Universidade Federal do Rio de Janeiro
Escola Politécnica

# IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO PARA A SIMULAÇÃO DA CIRCULAÇÃO DE UM *KICK* DE GÁS PELO MÉTODO DO SONDADOR

Raphael Pádua Santos

Projeto de Graduação apresentado ao Curso de Engenharia de Petróleo da Escola Politécnica, Universidade Federal do Rio de Janeiro como parte dos requisitos necessários à obtenção do título de Engenheiro de Petróleo.

Orientador: Paulo Couto

Co-orientador: Roni Abensur Gandelman

RIO DE JANEIRO

Março de 2013

IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO PARA A SIMULAÇÃO DA CIRCULAÇÃO DE UM *KICK* DE GÁS PELO MÉTODO DO SONDADOR

Raphael Pádua Santos

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE PETRÓLEO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE PETRÓLEO.

Examinado por:

Prof. Paulo Couto, Dr.Eng.

Prof. Virgílio José Martins Ferreira Filho, D. Sc.

Eng. Roni Abensur Gandelman, M.Sc.

Prof.Heitor Lima, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

MARÇO de 2013

Santos, Raphael Pádua

Implementação de um Modelo Numérico para a Simulação da Circulação de um *Kick* de Gás pelo Método do Sondador/ Raphael Pádua Santos – Rio de Janeiro: UFRJ/Escola Politécnica, 2013.

XXI, 60 p.: il.; 29,7 cm.

Orientador: Paulo Couto

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Petróleo, 2013.

Referências Bibliográficas: p. 58 - 60.

1. Simulador de *Kick*. 2. Controle de Poço. 3. Método do Sondador. 4. *Kick* de Gás I. Couto, Paulo. II. A. Gandelman, Roni. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Petróleo. III. Implementação de um Modelo Numérico para a Simulação da Circulação de um *Kick* de Gás pelo Método do Sondador.

“Infinite dreams I can’t deny them

Infinity is hard to comprehend”

(Steve Harris, Iron Maiden)

# Agradecimentos

Em primeiro lugar, gostaria de agradecer a minha mãe, Vera Luiza, por tudo. Não há palavras para agradecer todo apoio, ajuda, carinho e dedicação ao longo de toda minha vida. Sem sua constante presença, não seria hoje a pessoa que sou, e certamente não teria alcançado mais esse objetivo. Obrigado por ser simplesmente a melhor mãe do mundo.

Além disso, gostaria de agradecer ao meu pai, Mauricio Santos, não somente por ter me aconselhado e ajudado a crescer como profissional e pessoa, mas também por ser um pai fantástico. Apesar da distância, tudo o que você sempre representou para mim faz parte da minha essência. Agradeço também aos meus avós, tios e irmãos por todo carinho, incentivo e confiança depositados ao longo dos anos.

Agradeço, também, ao meu professor e orientador Paulo Couto pela ajuda e orientação durante a confecção deste trabalho, assim como por toda a paciência e disponibilidade para tirar todas as dúvidas que apareceram durante essa longa jornada.

Gostaria de agradecer também ao engenheiro e amigo Roni Abensur Gandelman por ter me dado a minha primeira chance dentro da indústria como estagiário no CENPES, permitindo meu crescimento como profissional e o início deste trabalho. Da mesma forma que agradeço toda a competência, orientação e ajuda ao longo de todo esse projeto, sem as quais, seria impossível terminar este trabalho com qualidade. Além disso, agradeço também por toda a amizade, as brincadeiras e as conversas sobre Krav Magá que com certeza ajudaram dentro desse processo nos momentos de dificuldade.

Também agradeço aos demais colegas de trabalho no CENPES por todo incentivo e apoio, principalmente ao André Leibsohn Martins por ter sido um mentor dentro da empresa e me passado milhões de trabalhos diferentes e complicados, me ensinando assim, a crescer como engenheiro e a conseguir distribuir melhor meu tempo. Agradeço, também, ao Carlos Eduardo Fontes, por ter me dado a chance de continuar meu projeto dentro da ESSS, com todo o apoio e estrutura necessários para um trabalho de qualidade, assim como ao Rafael March Castaneda por toda ajuda e dicas durante a elaboração do programa e os mais diversos debates sobre o Vasco.

Agradeço, também, a todos os meus amigos, em especial ao Milton Mikio, ao Matheus Moreira e ao Leandro Galves, sem os quais esses 5 anos não teriam tido a menor graça. Todos os momentos passados juntos, desde as incontáveis horas de estudo até as, também incontáveis, horas de brincadeiras e piadas foram fundamentais para que fosse possível aguentar essa longa jornada que foi cursar Engenharia do Petróleo na UFRJ. Sei que fiz amigos para a vida toda, e espero que toda esse carinho e amizade seja mantido entre todos os membros da nossa turma. Devo agradecer também a minha namorada, Rayssa Campos, por todo o carinho e paciência, e por ter suportado junto comigo todas as dificuldades que esses anos de faculdade apresentaram.

Por fim, agradeço, também, a mim mesmo por nunca ter desistido, mesmo quando tudo parecia impossível e as chances de sucesso eram mínimas. Apesar de todas as dificuldades, foi uma jornada maravilhosa que agora se acaba para dar início a um novo recomeço. Obrigado, Raphael.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Petróleo.

# IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO PARA A SIMULAÇÃO DA CIRCULAÇÃO DE UM *KICK* DE GÁS PELO MÉTODO DO SONDADOR

Raphael Pádua Santos

Março/2013

Orientador: Paulo Couto

Co-orientador: Roni Abensur Gandelman

Curso: Engenharia de Petróleo

Dentro da indústria do petróleo, um dos temas de maior preocupação e estudo é a contenção de forma rápida, segura e efetiva de qualquer influxo indesejado do reservatório para dentro do poço, evitando-se a assim a ocorrência de um *blowout*. Este trabalho apresenta uma modelagem feita em *Python* com o objetivo de simular um *kic*k de gás e o processo de circulação do mesmo através do Método do Sondador. Diversas considerações foram feitas com relação à geometria do poço, aos cálculos de perda de carga, ao fluido utilizado para perfuração, visando resultados mais próximos do esperado. Da mesma forma que casos hipotéticos foram simulados previamente para calibrar o modelo com os resultados teóricos da literatura.

*Palavras-chave:* Simulador de *Kick*, Controle de Poço, Método do Sondador, *Kick* de Gás.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Petroleum Engineer.

# IMPLEMENTATION OF NUMERICAL MODEL TO SIMULATE THE CIRCULATION OF A GAS KICK BY THE DRILLER'S METHOD

Raphael Pádua Santos

March/2013

Advisor: Paulo Couto

Co-advisor: Roni Abensur Gandelman

Course: Petroleum Engineering

In the oil industry, one of the issues of greatest concern and study is the contention of any undesired influx from the reservoir into the well in a safe and effective manner avoiding, thus, the occurrence of blowouts. This study presents a model developed in *Python Language* in order to simulate a well under a gas kick and the circulation process by the Driller's Method. Several considerations were made regarding the well geometry, the pressure drop calculations, the fluid used for drilling, seeking better overall results. With the same idea, hypothetical cases were simulated aiming to calibrate the model with the theoretical results available in the literature.

*Keywords:* Kick Simulator, Well Control, Driller's Method, Gas Kick.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Símbolos

| | | SI |
|---|---|---|
| A | Área (interna ou anular) | $m^2$ |
| $C_a$ | Capacidade do anular | $m^3/m$ |
| $c_g$ | Compressibilidade do gás | $Pa^{-1}$ |
| *D* | Profundidade | m |
| $d$ | Diâmetro interno ou hidráulico | m |
| $d_{bi}$ | Diâmetro da região bifásica | m |
| $d_{e,c}$ | Diâmetro externo do tubo de perfuração | m |
| $d_h$ | Diâmetro hidráulico da região anular | m |
| $d_i$ | Diâmetro interno da coluna | m |
| $d_{i,r}$ | Diâmetro interno do revestimento | m |
| $d_t$ | Diâmetro do tubo | m |
| $D_v$ | Profundidade vertical do poço | m |
| $f$ | Fator de atrito (interno ou anular) | - |
| $f_{bi}$ | Fator de atrito bifásico | - |
| FCP | Pressão Final de Circulação | Pa |
| g | Gravidade | $m/s^2$ |
| H | *Holdup* do líquido | - |
| $h$ | Altura do trecho | m |
| $h_{bi}$ | Comprimento do trecho da região bifásica | m |
| $h_e$ | Espessura | m |
| $h_{kick}$ | Altura do kick | m |
| *ICP* | Pressão de Circulação Inicial | Pa |
| *k* | Permeabilidade | $m^2$ |

| | | |
|---|---|---|
| L | Comprimento da tubulação | m |
| $L_E$ | Limite de escoamento | kg/100m² |
| MSM | Margem de segurança de manobra | kg/m³ |
| $P_a$ | Pressão de fechamento do anular | Pa |
| $P_b$ | Pressão no fundo do poço | Pa |
| $P_{ck,max}$ | Máxima pressão lida no manômetro do *choke* (gás na superfície ) | Pa |
| $P_{dp}$ | Pressão de fechamento da coluna de perfuração | Pa |
| $P_i$ | Pressão inicial | Pa |
| $P_{ks}$ | Pressão da bomba na velocidade reduzida de circulação | Pa |
| $P_w$ | Pressão no fundo do poço | Pa |
| Q | Vazão de injeção | m³/s |
| $q_w$ | Vazão de entrada do kick | m³/s |
| $r_w$ | Raio do poço | m |
| *SIDPP* | Pressão de fechamento da coluna de produção | Pa |
| $U_{gl}$ | Velocidade superficial do líquido | m/s |
| $U_{gs}$ | Velocidade superficial do gás | m/s |
| $U_{sl}$ | Velocidade de escorregamento | m/s |
| v | Velocidade da manobra | m/s |
| $v_f$ | Velocidade do fluido | m/s |
| $V_g$ | volume de gás | m³ |
| $v_g$ | Velocidade de ascensão da bolha | m/s |
| $V_k$ | Volume do *kick* medido no tanque de lama | m³ |
| $V_m$ | Volume de lama necessário para gerar pressão hidrostática de 50 psi | m³ |
| $v_m$ | Velocidade da mistura | m/s |
| $v_{sl}$ | Velocidade de escorregamento | m/s |
| $\gamma$ | Constante de Euler | - |
| $\Delta P$ | Pressão de pistoneio | Pa |

