

Recall TPWL

Offline → Training runs
• Run fine scale simulator using a set of training controls
• Store snapshots of state and Jacobian

\[ X = [X_1 \ X_2 \ \cdots \ X_n] \quad X = \begin{bmatrix} X_p \\ X_{sw} \end{bmatrix} \]

Physics – porous media flow?

3 Loss functions:
\[ \mathcal{L}_{rec} \rightarrow \text{reconstruction} \]
\[ \mathcal{L}_{pred} \rightarrow \text{prediction} \]
\[ \mathcal{L}_{trans} \rightarrow \text{transition} \]
Proposition 1 - E2C to build a reservoir simulation proxy

Physical Loss Function

\[
(L_{\text{flux,rec}})_i = K \{ \| \nabla p_t - \nabla \hat{p}_t \|_2 \}^2_i, \\
(L_{\text{flux,pred}})_i = K \{ \| \nabla p_{t+1} - \nabla \hat{p}_{t+1} \|_2 \}^2_i.
\]

Predicted fluid flow between grid blocks has to be close to the input values.
Proposition 2 - E2C to build a reservoir simulation proxy

Physical Loss Function – 2 Phase

\[
(L_{\text{flux2Ph,rec}})_i = K \left\{ \left\| K_{ro}(S_{w,t}) \nabla p_t - K_{ro}(\hat{S}_{w,t}) \nabla \hat{p}_t \right\|^2 + \left\| K_{rw}(S_{w,t}) \nabla p_t - K_{rw}(\hat{S}_{w,t}) \nabla \hat{p}_t \right\|^2 \right\}_i,
\]

\[
(L_{\text{flux2Ph,pred}})_i = K \left\{ \left\| K_{ro}(S_{w,t+1}) \nabla p_{t+1} - K_{ro}(\hat{S}_{w,t+1}) \nabla \hat{p}_{t+1} \right\|^2 + \left\| K_{rw}(S_{w,t+1}) \nabla p_{t+1} - K_{rw}(\hat{S}_{w,t+1}) \nabla \hat{p}_{t+1} \right\|^2 \right\}_i.
\]
Proposition 3 - E2CO – Embed to control and observe

- Calculate output data from the reduced space state

\[
\begin{align*}
\Delta t_t & \rightarrow \text{Transition} \\
A_t, B_t & \rightarrow \hat{z}_{t+1} = A_t z_t + B_t u_t \\
\approx & \hat{y}_{t+1} = C_t \hat{z}_{t+1} + D_t u_t \\
\approx & y_{t+1} \\
\end{align*}
\]
Data Set

- 60 x 60 x 1 Oil-Water Model (CMG Imex)
- Wells:
  - 5 producers (controlled by BHP)
  - 4 injectors (controlled by Injection Rate)
- Fixed permeability field
- Field operating through 2000 days changing controls at each 100 days
- Controls: We generated 300 control sets, where each control set will have the information for 20 timesteps (full simulation time)
  - Producers: BHP $\sim U(260, 275) \text{ kgf/cm}^2 \ (U(3700, 3900) \text{ psi})$
  - Injector:
    - for each control set $(i)$ we sample $q_{\text{inj,base}} \sim U(300, 950) \text{ m}^3/\text{d} \ (U(1900, 6000) \text{ bbl/d})$
    - for each timestep $(t)$ we sample a $q_{\text{inj, pert}} \sim U(-80, 90) \text{ m}^3/\text{d}$
    - $q_{\text{inj}}(i, t) = q_{\text{inj, base}}(i) + q_{\text{inj, pert}}(i, t)$

Adapted from Jin, et. al 2020
Control set examples
NN implementation

• We built E2C and E2CO network using Python and Tensorflow v 2.4/Keras
• Using **GPUs** during the training procedure reduces its time by a factor of 10

<table>
<thead>
<tr>
<th>Processor</th>
<th># of Cores</th>
<th>Memory</th>
<th>100 Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CPU Intel Xeon CPU E4-1660 v3 @ 3.00 GHz</td>
<td>8</td>
<td>64 GB</td>
<td>465 min</td>
</tr>
<tr>
<td>GPU NVIDIA GeForce GTX 1050</td>
<td>640 Cuda</td>
<td>2 GB</td>
<td>152 min</td>
</tr>
<tr>
<td>2 GPUs NVIDIA Tesla K80 (TAMU HPRC Terra)</td>
<td>4,992 Cuda</td>
<td>24 GB</td>
<td>142 min</td>
</tr>
<tr>
<td>2 GPUs NVIDIA Tesla V100 (TAMU HPRC Ada)</td>
<td>10,240 Cuda</td>
<td>32 GB</td>
<td>38 min</td>
</tr>
</tbody>
</table>

• Optimization algorithm:
  - Adam with learning rate: 1E-4
  - 100 epochs
  - Batch size: 4
Results

Next steps - uncertainty

\[
\begin{align*}
z^{n+1} &= A(t, \theta)z^n + B(t, \theta)u^n \\
y^{n+1} &= C(t, \theta)z^{n+1} + D(t, \theta)u^n
\end{align*}
\]

Reservoir Characterization
Discussions + Q&A

- How to “close the loop”?
- Open question: Which ML Architecture should be used?
- **Consensus so far → no magic bullet - there is not a single approach to be used**
- **Iterative process → No “push button” approach so far**

  - Reservoir/Hydraulic fracture characterization and simulation/prediction
    - Data/model compression/reduction
  - How to incorporate physics (Scientific Machine Learning)
    - Borrow ideas from reservoir simulation
    - Other areas
    - Deep Learning
  - **Open to collaborations/ideas/etc**

Harold Vance Department of Petroleum Engineering
Background/ Additional Slides
Part 2

Drilling Automation, Drilling Simulation and Applications of ML in Drilling Processes

By
Eduardo Gildin, Narendra V. and Enrique Z.
Looking ahead - Drilling Dysfunctions Detection

- Drilling sensors at or near Bottomhole Assembly (BHA) generate data at high frequency
- Dysfunctions → are they really bad?
  - Bit Balling, Stick-Slip, Whirl

Can we predict the onset of dysfunctions?

Optimal Control parameters?
- Rate of Penetration (ROP)
- Weight on bit (WOB)
- RPM
Drilling Automation – Instrumented Miniaturized Drilling Rigs (test-beds)

Source: TAMU Drillbotics Team/NASA Team
Drilling Automation – Drilling on Mars/Moon (Tensegrity-based Drilling Rig)

Source: TAMU Drillbotics Team/NASA Team
• Other dysfunctions could be developed following similar methodology with the availability of big data consisting of high-frequency downhole drilling information.

• Develop a real-time computer advisory system to help drillers make more effective decisions and optimize the Rate of Penetration (ROP) achieved during geothermal drilling operations

  ▪ Looking ahead the bit – identify lithology X Dysfunctions X drilling performance