DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Derivative-Free Optimization for Automatic Tuning of an Oil Well Simulator

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Willian de Medeiros Galvani

Florianópolis, Março de 2017

Derivative-Free Optimization for Automatic Tuning of an Oil Well Simulator

Willian de Medeiros Galvani

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Eduardo Camponogara

For even the very wise cannot
see all ends.

J. R. R. Tolkien

Resumo

Para maximar a produção, e assim os lucros, um campo de petróleo precisa operar em um ponto ótimo.

Encontrar esse ponto ótimo não é uma tarefa fácil. Para testar novos modelos, sistemas de controle e de predição, são utilizados simuladores.

Mas tais simuladores são complexos por si só, e precisam ser sintonizados regularmente para refletir as mudanças nas reservas de petróleo com o tempo. Esta sintonia pode tomar tempo e ser trabalhosa para o engenheiro responsável, além de não ser ótima.

Nossa solução é utilizar um método de Otimização Sem-Derivadas para sintonizar o Marlim, simulador construído in-house pela Petrobras, minimizando as distancias entre as variaveis reais medidas e os resultados simulados.

Palavras-chave: otimização sem derivada, poços de petróleo, simulação, sintonia automática

Abstract

In order to maximize production, hence profit, an oilfield must operate at an optimum levels.

Finding these operating points is not an easy task. In order to test new models, control and predictions systems, simulators are used.

But such simulators are complex themselves, and must be tuned regularly to reflect the changes on the oilfields with time. This tuning itself can be time-consuming and cumbersome for the responsible engineer, and not optimal.

Our approach is to use a Derivative-Free Optimization method to tune the Petrobras' in-house built Marlim simulator by minimizing the distance between simulated and real-world gathered data.

(RESULTS HERE)

(CONCLUSION HERE)

Results: What's the answer? Specifically, most good computer architecture papers conclude that something is so many percent faster, cheaper, smaller, or otherwise better than something else. Put the result there, in numbers. Avoid vague, hand-waving results such as "very", "small", or "significant." If you must be vague, you are only given license to do so when you can talk about orders-of-magnitude improvement. There is a tension here in that you should not provide numbers that can be easily misinterpreted, but on the other hand you don't have room for all the caveats.

Conclusions: What are the implications of your answer? Is it going to change the world (unlikely), be a significant "win", be a nice hack, or simply serve as a road sign indicating that this path is a waste of time (all of the previous results are useful). Are your results general, potentially generalizable, or specific to a particular case?

Keywords: derivative-free optimization, oil well, simulation, automatic tuning

Lista de ilustrações

Lista de tabelas

Sumário

1 Introduction

1.1 Motivation

In order to get the maximum financial returns from an oilfield, the processes must be always at a point of maximum performance. Since the oilfield systems are complex and change with time, simulators are used to approximate the real world system. In order to reproduce the observed measurements properly, the simulators are tuned routinely by the process engineer. This tuning is often a cumbersome and time-consuming task.

A well tuned simulator can be user for process modeling and prediction by control and optimization algorithms.

How long it takes to tune the simulator reflects on how long the production can be optimized, thus potentially affecting the oilfields revenue.

- Oil well simulators are tuned routinely in order to reproduce observed measurements, a process that can be cumbersome and time-consuming.
- The simulator of oil wells should be adequately tuned before it can be used for process modeling and prediction by control and optimization algorithms.
- The degree of accuracy can have major production and economic impact.

1.2 Goals

The ultimate goal is to optimize the production and profit. We aim to identify, implement, and test suitable derivative-free optimization methods to tune the Marlim simulator. The methods being studied are the Simplex, Baricenter, and OrthoMADS aswell as NOMAD's implementation. We hope the system helps relieving the engineer's burden of tuning the simulator, moving him to a overseeing task.

- Study and identify suitable derivative-free optimization methods for an oil well simulator.
- Implement and test derivative-free optimization algorithms for the Marlim simulator, an in-house simulator developed by Petrobras.