| | | |
|---|---|---|
| $\Delta P_h$ | Ganho de pressão hidrostática no fundo do poço | Pa |
| $\Delta t$ | Passo de tempo | s |
| $\lambda$ | Fração de vazio | - |
| $\mu$ | Viscosidade do fluido | Pa.s |
| $\mu_g$ | Viscosidade do gás | Pa.s |
| $\mu_p$ | Viscosidade plástica do fluido | Pa.s |
| $\rho$ | Massa específica do fluido | kg/m³ |
| $\rho_f$ | Gradiente de pressão do *kick* | kg/m³ |
| $\rho_g$ | Massa específica do gás | kg/m³ |
| $\rho_{km}$ | Massa específica da nova lama usada no poço | kg/m³ |
| $\rho_m$ | Gradiente de pressão da lama | kg/m³ |
| $\rho_m$ | Massa específica da mistura | kg/m³ |
| $\gamma$ | Taxa de deformação interna ou anular | - |
| $\varphi$ | Porosidade | - |

# 1. INTRODUÇÃO

O controle de poço é um tópico extremamente importante nas operações de exploração e explotação de óleo e gás, uma vez que diversos aspectos ambientais, econômicos e de segurança estão envolvidos. Por isso, é necessário que seja realizado de uma forma altamente eficaz. Todo o estudo voltado para as operações de controle de poço visa estabelecer parâmetros relevantes, os quais devem ser monitorados para que seja possível evitar a ocorrência de influxos de fluidos provenientes da formação para o poço, da mesma forma que os métodos a serem utilizados para combater esse influxo, caso ele ocorra. Segundo AIRD (2009), existem 3 níveis de procedimentos que devem ser executados sobre a pressão das formações perfuradas para evitar esses influxos:

1) Controle Primário: Ação da pressão hidrostática sobre a rocha, isto é, a pressão do fluido de perfuração, deve ser mantida superior à pressão existente nos poros da rocha a ser perfurada. O *kick*, fluxo inesperado e indesejado de fluido da formação para o poço, ocorre quando esse primeiro controle não é satisfatório;
2) Controle Secundário: Conjunto de equipamentos de segurança a ser utilizado quando o controle primário é perdido. Nessa etapa o *kick* já ocorreu e se quer evitar o *blowout*, ou seja, fluxo descontrolado de fluido da formação para a superfície;
3) Controle Terciário: Caso o controle do poço a nível secundário não possa ser mantido, um *blowout* irá ocorrer e o controle da formação só poderá ser conseguido através de medidas especiais.

Além disso, para que as decisões que venham a ser tomadas sejam corretas e o controle de poço eficaz, é fundamental ter conhecimento não só das propriedades do fluido que está sendo usado na circulação, mas também do próprio fluido invasor. Para isso, é necessário que seja feita uma boa modelagem desse influxo para que haja certeza do tipo de *kick* (água, óleo ou gás), assim como prever o comportamento das pressões, vazões e volumes desenvolvidos ao longo da circulação. Essa modelagem se torna ainda mais importante na presença de um *kick* de gás, pois este é um fluido altamente compressível, que pode facilmente colocar o poço fora de controle, transformando-se, eventualmente, em um *blowout*. Adicionalmente, dependendo da base do fluido de perfuração, o gás pode se solubilizar e com isso se tornar muito mais difícil de ser detectado na superfície. Outros

trabalhos já foram desenvolvidos visando estudar e modelar situações em que o poço está sendo perfurado com um fluido à base óleo e um influxo de gás ocorre, tais como EKRANN e ROMMETVEIT (1985), e WHITE e WALTON (1990).

Diversos acidentes já ocorreram na indústria do petróleo ao longo dos anos, causando grande prejuízo socioeconômico e ambiental. Entre os mais recentes e de maior repercussão, podemos citar o *blowout* de Macondo e o vazamento no campo de Frade. O primeiro, ocorrido em abril de 2010 (Figura 1), resultou na morte de 11 pessoas e num vazamento de aproximadamente 5 milhões de barris, atingindo uma área de 180 mil km² e sendo classificado como o pior vazamento de óleo no mar dos EUA (MCANDREWS, 2011). Já o acidente no campo de Frade, localizado na região da Bacia de Campos, em novembro de 2011, resultou, segundo o relatório nº 48610.003638/2012-92 da ANP, em um vazamento de 3,7 mil barris de óleo gerando uma multa de aproximadamente 50 milhões de reais para a empresa. Segue abaixo, alguns dos principais *blowouts* em mar ocorridos ao longo da história.

Em 1969, ocorreu um *blowout* que durou 11 dias na plataforma *Union Oil A*, que operava no canal de Santa Barbara, resultando em um vazamento de 80 mil barris de petróleo (GAINES, 1970). Oito anos depois, na plataforma continental da Noruega, ocorreu um *blowout* na plataforma *Ekofisk Bravo*, que só foi controlado 8 dias depois, acarretando em um vazamento aproximado de 200 mil barris (HAEGH e ROSSEMYR, 1980). Dois anos depois, em 1979, ocorreu um *blowout* na plataforma *Sedco 135F* durante uma perfuração na Baía de Campeche no México. O *blowout* só foi controlado 9 meses depois da data do acidente, resultando em um vazamento de 3,7 milhões de barris (ERCO/ENERGY RESOURCES CO. INC, 1982). Por fim, no ano seguinte, 2 *blowouts* marcantes ocorreram, um na plataforma *Hasbah* no Golfo Persa, que durou 8 dias, vazando o equivalente a 100 mil barris de petróleo e matando 19 pessoas (FAKHRO, 1991), e o outro, no poço *Funiwa* número 5 na Nigéria, durou 2 semanas com um vazamento registrado de 200 mil barris (HUTCHFUL, 1985).

**Figura 1 - Plataforma *DeepWater Horizon* em chamas após o *blowout* do Campo de Macondo, Golfo do México. (Fonte: http://www.telegraph.co.uk/finance/markets/questor/9124547/Questor-Gulf-of-Mexico-oil-spill-settlement-leaves-BP-in-a-safer-place.html)**

## 1.1 Motivação

As novas descobertas de jazidas de petróleo em cenários cada vez mais desafiadores, com lâminas d'água mais profundas e janelas operacionais mais estreitas, aumentam ainda mais os grandes riscos operacionais durante toda a etapa de perfuração, assim como os custos para a prevenção e contenção de quaisquer problemas que possam vir a ocorrer. Por isso, uma operação de controle de poço eficiente e segura é indispensável.

Para tal, a possibilidade de simular e estudar casos similares com a realidade é de vital importância. A simulação das operações de controle de poço fornece um maior embasamento técnico e teórico, visando assim aumentar a experiência dos técnicos e engenheiros responsáveis pela sonda. Para que, no futuro, dentro do campo, ao serem confrontados com problemas semelhantes, decisões possam ser tomadas de forma mais rápida, segura e eficaz evitando dessa forma situações que poderiam levar a grandes desastres na indústria do petróleo.

## 1.2. Objetivo

O objetivo desse trabalho é desenvolver um código computacional programado em linguagem *Python* que simule um poço de petróleo, o qual sofre um influxo indesejável de gás e que utilizará do método do sondador para circular o influxo para fora do poço. Dentro disso, o programa auxiliará no estudo do monitoramento das pressões no poço, assim como calculará de forma iterativa a variação da pressão na válvula de *choke* ao longo do tempo até que todo o gás seja retirado do poço sem fraturar a formação e sem permitir que o influxo da formação volte a ocorrer.

## 1.3. Organização do Trabalho

Este trabalho foi dividido de forma a conter 6 capítulos, incluindo esta introdução, e uma seção de anexos no seu final.

O capítulo 2 fornece ao leitor uma breve revisão de alguns conceitos necessários para uma melhor compreensão dos processos de controle de poço. Este capítulo foca 3 grandes tópicos: as principais causas e indícios de um *kick*, os métodos mais comuns de controle de poço e outros modelos previamente propostos para simulação computacional de um *kick*.

No terceiro capítulo, é feita a modelagem do código desenvolvido, explicitando as considerações feitas, assim como explicando a metodologia desenvolvida.

O quarto e quinto capítulos apresentam, respectivamente, os resultados obtidos com o modelo desenvolvido e as conclusões que foram obtidas ao término do trabalho, assim como, sugestões de continuidade para o trabalho.

O sexto e último capítulo possui todas as referências bibliográficas que foram consultadas durante a confecção deste trabalho.

Por fim, dentro da seção de anexos, é possível encontrar todo o código que foi desenvolvido em Python.

# 2. REVISÃO DA LITERATURA

Este capítulo se destina a fazer uma revisão de alguns conceitos fundamentais visando facilitar o entendimento deste trabalho. Os tópicos que serão abordados irão fornecer ao leitor algumas informações básicas e necessárias para que o mesmo ganhe um maior embasamento teórico e possa ficar mais familiarizado com certos detalhes e procedimentos.

## 2.1 Causas de *Kicks*

De forma resumida, durante as operações normais de perfuração, a pressão dentro do poço deve ser maior que a pressão de poros das formações permeáveis expostas, para se evitar a ocorrência de *kicks*. Da mesma forma que deverá ser também menor que a pressão de fratura das mesmas formações, para que não ocorra o fraturamento dessas formações e consequente perda de fluido do poço para as mesmas. Assim sendo, são criadas curvas de pressão máxima e mínima que podem ocorrer durante as operações. Através do estudo e análise dessas curvas é possível elaborar a janela operacional (figura 2) de um poço.

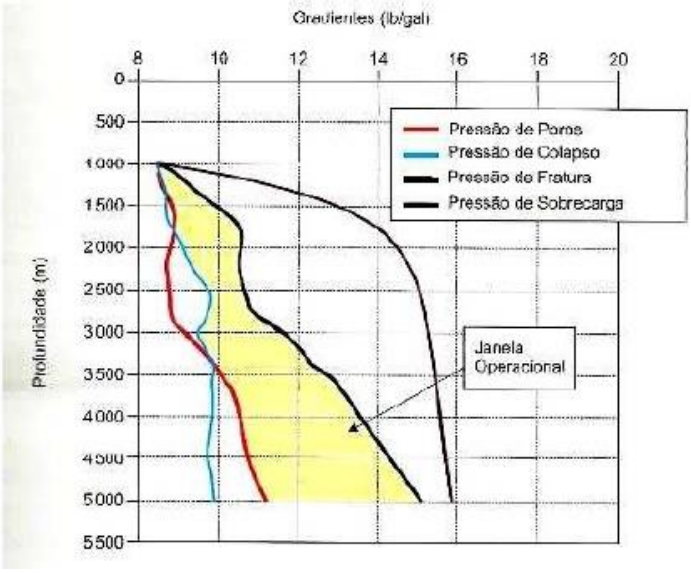


**Figura 2 - Janela operacional de um poço de petróleo. Fonte: ROCHA e AZEVEDO, 2009**

Quando ocorre um decréscimo da pressão hidrostática no poço fazendo com que esta caia abaixo do limite inferior da janela operacional, o poço estará sujeito a ocorrência de um

*kick*. Isto é, um influxo indesejado de fluidos provenientes da formação para o interior do poço. As principais causas de um *kicks* estão, geralmente, relacionadas à redução do nível hidrostático no interior do poço e/ou à redução da massa específica do fluido de perfuração. As principais causas de *kicks* serão brevemente apresentadas a seguir.

### 2.1.1 Não Completar o Poço Durante as Manobras

Sempre que uma manobra de retirada de coluna for realizada, devido ao volume do aço retirado, o nível de fluido dentro do poço irá diminuir. Com isso, se torna necessário enchê-lo com um volume equivalente de fluido de perfuração. A prática usual é manter o poço cheio durante toda a manobra ou completá-lo a cada retirada de três a cinco seções de tubos e a cada seção de comando. Esse enchimento deve ser monitorado através do tanque de manobra, cuja instalação é obrigatória em sondas de perfuração e deve seguir o programa de ataque ao poço, previamente elaborado.

Se o volume de fluido de perfuração para completar o poço é menor que o calculado, pode-se estar caminhando para uma situação de *kick*. Nesse caso, a manobra dever ser interrompida e o poço observado para ver se ele está fluindo (*flow check*). Caso haja fluxo, deve-se fechar o poço imediatamente.

### 2.1.2 Pistoneio

Esse fenômeno consiste na redução da pressão no poço, causada pela retirada da coluna de perfuração. Esse efeito pode se manifestar de duas maneiras:

- Pistoneio mecânico - é a redução do nível hidrostático, causada pela remoção mecânica do fluido de perfuração para fora do poço, devido à restrição no espaço anular (enceramento da broca ou dos estabilizadores, poços delgados, utilização de *packers*, etc). Esse tipo de pistoneio manifesta-se pelo retorno do fluido de perfuração à superfície e por um possível aumento do peso da coluna na sua retirada. A redução da velocidade de retirada da coluna contribui para a redução desse efeito;

- Pistoneio hidráulico - é a redução da pressão no poço, devido à indução de perdas de carga por fricção, através do movimento descendente do fluido de perfuração que irá ocupar o espaço vazio deixado abaixo da broca, durante a retirada da coluna de perfuração. A variação da pressão gerada por esse tipo de pistoneio pode ser calculada pela fórmula:

$$\Delta P = \frac{L\, L_E}{60{,}69(d_{i,r} - d_{e,c})} + \frac{L\, \mu_p\, v}{5574(d_{i,r} - d_{e,c})^2} \qquad (2.1)$$

onde,

$\Delta P$ = pressão de pistoneio, em psi;

$L$ = comprimento da tubulação, em m;

$L_E$ = limite de escoamento, em lb/ 100 ft²;

$\mu_p$ = viscosidade plástica do fluido, em cp;

$d_{i,r}$ = diâmetro interno do revestimento, em pol;

$d_{e,c}$ = diâmetro externo do tubo de perfuração, em pol;

v = velocidade da manobra, em m/ min.

Com isso, adiciona-se uma margem de segurança a massa específica do fluido de perfuração com o intuito de evitar a ocorrência de um *kick* devido ao pistoneio hidráulico. Essa margem pode ser calculada pela fórmula:

$$MSM = 2\, \frac{\Delta P}{0{,}17\ D_v} \qquad (2.2)$$

onde,

*MSM* = margem de segurança de manobra, em lb/gal;

$D_v$ = profundidade vertical do poço, em m.

Além disso, o pistoneio hidráulico pode ser minimizado, reduzindo-se a viscosidade do fluido de perfuração a valores mínimos possíveis antes da manobra e/ou controlando-se a velocidade de retirada da coluna de perfuração. É importante notar que, se o pistoneio é detectado durante a retirada da coluna, a manobra dever ser interrompida e o poço observado para ver se ele está fluindo (*flow check*). Caso haja fluxo o poço deve ser fechado imediatamente.

### 2.1.3 Perda de Circulação

Resulta em um abaixamento do nível de fluido de perfuração no poço, com a consequente redução da pressão hidrostática. Se houver, no poço, uma formação permeável cuja pressão se torne maior que a pressão hidrostática na sua frente, o fluido contido nessa formação invadirá o poço, ocasionando um *kick*. Uma situação potencialmente perigosa ocorre quando a perda de circulação se dá numa formação profunda, pois as mais rasas poderão entrar em *kick.* A perda de circulação poderá ser:

- Natural em formações fraturadas, vugulares, cavernosas, com pressão anormalmente baixa ou depletadas;
- Induzida através da massa específica excessiva do fluido de perfuração, da pressão de circulação excessiva no espaço anular, do surgimento de pressão devido à descida da coluna de perfuração ou de revestimento e de outras causas que resultem no aumento de pressão no poço, ultrapassando a pressão de fratura de alguma das formações presentes.

### 2.1.4 Massa Específica do Fluido de Perfuração Insuficiente

Uma massa específica baixa de um fluido de perfuração, irá acarretar em menores pressões hidrostáticas e perdas de carga, que pode reduzir as pressões dentro do poço a valores inferiores que a pressão de poros da formação.

Essa causa de *kicks* está normalmente associada à perfuração em áreas com formações de pressão anormalmente alta. Em perfurações efetuadas nessas áreas, os indicadores e as técnicas de detecção e medição de pressões anormalmente altas devem ser empregados para se elevar adequadamente a massa específica do fluido de perfuração, de forma a se evitarem influxos. É importante também lembrar que a massa específica do fluido de perfuração pode ter o seu valor reduzido pelo descarte de baritina no sistema de remoção de sólidos (centrífugas e *mud cleaner*) e pela sedimentação da baritina no poço ou nos tanques de lama, nas diluições e no aumento de temperatura do fluido, especialmente em poços HP/HT (*High Pressure, High Temperature*). Assim, para minimizar essa causa de *kicks*, é necessário sempre comparar a massa específica do fluido de perfuração com a equivalente de pressão de poros da formação. Uma solução óbvia para se evitar o *kick* causado por peso de lama insuficiente seria elevar o valor dessa propriedade. Entretanto, esse aumento, sendo excessivo, pode resultar em fratura de formações frágeis, redução da taxa de penetração e aumento das chances de prisão por pressão diferencial.

### 2.1.5 Corte da Lama por Gás

A incorporação de fluidos da formação no fluido de perfuração é conhecida com o nome de corte de lama. O corte de lama por gás é o que causa mais problemas à segurança do poço, pois o gás se expande quando trazido à superfície, causando uma diminuição na massa específica da lama e um consequente decréscimo da pressão no poço, que pode ser suficiente para gerar um *kick*. Pequenas quantidades de gás, incorporadas ao fluido de perfuração, ao chegarem à superfície, são registradas pelos detectores de gás. Embora a massa específica do fluido de perfuração muitas vezes esteja bastante reduzida na superfície, a pressão hidrostática no poço não decresce significativamente, pois a maior

expansão do gás ocorre próximo à superfície. Assim, na maioria dos casos, o corte do fluido de perfuração por gás não provoca a ocorrência de um *kick.*

Entretanto, é importante que o gás já incorporado ao fluido de perfuração seja removido pelo uso de degaseificadores e que a causa da contaminação seja identificada e eliminada.