1.3 Report Structure

This report is organized in four main sections. On section 2, an overview of the oilfield systems and how they work is given, explaining the structure of an Floating Production, Storage and Offloading (FPSO) unit, as well as each of their main components. As the systems are presented, the variables used for control or feedback on next sections will be highlighted.

On section 3, we give a general presentation of the Derivative-Free optimization concept, showcasing the utilized methods.

On section 4, we present the application structure, how everything connects, the experiments setup and conditions, input and output data, and discuss the results.

Section 5 contains the experiments conclusions.

2 Oil Well Modeling and Simulation

2.1 Offshore Oil Production

The oilfields in question can have a range of different layouts. They will be presented and explained from bottom (of the sea) to top.

2.1.0.1 Wellbore

The "hole"itself connecting to the oil and gas reserves. It's design and architecture varies wildly with it's location (on-shore, off-shore) the composition of the reserves interfaces(rock properties, such as porosity).

2.1.0.2 Wellhead

The most upstream component, the wellhead sits on top of the wellbore, oil and gas reserves. It's the interface between the extraction systems and the reservoir.

2.1.0.3 Choke

It's used to manipulate the flow out of the wellhead.

2.1.0.4 Manifold

2.1.0.5 Riser

2.1.0.6 Flow Lines

2.1.0.7 ...

- Discuss in the general terms the structure of an offshore oilfield, which consists of subsea oil wells with risers connecting to a production platform.
- Discuss in general terms the surface processing facilities, valves, separator, compressor, flare and exportation.

2.2 Satellite Oil Wells

Discuss the structure of offshore oil wells that operate with continuous lift-gas injection (gas-lift).

2.3 Oil Well

- Give a brief presentation about Marlim and/or Pipesim.
- Present some curves of oil wells modeled in Pipesim.

3 Derivative-Free Optimization

3.1 Visão Geral

- Motivate the use of derivative-free optimization. Frequentemente em problemas de engenharia os modelos utilizados são aproximações simplificadas da realidade, e em outros casos completamente desconhecidos.

  Métodos de otimização sem derivadas são adequados em casos aonde o modelo matemático é não-explicito, custoso ou as derivadas não estão disponíveis, devido a inexistência do modelo explícito ou presença de ruídos impossibilitando a estimação das derivadas.

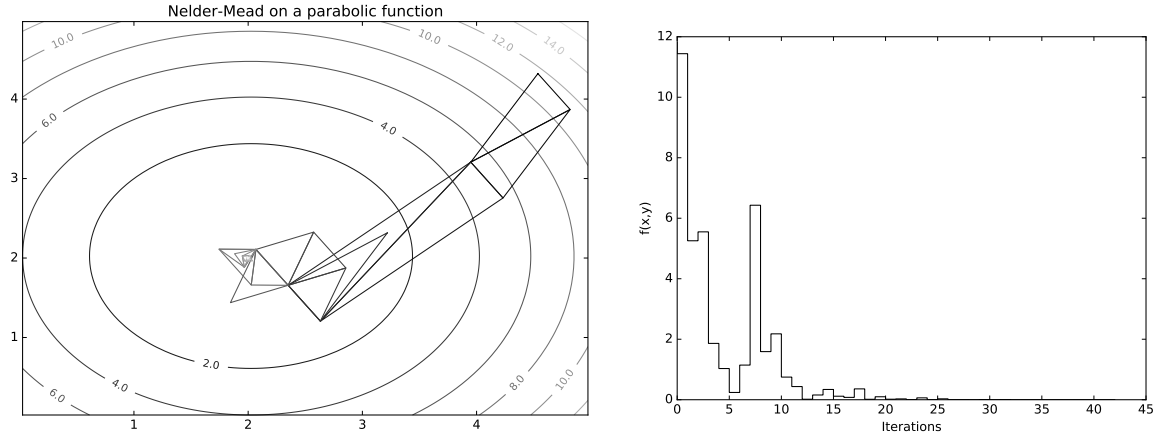
  Os métodos de otimização sem derivadas procuram encontrar minimos computando o menor número possível de pontos do problema, de modo a minimizar também o tempo de execução da otimização.