### 2.1.6 Cimentação Inadequada

Antes de alcançar sua resistência compressiva final, é formada uma estrutura autossustentável que faz com que a pressão hidrostática da pasta se reduza à pressão hidrostática da água de mistura, enquanto existe permeabilidade ao gás.

Além disso, a redução do volume da pasta através da perda de filtrado é outro fator que reduz a pressão hidrostática da pasta antes da pega, permitindo influxos de gás. Com isso, é possível ocorrer um *kick.*

Certas medidas podem ser tomadas com o intuito de evitar esse problema, como:

- Minimizar a altura da pasta;
- Manter o anular pressurizado;
- Usar sais para aumentar a densidade da água de mistura;
- Usar pastas com tempos de pega diferenciados;
- Aumentar a massa específica do fluido antes da cimentação;
- Usar múltiplos estágios de cimentação;
- Usar pastas com aditivos bloqueadores de gás;
- Usar *External Casing Packer* (ECP) na coluna de revestimento;

## 2.2 Indícios e Detecção de *Kicks*

O tempo gasto no controle e a magnitude da pressão gerada durante uma operação de controle de poço estão diretamente relacionados ao volume do influxo do fluido para o poço. Assim, esse volume deve ser o mínimo possível, principalmente em perfurações em águas profundas, nas quais existem altas taxas diárias de sonda e baixos gradientes de pressão de fratura. O volume de um *kick* é minimizado quando a sonda possui equipamentos de detecção precisos e a equipe está treinada para detectar prontamente o *kick* e fechar o poço o mais rapidamente possível. Fica evidenciada assim a importância da rápida detecção do *kick* para minimizar os riscos de *blowouts* com todas as suas possíveis consequências (perdas de vidas humanas, da sonda e de reservas, poluição e liberação de gases venenosos para a atmosfera).

### 2.2.1 Indicadores de Aumento da Pressão de Poros

Há sempre o risco da ocorrência de um *kick* quando se perfura em áreas onde são encontradas pressões anormalmente altas. Existem os indicadores diretos e indiretos de pressão anormal. Enquanto os indicadores indiretos são obtidos antecipadamente como uma possibilidade de pressão alta, os diretos são obtidos durante a perfuração do poço com mais precisão.

### 2.2.2 Indicadores Diretos de Pressão Anormal

Quando a pressão anormalmente alta é causada pelo fenômeno da subcompactação, existe sempre uma zona de transição onde a pressão de poros aumenta com a profundidade. Nestas zonas, certas propriedades das formações e do fluido de perfuração são alteradas dando indicativos de aumento da pressão de poros. A observação e análises dos indicadores obtidos na superfície são necessárias para que as ações preventivas sejam tomadas para evitar a ocorrência de um *kick*. As formações com pressão anormalmente

alta possuem um teor de água maior que as com pressão normal devido ao fenômeno da subcompactação. Os indicadores mais importantes observados durante a perfuração são:

- Tamanho e forma dos cascalhos;
- Aumento do torque;
- Aumento do arraste;
- Mudança na temperatura do fluido de perfuração;
- Teor de gás no fluido de perfuração;
- Mudança das propriedades do fluido de perfuração.

### 2.2.3 Indicadores Indiretos de Pressão Anormal

Comumente, dois métodos são usados para avaliar pressões anormais de forma indireta: Análises sísmicas e perfilagem.

- Análises sísmicas: Através das interpretações sísmicas é possível perceber as primeiras indicações de pressões anormais em uma região. Para tal, a medida do tamanho da estrutura, assim como a profundidade e espessura de uma camada de sal são utilizadas.

As pressões encontradas em espessas camadas de folhelho podem ser identificadas e medidas com certo grau de precisão, pois à medida que a pressão cresce a velocidade da onda sonora diminui, e as medidas sísmicas se baseiam na velocidade de ondas sonoras.

- Perfilagem: Em áreas onde há disponibilidade de informações de outros poços, os perfis apresentam uma das melhores fontes de informação. Com isso, mudanças nas pressões causam mudança bem definida nos perfis.

### 2.2.4 Detecção de um *Kick*

A detecção de um *kick* é feita através de sinais detectados na superfície. Esses sinais podem ser captados durante a perfuração, durante uma manobra ou em uma perda de

circulação. Quanto mais rápido um *kick* for detectado, tomando-se as providências necessárias, mais fácil será o seu controle. Isso ocorre, porque são minimizados diversos fatores importantes como o volume do *kick*, as pressões no *choke* e no tubo bengala e o tempo gasto nas operações de controle.

Além disso, quanto maior o tempo gasto para identificar um *kick* e/ou tomar as atitudes necessárias para contê-lo, aumenta a probabilidade de ocorrer desastres como a transformação do *kick* em um *blowout*, a liberação de gases venenosos na área, a poluição do meio ambiente em geral e incêndios na plataforma.

Os principais indícios de que um *kick* está ocorrendo ou na iminência de ocorrer serão comentados a seguir.

### 2.2.5 Aumento do volume de fluido nos tanques

O aumento de volume de fluidos nos tanques é um dos melhores indicadores de *kick*, pois por ser um sistema fechado de circulação, se não houver nenhuma adição de fluido a esse sistema, qualquer invasão de fluido proveniente da formação irá resultar em um aumento no nível de lama nos tanques. De uma forma bem simples, o tanque passa a receber um volume de fluido maior do que o que foi injetado inicialmente, logo, existe outra fonte de fluidos sem ser a própria bomba. Nesse caso, a formação.

### 2.2.6 Aumento da Vazão de Retorno

Com a manutenção da vazão de injeção constante, um aumento da vazão de retorno indica que um *kick* está acontecendo ou que o gás existente no poço está em processo de expansão.

### 2.2.7 Aumento da Taxa de Penetração

Considerado um indicador secundário de influxo, uma vez que as alterações na taxa de penetração podem ter diversas causas, como variações do peso sobre a broca, da rotação ou da vazão, ou mudanças das formações cortadas pela broca.

No caso de um *kick*, ocorre um aumento brusco da taxa de penetração. Isso ocorre porque a pressão da formação é maior que a pressão de fundo do poço, o que cria um diferencial negativo de pressão atuante na formação a qual está sendo perfurada.

### 2.2.8 Fluxo com as Bombas Desligadas

Considerado um indicador primário de *kick*. Ao se desligar as bombas, a pressão no fundo do poço decresce num valor correspondente às perdas de carga no anular. Com essa diminuição na pressão a entrada de fluidos da formação fica facilitada. O contínuo deslocamento de fluidos no anular por parte do fluido proveniente da formação irá se refletir no tanque. Logo, um poço que está fluindo com as bombas desligadas é um sinal de *kick*.

### 2.2.9 Corte do Fluido de Perfuração por Gás e/ou Óleo

Quando o fluido mais leve da formação é injetado no poço a massa específica do fluido de perfuração é afetada, isto é, a massa específica decresce. Diz-se então que houve um corte.

Ocorrendo um corte de gás causado pelo gás contido nos cascalhes gerados pode também indicar que um influxo é iminente caso as providências já comentadas não sejam tomadas. Sempre que houver um corte de água e uma consequente alteração na salinidade da lama indicam um *kick* de água, neste caso é um indicador primário. Verificando-se na superfície um corte do fluido de perfuração quer seja por gás, óleo ou água as ações positivas devem ser imediatamente tomadas.

### 2.2.10 Aumento da Velocidade da Bomba e Diminuição da Pressão de Bombeio

Considerado um indicador secundário, dado que outros problemas como um furo na coluna e queda de jatos da broca podem apresentar esse mesmo sinal.

Porém, no caso de um *kick*, inicialmente, a entrada do fluido invasor no poço pode causar floculação da lama e temporariamente um aumento da pressão de bombeio. Como a circulação é contínua este efeito logo deixa de ser significativo. O menos denso fluido da

formação torna a hidrostática do anular mais leve que a do interior da coluna, como se trata de um tubo em "U" isto resulta num desbalanceio, aliviando o esforço da bomba.

### 2.2.11 Poço Aceitando Volumes Impróprios de Fluido Durante as Manobras

Devido ao volume do fluido da formação que adentrou o poço e ocupa o espaço anular, ao se retirar a coluna, o poço aceita um volume de fluido menor que o volume de aço retirado, ou então, no momento da descida da coluna, o poço devolve para os tanques um volume maior de fluido que o volume de aço introduzido.

## 2.3 Métodos de Controle

O objetivo principal dos métodos usados para controlar um *kick* é a circulação do fluido invasor, restabelecendo o controle do poço, substituindo a lama de perfuração que estava no poço por uma de densidade mais adequada visando igualar e conter a pressão da formação e sem ultrapassar a pressão de fratura. Os métodos que serão discutidos nesse capítulo são:

- Método do Sondador;
- Método do Engenheiro;
- Método Volumétrico;
- *Bullheading.*

Após o *kick* ser detectado, é necessário que a equipe responsável pelo controle do poço seja bem treinada para poder tomar decisões rápidas e acertadas. É preciso que a equipe esteja atenta para registrar algumas informações que serão importantes na realização de qualquer procedimento de circulação de um *kick*, tais como:

- Pressão de fechamento do revestimento (SICP);
- Pressão de fechamento do *drillpipe* (SIDPP);
- Aumento do volume de lama nos tanques ($V_k$);

- Profundidade vertical do poço ($D_v$);
- Dados do poço e coluna (BHA – *Bottom Hole Assembly*).
- Planilhas de *kick*;

Através dessas informações, é possível determinar fatores como o tipo de *kick*, a pressão de poros da formação, a densidade da nova lama a ser injetada no poço e as pressões inicial e final de circulação.

### 2.3.1 O Modelo do Tubo – U

Todos os procedimentos de circulação clássicos são baseados no modelo de tubo em U ilustrado na figura 2.2. É importante compreender este modelo e premissa. Muitas vezes, o pessoal de campo tentar aplicar os procedimentos clássicos de controle de poço para uma situação não clássica. Se o modelo do tubo em U não descreve com precisão o sistema, os procedimentos de controle clássicos de pressão não podem ser invocados.

Como ilustrado na figura 3, o lado esquerdo do tubo em U representa a coluna de perfuração enquanto o lado direito do tubo em U representa o espaço anular. Portanto, esse modelo descreve um sistema em que a broca está no fundo do poço e é possível fazer a circulação a partir desse ponto. Caso não seja possível circular a partir do fundo do poço, os conceitos clássicos de controle de poço são insignificantes e não aplicáveis. Uma vez que este tipo de situação não é de interesse deste projeto, não será discutido aqui.

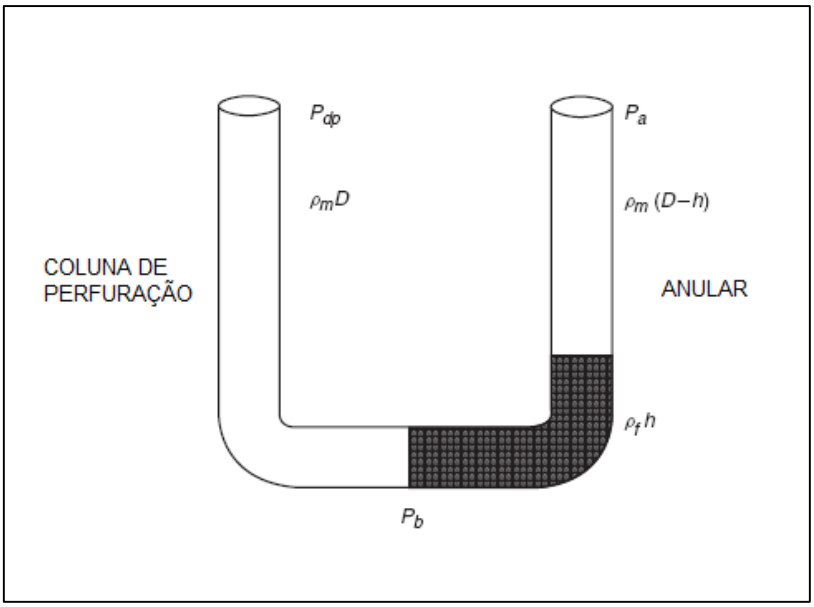


**Figura 3 - Modelo do Tubo em U**

Como também ilustrado na figura 3, um influxo de fluidos da formação entrou no anular (lado direito do tubo em U). O poço foi fechado, o que significa que o sistema tenha sido fechado. Sob estas condições, existe uma pressão estática na coluna de perfuração, que é denominada por $P_{dp}$, e uma pressão estática no anular, denominada por $P_a$. O fluido da formação, que possui uma massa específica $\rho_f$, entrou no anular e ocupou um volume definido pela área do anular e pela altura, h, do *kick*. É importante citar que a altura do *kick* é calculada pela equação 2.3:

$$h_{kick} = \frac{V_k}{C_a} \qquad (2.3)$$

onde,

$V_k$ = volume do *kick* medido no tanque de lama, em m³;

$C_a$ = capacidade do anular, em m³/m.

Uma inspeção da figura 3 indica que o lado da coluna de perfuração do Modelo de tubo em U é mais simples para analisar uma vez que as pressões são apenas influenciadas pela lama, que possui uma densidade conhecida e pela pressão na coluna, a qual é facilmente medida. Sob condições estáticas, a pressão no fundo do poço é facilmente determinada utilizando a equação 2.4:

$$P_b = \rho_m D + P_{dp} \quad (2.4)$$

onde:

$P_b$ = Pressão no fundo do poço, em psi;

$\rho_m$ = Gradiente de pressão da lama, em psi/ft;

$D$ = Profundidade, em ft;

$P_{dp}$ = Pressão de fechamento da coluna de perfuração, em psi.

A equação 2.4 descreve a pressão de fechamento no fundo do poço em termos da hidrostática total no lado da coluna de perfuração do Modelo de tubo em U. Essa pressão também pode ser descrita em termos da hidrostática total no lado do anular, tal como ilustrado pela equação 2.5:

$$P_b = \rho_f h_{kick} + \rho_m (D - h_{kick}) + P_a \quad (2.5)$$

onde:

$P_b$ = Pressão no fundo do poço, em psi;

$\rho_m$ = Gradiente de pressão da lama, em psi/ft;

$D$ = Profundidade, em ft;

$P_a$ = Pressão de fechamento do anular, em psi;

$\rho_f$ = Gradiente de pressão do *kick*, em psi/ft;

$h_{kick}$ = Altura do *kick*, em ft.

Independentemente da terminologia utilizada, os procedimentos clássicos para controle de poço devem manter pressão de fechamento no fundo do poço, $P_b$, constante para evitar influxo adicional de fluidos da formação enquanto estiver ocorrendo o deslocamento do influxo inicial para a superfície. Obviamente, a equação para o lado da coluna de

perfuração (equação 2.4) é a mais simples e todas as variáveis são conhecidas e, portanto, esse lado é usado para controlar a pressão no fundo do poço, $P_b$.

### 2.3.2 Método do Sondador

Este método consta de duas etapas ou circulações. A primeira circulação objetiva expulsar o fluido invasor utilizando apenas o fluido original. Com o poço já limpo do fluido invasor, inicia-se a segunda circulação que é o preenchimento do poço com o novo fluido de perfuração. Os procedimentos básicos consistem em:

- Manter a pressão constante no manômetro do *choke* enquanto a bomba é levada para a velocidade reduzida de circulação. Quando essa velocidade é atingida, a leitura no tubo bengala deverá ser ICP (Pressão de Circulação Inicial). Circular lama original na vazão reduzida de circulação, mantendo-se a ICP no tubo bengala, observando-se sempre as máximas pressões dinâmicas permissíveis. A ICP pode ser calculada pela equação 2.6:

$$ICP = P_{ks} + SIDPP \tag{2.6}$$

onde:

*ICP* = Pressão de Circulação Inicial, em psi;

$P_{ks}$ = Pressão da bomba na velocidade reduzida de circulação, em psi;

*SIDPP* = Pressão de fechamento da coluna de produção, em psi.

- Após circular, nessa situação, um volume equivalente ao do espaço anular, parar a bomba e fechar o *choke*. As pressões no tubo bengala e no *choke* deverão ser iguais a SIDPP;

- Se a pressão de fechamento da coluna de perfuração (SIDPP) for igual a pressão de fechamento do anular (SICP), pode-se determinar a densidade da nova lama que será circulada no poço e servirá para matar o poço. Esse cálculo é feito através da equação 2.7:

$$\rho_{km} = \frac{\rho_m D + P_{dp}}{0{,}052D} \quad (2.7)$$

onde:

$\rho_{km}$ = Massa específica da nova lama usada no poço, em ppg;

$\rho_m$.= Gradiente da lama original no poço, em psi/ft;

$P_{dp}$ = Pressão de fechamento da coluna de perfuração, em psi;

*D* =Profundidade, em ft.

- Bombear lama de matar pelo interior da coluna, mantendo a pressão no *choke* constante e iguala SIDPP até a lama nova atingir a broca. No início do bombeio, a pressão no tubo bengala deverá ser PIC. Essa pressão cairá constantemente até a lama nova chegar à broca quando seu valor será FCP (Pressão Final de circulação). O valor da FCP pode ser calculado pela equação 2.8:

$$FCP = P_{ks} \frac{\rho_{km}}{\rho_m} \quad (2.8)$$

onde:

FCP = Pressão Final de Circulação, em psi;

$P_{ks}$ = Pressão da bomba na velocidade reduzida de circulação, em psi;

$\rho_{km}$ = Massa específicada da nova lama usada no poço, em ppg;

$\rho_m$.= Massa específica da lama original no poço, em ppg;

- Manter a pressão no tubo bengala igual à FCP até a lama de matar chegar à superfície;
- Parar a bomba e fechar o *choke*. Observar as pressões no tubo bengala e no *choke*, que deverão ser nulas;
- Abrir o poço e observar se há fluxo;

- Se ainda existir fluxo no poço, deve-se repetir o procedimento;
- Se não for observado fluxo, aumenta-se o peso da lama para incluir uma margem de segurança desejada e começa-se a circular o poço até que a nova lama esteja completamente distribuída pelo sistema.

É possível visualizar através das figuras 4 e 5 o comportamento da pressão no revestimento e na coluna de perfuração durante a primeira circulação e durante a segunda circulação, respectivamente.