- Give a general presentation/introduction to derivative-free optimization

3.2 Método do Simplex de Nelder-Mead

Também conhecido como Downhill Simplex Method, ou Amoeba Method, consiste em utilizar um poligono com n+1 vértices em n dimensões que se expande, contraí, ou reflete de modo a se mover em direção ao gradiente da função objetivo.

Um exemplo de algoritmo Downhill Simplex é o seguinte:



(a) Passos do Nelder-Mead (b) Convergência do Nelder-Mead

Figura 1 – Nelder-Mead aplicado em um parabolóide.

Algorithm 1 Nelder-Mead's Downhill Simplex

Require: $X = (x_1, ..., x_{n+1})$ pontos de teste:
while Não Convergiu **do**
  ordenar X; $(f(x_1) \leq f((x_2) \leq \cdots \leq f(x_{n+1}))$
  Calcular o centroide x_0 de $(x_1, ..., x_{n+1})$
  Calcular x_r refletido: $x_r = x_0 + \alpha(x_0 - x_{n+1})$
  (Reflecção)
  if $f(f(x_1) < f(x_r) < f(x_n)$ **then**
    $x_{n+1} \leftarrow x_r$
    Continue
  end if
  (Expansão)
  if $f(f(x_r) < f(x_1)$ **then**
    Calcular ponto expandido $x_e = x_0 + \gamma(x_r - x_0)$
    if $f(x_e) < f(x_r)$ **then**
      $x_n \leftarrow x_e$
    else
      $x_n \leftarrow x_r$
    end if
    Continue
  end if(Contração)
  Computar o ponto contraído $x_c = x_0 + \rho(x_{n+1} - x_0)$
  if $f(x_c) < f(x_{n+1})$ **then**
    $x_{n+1} \leftarrow x_c$
    Continue
  end if(Contração)
  $x_i \leftarrow x_1 + \sigma(x_i - x_1), i = 2 \ldots n + 1$
end while

3.3 Baricenter Method

3.4 OrthoMADS

Os métodos MADS (Mesh Adaptative Direct Search) são uma classe de métodos baseados ...

A cada iteração k são executados dois passos de busca, Search e Poll, analisando a feasibility e valor da função. O objetivo de cada nova iteração é encontrar um ponto $f(x) < f(x_k)$, aonde x_k é o melhor ponto encontrado até a iteração atual.

As buscas são feitas sempre em uma grade, definida por:

$M_k = \{x + \Delta_k^m Dz : x \in V_k, z \in \mathbb{N}^{nD}\} \subset \mathbb{R}^n$ Aonde M_k é o conjunto de pontos da grade, x é o ponto mínimo atual, Δ_k^m é o parametro de tamanho da malha, D é uma matrix $\mathbb{R}^{n \times n_D}$ composta por n_D direções que definem um conjunto gerador position no $\mathbb{R}^n$. Para o OrthoMads e LtMads, D é simplesmente definida como $[I_n - I_n]$ aonde I_n é a

matriz identidade de dimensões n.

O passo search pode ser qualquer tipo de heurística que escolha um ponto mais adequado da malha para tentar acelerar a convergência.

O passo poll é a parte mais importante do método, que garante sua convergência. A cada iteração k os pontos a serem utilizados são definidos por:

$$P_k = \{x_k + \Delta_k^p d : d \in D_k\} \subset M_k$$

Aonde x_k é o ponto atual e cada coluna de D_k é formada por combinações inteiras das colunas de D de forma a criar um conjunto gerador positivo. Δ_k^p é o *parametro de tamanho de poll*.

Ambos LtMads e OrthoMads utilizam um parametro ℓ_k chamado de *indíce de malha* para atualizar os parametros de tamanho de poll e search de acordo com esta lógica:

$$\Delta_k^p = 2^{-\ell k} \text{ e } \Delta_k^m = min\{1, 4^{-\ell_k}\} \tag{3.1}$$

A cada nova iteração, se em uma iteração um novo *incumbente* não é encontrado, ela é dita mal sucedida, e $\ell_{k+1} \leftarrow \ell_k + 1$ (reduzindo Δ_k^m e Δ_k^p), por outro lado, se for encontrado um novo *incumbente*, a iteração é dita bem sucedidade, e $\ell_{k+1} \leftarrow \ell_k - 1$ (aumentando Δ_k^m e Δ_k^p). Devido a 3.1, no caso de uma iteração mal-sucedida o parametro de poll diminui mais rápido que o de malha, de modo a permitir o uso de muitos mais pontos, refinando a malha.