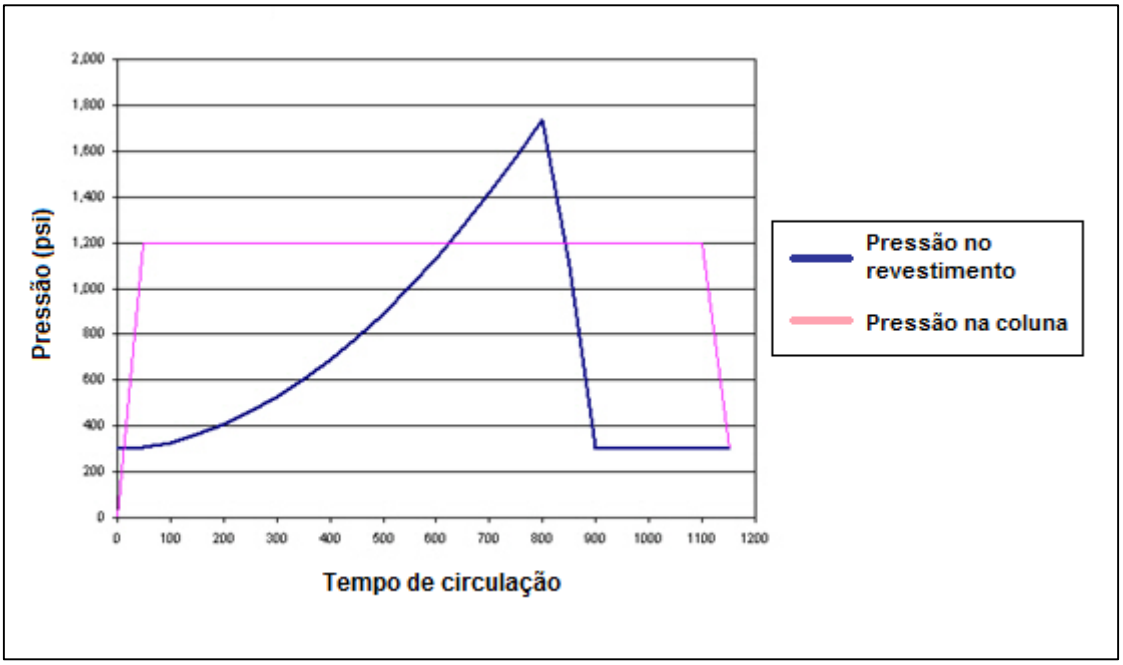


**Figura 4 - Perfil de pressões – 1ª circulação do Método do Sondador. Fonte: http://www.drillingformulas.com/how-are-pressure-and-pit-volume-doing-during-the-first-circulation-of-the-drillers-method/**

Nessa figura é possível perceber em um primeiro momento um aumento muito rápido da pressão na coluna e é nesse momento que se estabelece a ICP. Averiguamos em seguida um aumento continuo da pressão no revestimento com uma brusca queda no instante em que o gás chega à superfície. Por fim, depois de um período constante, a pressão na coluna sofre uma rápida queda devido à desaceleração da bomba.

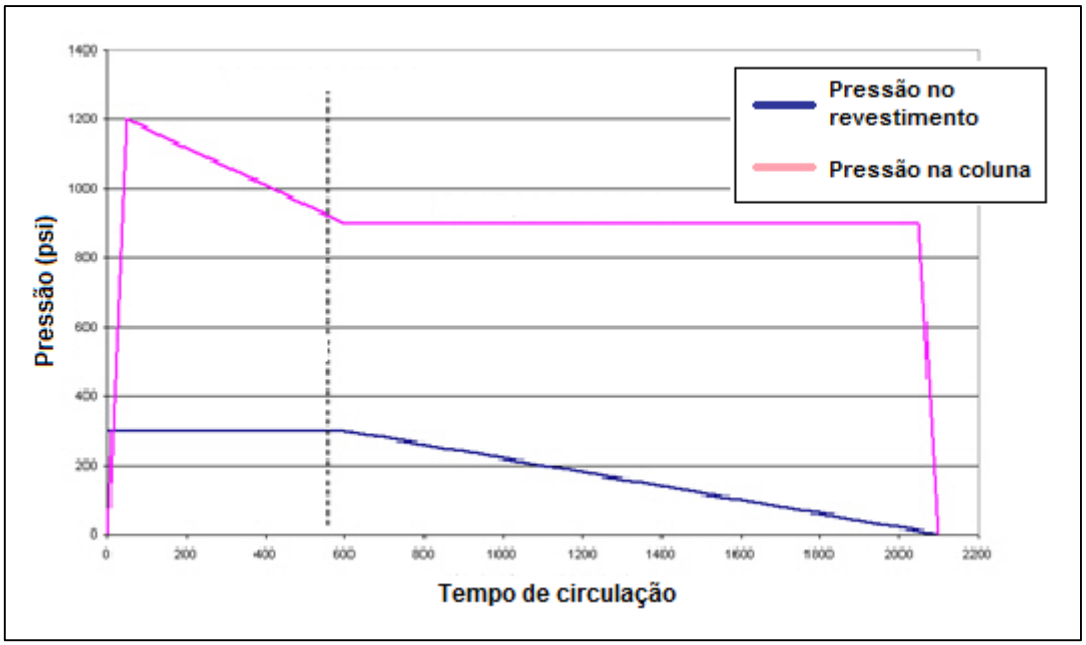


**Figura 5 - Perfil de pressões – 2ª circulação do Método do Sondador. Fonte: http://www.drillingformulas.com/how-is-pressure-doing-for-the-second-circulation-of-drillers-method/**

Na figura 5 temos, inicialmente, um aumento muito rápido da pressão na coluna, nesse instante, é estabelecida a vazão reduzida de circulação e a pressão passa a ser a ICP. Em seguida, temos uma queda continua da mesma até que seja alcançada a FCP, que ocorre quando a nova lama atinge a broca. Já a pressão no revestimento, permanece constante até ter inicio a abertura gradual do *choke*, a partir desse ponto ela começa a cair.

### 2.3.3 Método do Engenheiro

Nesse método, diferentemente do método supracitado, o poço é controlado com apenas uma circulação, ou seja, o influxo é removido do poço, utilizando-se o novo fluido de perfuração que já possui uma densidade maior, suficiente para matar o poço. Assim, a circulação começa após o fluido ter sido adensado. Na implementação do método, um gráfico ou uma planilha de pressão no tubo bengala em função do número de ciclos bombeados deve ser elaborado antes do início do bombeio. A necessidade da confecção do gráfico ou da planilha decorre do fato de que, quando o fluido novo está sendo deslocado no interior da coluna, a pressão no manômetro do choque não pode ser mantida constante, porque ao gás que se encontra no espaço anular deve ser permitida uma

expansão controlada. Assim, o *choke* deve ser manipulado de forma que a pressão no tubo bengala seja ICP, logo após o estabelecimento da velocidade reduzida de circulação, e caia linearmente até a FCP, quando o fluido novo atingir a broca.

A seguir, podemos observar o comportamento da pressão no revestimento e na coluna de produção através das figuras 6 e 7.

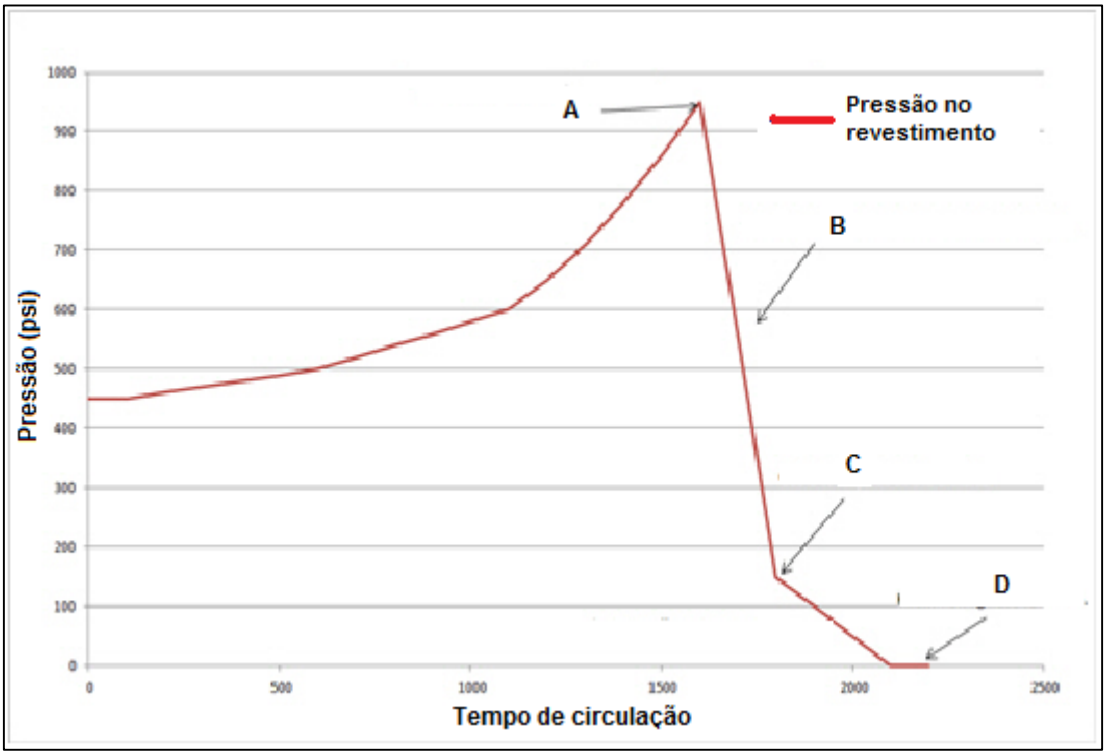


**Figura 6 - Pressão no revestimento durante a operação do Método do Engenheiro. Fonte: http://www.drillingformulas.com/pressure-profile-of-drillpipe-and-casing-pressure-while-killing-a-well-with-wait-and-weight-method/**

Na figura acima, podemos ressaltar alguns momentos importantes e que serão brevemente discutidos a seguir.

No momento A, temos a pressão máxima no revestimento, que ocorre quando o gás chega à superfície. Em seguida, podemos perceber uma queda na pressão, devido a migração desse mesmo gás para fora do poço (momento B). Quando a pressão alcança o ponto C, temos uma mudança na inclinação da curva da queda de pressão. Essa mudança ocorre no instante em que todo o gás é retirado do espaço anular. E por fim, a pressão passa a ter um comportamento constante igual a 0 quando a nova lama mais pesada chega à superfície (ponto D).

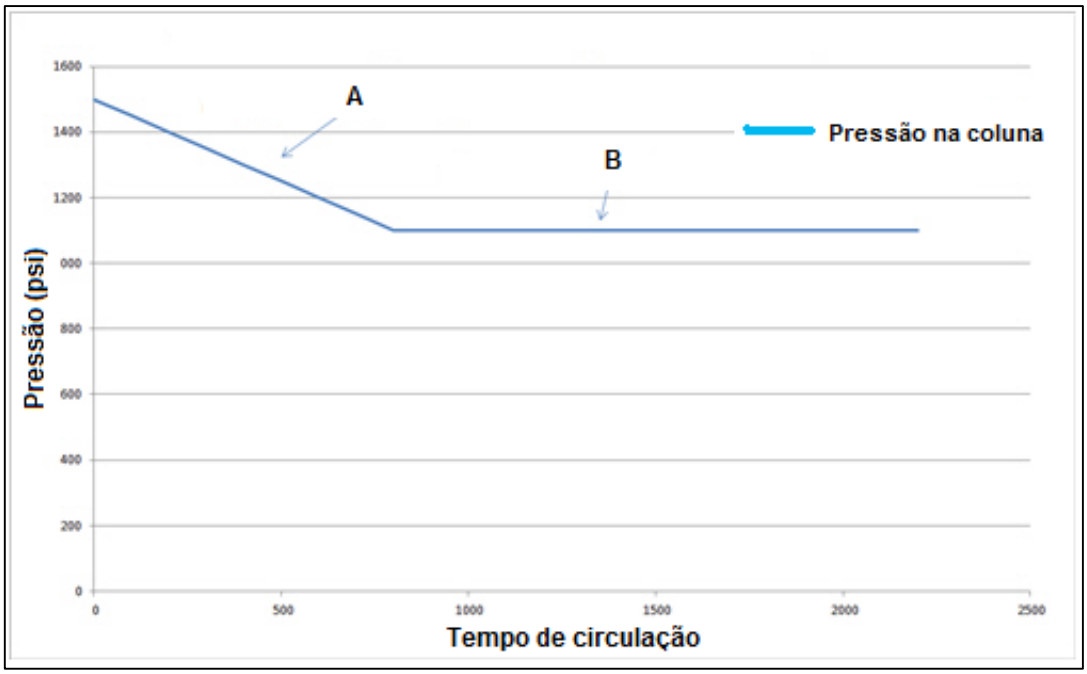


**Figura 7 - Perfil de pressões – Pressão na coluna de produção durante a operação do Método do Engenheiro. Fonte: http://www.drillingformulas.com/pressure-profile-of-drillpipe-and-casing-pressure-while-killing-a-well-with-wait-and-weight-method/**

Na figura acima, podemos observar o comportamento da pressão na coluna durante a operação. Em um primeiro instante A, é possível notar uma queda linear na pressão até que a nova lama de maior densidade chegue à cabeça do poço. E por fim, ela se mantém constante até que o poço esteja seguro e pronto para o recomeço da perfuração (ponto B).

Em comparação com o método do sondador, é mais rápido e necessita de menores pressões durante a circulação, o que é um ponto favorável, pois quanto menores as pressões, menor a probabilidade de ocasionar uma fratura na sapata. No entanto, apresenta maior dificuldade operacional.

### 2.3.4 Método Volumétrico

Os métodos volumétricos são utilizados nas situações em que o fluido de perfuração não pode ser circulado para deslocar o *kick* para fora do poço. Essas situações podem incluir:

- Jatos da broca entupidos;
- Problemas com as bombas ou equipamentos de superfície;

- Coluna fora do poço.

Durante a utilização de um método volumétrico, é preciso manter a pressão no fundo do poço aproximadamente constante, em um valor mínimo igual à pressão da formação que originou o *kick*, acrescentando-se uma margem de segurança arbitrária (normalmente 100 psi).

A primeira fase desse método consiste em permitir a migração do gás sob expansão controlada até que o mesmo atinja a superfície. Essa expansão controlada pode ser obtida através da drenagem de lama na superfície utilizando-se o *choke*. A aplicação do método consiste em seguir um procedimento em ciclos de migração e drenagem, nos quais a pressão no fundo do poço é mantida aproximadamente constante. Operacionalmente, o método é implementado da seguinte maneira:

- Após o fechamento do poço, permitir um crescimento de pressão de 100 psi (margem de segurança) no manômetro do *choke*;
- Permitir um novo acréscimo de 50 psi (margem operacional);
- Drenar, mantendo a pressão constante no *choke*, um volume de lama que origine uma pressão hidrostática de 50 psi. Esse volume ($V_m$) pode ser estimado pela equação 2.9:

$$V_m = 294 \frac{C_a}{\rho_m} \tag{2.9}$$

- Repetir o ciclo a partir do passo 2 até o gás atingir a superfície.

Na implementação desse procedimento, a pressão no fundo do poço permanecerá aproximadamente constante, variando entre 100 e 150 psi acima da pressão da formação, enquanto a pressão no choque será sempre crescente, atingindo o valor máximo quando o gás chegar à superfície. Nesse instante, a segunda fase do método, conhecido como *top kill*, pode ser implementada. Essa fase consiste em ciclos envolvendo períodos de injeção de fluido adensado pela linha de matar, segregação desse fluido adensado no poço e drenagem de gás pelo *choke*. O peso de fluido de perfuração a ser injetado pode ser estimado se o volume de gás no poço é conhecido. Assim, a massa específica do fluido de perfuração após adensamento será calculado pela equação 2.10:

$$\rho_{km} = \frac{P_{ck,max}\ C_a}{0.17 V_g} \tag{2.10}$$

onde,

$P_{ck,max}$ = máxima pressão lida no manômetro do *choke* (gás na superfície), em psi;

$V_g$ = volume de gás em barris, em bbl.

A segunda fase é implementada, utilizando-se o seguinte procedimento operacional:

- Injetar, pela linha de *kill*, um volume de lama nova ($V_m$) até que a pressão no *choke* ($P_{ck}$) aumente 100 psi acima da pressão da formação. Registrar esse volume e calcular o ganho de pressão hidrostática no fundo do poço ($\Delta P_h$) pela equação 2.11:

$$\Delta P_h = 0.17 \rho_{km} \frac{V_m}{C_a} \tag{2.11}$$

- Permitir a segregação da lama (3 minutos por cada barril injetado);
- Drenar o gás pelo *choke* até que a pressão no *choke* caia para $P_{ck}$ - $\Delta P_h$;
- Repetir o processo a partir do primeiro passo até que todo o gás tenha sido substituído pelo fluido adensado.

Durante a implementação dessa fase, a pressão no *choke* decresce ao longo do tempo, enquanto a pressão no fundo do poço é mantida aproximadamente constante.

### 2.3.5 Bullheading

A operação de *bullheading* consiste em deslocar ou injetar a mistura de fluido de perfuração e de influxo na formação exposta mais fraca no poço. Essa operação é empregada, em muitos casos, como o último recurso disponível, pois, em algumas situações, ela pode criar ou agravar um *underground blowout* ou causar um *blowout* em volta do revestimento. Essa operação pode ser usada quando:

- Há *kick* de $H_2S$;

- A circulação normal não é possível (jatos da broca entupidos, coluna fora do fundo do poço, partida ou fora do poço, falta de material para preparo do fluido de perfuração, defeito de equipamento, etc.);

- Há volume de gás elevado no poço (dificuldade para ser processado pelo separador e geração de pressões altas no *choke*);

- Houver combinação de *kick* e perda de circulação.

O sucesso da operação aumenta se forem observados os seguintes aspectos:

- As limitações de pressão da bomba, de ESCP e de revestimento devem ser sempre lembradas e observadas;

- O início da operação deve acontecer o mais cedo possível.

## 2.4 Revisão dos Modelos Propostos

Diversos modelos computacionais de simulação de *kick* foram propostos ao longo dos anos. O primeiro modelo foi proposto por LEBLANC e LEWIS (1968). Nesse modelo, as perdas de pressão pro fricção no anular são desconsideradas e esta região possua uma capacidade uniforme. Além disso, o gás é completamente insolúvel no fluido de perfuração e possui uma velocidade de escoamento igual a do fluido.

HOBEROCK e STANBERRY (1981) introduziram um modelo que simulava o comportamento dinâmico de um escoamento, incorporando as equações do momento para descrever a pressão numa linha rígida de transmissão vertical com uma seção de área constante. Eles consideraram um regime de escoamento de bolha e ajustaram as propriedades de uma região de escoamento bifásico como propriedades médias, sendo assim, possível de considerar o escoamento bifásico como um escoamento monofásico.

SANTOS (1982) apresentou um modelo matemático para circulação de *kick* em águas profundas considerando o escorregamento entre gás e fluido de perfuração, perdas de pressão por fricção na região bifásica e fração de vazios. O modelo também considera o regime de bolhas na região bifásica e geometria do poço constante. Santos utilizou o método de ORKISZEWSKI (1967) para computar o modelo reológico de Lei de Potência

para a lama. De acordo com os resultados, a densidade do gás, o gradiente geotérmico e o diâmetro mínimo das bolhas de gás causam um efeito mínimo na circulação do *kick*. Por outro lado, variáveis como a fração inicial de gás, geometria do poço, profundidade da lâmina d'água, diâmetro da linha de *choke* e parâmetros reológicos da lama exercem um efeito moderado na circulação. E também concluiu que o volume inicial de *kick* e a massa específica do fluido de perfuração exercem grande efeito durante a circulação do *kick*.