A diferença entre o OrthoMads e LtMads se dá na geração da base D_k. O LtMads utiliza uma matriz triangular inferior para a geração da base, fazendo permutações entre os elementos e completando ela em uma base maximal ou minimal, sem garantir ortogonalidade entre as direções, de modo que os ângulos entre as direções podem ser grandes, causando grandes cones de espaço não explorado. Já o OrthoMads utiliza uma base maximal definida por $[H_k - H_k]$, aonde as colunas de H_k formam um base ortogonal de $\mathbb{R}^n$. Além disso, os as direções de D_k são inteiras, de modo que os pontos gerados estão automaticamente contidos na malha definida por $D = [I_n - I_n]$.

Para a geração de D_k, o OrthoMads utiliza a sequencia pseudo-aleatoria de Halton, que cobre mais uniformemente o espaço que uma sequência aleatória real, para gerar vetores u_t.

A saída da sequência de Halton, no entanto, não respeita as restrições impostas pela malha. É necessário arrondar, escalar, e rotacionar o vetor u_t. O indíce ℓ é utilizado para transformar a direção u_t na *direção ajustada de Halton* $q_{t,\ell} \in \mathbb{Z}^n$, uma direção cuja norma é próxima a a $2^{\frac{|\ell|}{2}}$

HERE, BLACK MAGIC IS USED TO MAKE THE VECTOR ALIGN TO THE MESH.

Para definir $q_{t,\ell}$, primeiramente são definidas duas funções baseadas na t-ésima direção de Halton u_t:

$$q_t(\alpha) = \text{round}\left(\alpha \frac{2u_t - e}{||2u_t - e||}\right) \in \mathbb{Z}^n \cap \left[-\alpha - \frac{1}{2}, \alpha + \frac{1}{2}\right]^n$$

Aonde *round* é a operação arrendodar para cima ($round(0,5) = 1$, $round(-0,5) = -1$) e $\alpha \in \mathbb{R}_+$ é um fator de escala.

Desta forma, temos um problema de otimização, precisamos encontrar um $\alpha_{t,\ell}$ tal que $||q_t(\alpha_{t,\ell})$ seja o mais próximo possível de $2^{\frac{|\ell|}{2}}$ sem ultrapassá-lo.

$$\begin{aligned} \alpha_{t,\ell} \in \underset{\alpha in \mathbb{R}_+}{argmax} & ||q_t(\alpha)|| \\ \text{s.t.} & ||q_t(\alpha)|| \leq 2^{\frac{|\ell|}{2}} \end{aligned}$$

O problema pode ser resolvido facilmente, já que que os degraus da função $||q_t(\alpha)||$ acontecem em todos os α no conjunto

$$\left\{ \frac{(2j+1)||2u_t - e||}{2|u_t^i - e|} : i = 1, 2, \ldots, n, j \in \mathbb{N} \right\}$$

De forma que o problema pode ser solucionado varrendo os pontos do conjunto.

Com um vetor normalizado e na malha, $q \in \mathbb{Z}^n$, é necessário transformá-lo em uma base ortonormal de $\mathbb{R}^n$. Para isto é utilizada a transformação de Householder:

$$H = ||q^2||(I_n - 2vv^T), \ \ onde\ v = \frac{q}{||q||} \tag{3.2}$$

ALGO MAIS DEVE SER ESCRITO AQUI, MAS OQ?

4 Application and Results

4.1 Marlim (or Pipesim) Simulator

4.2 Experiments

4.3 Discussion

5 Conclusion

Apêndices

APÊNDICE A – Revisão de Probabilidade

Anexos

ANEXO A – Title of Appendix A