NICKENS (1987) apresenta um modelo baseado nas equações de conservação de massa do gás e da lama, do momento (mistura gás-lama), com uma correlação empírica que associa a velocidade do gás com a velocidade da mistura adicionado o fator de escorregamento entre as fases, além das equações de estado para a lama e para o gás. O modelo também considera os efeitos da geometria do poço, coluna de perfuração, broca, bomba de lama e do acoplamento entre o poço aberto e o reservatório.

PODIO e YANG (1986) propuseram um simulador de controle de poço para com características similares de modelos anteriores (NICKENS, 1987), com método diferente de solução das equações diferenciais. Ao invés de uma malha fixa, utilizou-se a técnica de fronteira móvel.

NEGRÃO e MAIDLA (1989) modelaram um *kick* de gás em águas profundas aplicando correlações bifásicas para o fluxo vertical da lama de perfuração e da mistura de gás. Este modelo pode prever a pressão ao longo da linha de *choke* durante o controle de *kick*. Utilizou-se a correlação de BEGGS e BRILL (1973) para computar a perda de carga por fricção na região bifásica.

OHARA (1996) desenvolveu um modelo matemático para controle de *kick* em poços localizados em lâminas d'água profundas com uma formulação similar a de estudos realizados por NICKENS (1987). O programa foi dividido em submodelos: anular do poço, reservatório de gás, linha de *choke* e escoamento em região bifásica. O programa também considerou o regime de escoamento de bolha de um gás natural em um fluido de perfuração de base água. Além disso, são incorporadas as equações de velocidade escorregamento do gás, assim como as perdas de carga na região monofásica e bifásica de escoamento. Por fim, o desenvolvimento do escoamento do gás dentro do anular foi baseado em dados experimentais do poço de teste da *Louisiana State University*.

NUNES (2002) apresentou um modelo matemático para prever o comportamento das pressões dentro da região anular durante a circulação de um *kick* de gás. O modelo leva em consideração diferentes configurações da geometria do anular com uma coluna de perfuração concêntrica, assim como as perdas de carga por fricção na região monofásica e bifásica e o escorregamento entre o gás e o fluido de perfuração. Foi utilizado o método do sondador para efetuar a circulação do *kick*, mantendo a pressão de fundo constante durante a operação e considerando a expansão do volume do gás. Por fim, foi realizada uma análise dos efeitos sobre a pressão na superfície de alguns fatores, como o ganho no tanque de lama, profundidade da lâmina d'água, densidade do fluido de perfuração e velocidade de bombeio.

# 3. DESENVOLVIMENTO

Inicialmente desenvolvido em *MatLab*, o programa foi posteriormente reescrito e finalizado até o dado momento em *Python*, visando assim possuir uma linguagem mais abrangente e de acesso livre a todos.

## 3.1 Considerações

Foram definidas algumas considerações com o intuito de simplificar o código desenvolvido, assim como otimizar os esforços computacionais do programa. As principais considerações feitas foram:

- Gás ideal;
- Gradiente de temperatura constante em todo o poço;
- Escoamento do tipo *slug;*
- Fluido de perfuração é totalmente incompressível;
- Velocidade de propagação da onda de pressão no poço é desconsiderada;
- Poço completamente vertical;
- Fluido de perfuração a base água;
- *Kick* de gás;
- Método de circulação utilizado: Método do Sondador;
- Não há solubilização do gás no fluido de perfuração;
- A expansão do gás não gera aumento de velocidade no fluido acima da bolha.

## 3.2 Dados de Entrada

O código foi elaborado de forma a ler todos os dados que o usuário precisa fornecer de um arquivo texto separado. No arquivo texto, o usuário deverá fornecer todos os dados requisitados em unidades comumente utilizadas na indústria, enquanto que o programa ao receber esses dados irá fazer as conversões necessárias para que todos os cálculos tenham um sistema único e consistente de unidades, evitando assim, erros e valores inesperados devido a falta de coerência de unidades. O sistema utilizado para os cálculos dentro do programa é o Sistema Internacional de Unidades (SI).

Dentro do arquivo texto, o usuário deverá fornecer todos os dados requisitados dentro das unidades sugeridas nos comentários ao lado de cada variável. Os dados foram divididos em cinco grandes grupos que serão mostrados a seguir.

### 3.2.1 Dados do Poço

- Profundidade
- Profundidade da linha de *choke*
- Profundidade de assentamento da sapata
- Altura de cada trecho do poço
- Diâmetro externo e interno da coluna
- Diâmetro interno do revestimento
- Diâmetro do poço aberto
- Diâmetro da linha de *choke*

### 3.2.2 Dados do Fluido

- Vazão de injeção
- Densidade do fluido de perfuração
- Parâmetros reológicos do fluido (*k* e *n*)

### 3.2.3 Dados do *Kick*

- Volume do *kick*
- Profundidade de entrada do *kick*
- Massa específica inicial do gás
- Viscosidade do gás

### 3.2.4 Dados Operacionais

- Gravidade
- Temperatura da superfície
- Gradiente de pressão de poros
- Gradiente de pressão de fratura
- Constante dos gases
- Peso molecular do gás

### 3.2.5 Dados do Reservatório

- Permeabilidade
- Porosidade

- Espessura

## 3.3 Dados Iniciais

Após o programa ler todos os inputs fornecidos, ele começa a criar variáveis próprias para utilização ao longo de todo o código em cálculos mais complexos.

Para isso, foi escolhida a utilização de vetores para que ao variar somente a uma posição dentro do mesmo vetor, fosse possível acessar mais de uma parte dentro do poço, sem a necessidade de criar diversas variáveis, poupando assim, memória do programa.

Primeiramente, foram criados sete vetores para as propriedades geométricas do poço, os quais são: diâmetro interno; diâmetro externo; diâmetro anular; área interna; área anular; volume interno e volume anular. Cada variável supracitada é um vetor.

A lógica para o preenchimento de cada posição dos vetores é simples e será repetida em outras partes do código. Cada vetor possui três posições que são, respectivamente, referentes às condições geométricas da linha de *choke*, às condições geométricas do poço revestido e às condições geométricas do poço aberto. Por exemplo, o vetor correspondente ao diâmetro anular seria da forma:

diametro_interno = [posição 0, posição 1, posição 2]

onde,

posição 0 = diâmetro da linha de *choke*, que nesse caso é nulo, pois a linha de *choke* não possui uma região anular.

posição 1 = diâmetro interno do revestimento, que começa no final da linha de *choke* até a sapata.

posição 2 = diâmetro do poço aberto, que começa no fim da sapata e vai até o final do poço.

Para os vetores correspondentes aos volumes, o programa faz uma discretização de todo o poço, dividindo toda a profundidade por uma altura que é previamente escolhida pelo

usuário. Para os testes realizados, essa altura foi definida como 1 metro, logo para um poço com 4000 metros de profundidade existiriam 4000 trechos discretizados.

Em seguida, foram criadas outras variáveis que também assumiam a forma de vetor com a mesma lógica de utilização de posições anteriormente utilizada. Essas variáveis são parâmetros importantes do poço e serão demonstradas a seguir.

- Velocidade do fluido:

$$v = \frac{Q}{A} \tag{3.1}$$

onde,

$Q$ = vazão de injeção, em m³/s;

$A$ = área (interna ou anular), em m².

- Taxa de deformação:

$$\gamma = 8\frac{v}{d_i} \quad (região\ interna) \tag{3.2}$$

$$\gamma = 12\frac{v}{d_h} \quad (região\ anular) \tag{3.3}$$

onde,

$d_i$ = diâmetro interno da coluna, em m;

$d_h$ = diâmetro hidráulico da região anular, em m;

$v$ = velocidade do fluido, em m/s.

- Viscosidade do fluido:

$$\mu = k\gamma^{(n-1)} \tag{3.4}$$

onde,

$\gamma$ = taxa de deformação interna ou anular, dependendo da região de interesse.

- Número de Reynolds:

$$R_e = \frac{\rho v d}{\mu} \tag{3.5}$$

onde

$\rho$ = massa específica do fluido, em kg/m³;

$d$ = diâmetro interno ou hidráulico, em m;

$\mu$ = viscosidade do fluido, em Pa.s;

$v$ = velocidade do fluido, em m/s.

- Fator de atrito

$$\frac{16}{R_e} \quad (região\ interna) \tag{3.6}$$

$$\frac{24}{R_e} \quad (região\ anular) \tag{3.7}$$

Por fim, de posse de todas as variáveis criadas acima, elaborou-se vetores que calculavam a pressão hidrostática e as perdas de carga a cada trecho. Esses vetores computavam desde a superfície até a profundidade final do poço cada trecho e guardavam o valor atual, adicionando os valores passados, de forma que, é possível acessar seus valores para cada profundidade diferente, caso seja preciso.

Para calcular a pressão hidrostática utilizamos a fórmula:

$$P_h = \rho g h \tag{3.8}$$

De forma que h é a altura de trecho selecionado pelo usuário. Logo, é possível calcular a hidrostática para cada trecho discretizado e depois calcular o total.

## 3.4 Cálculo da Perda de Carga

Com ideia semelhante, foram calculadas as perdas de carga para a parte interna da coluna, para a linha de *choke*, para a região anular do poço revestido e para a região anular do poço aberto. Além disso, os cálculos foram feitos levando em conta a presença do gás em determinada parte da região anular, considerando tal região como um escoamento bifásico.

### 3.4.1 Cálculo da Perda de Carga na Região Monofásica

As perdas de carga foram calculadas pela fórmula:

$$\Delta P = \frac{2 f \rho h v^2}{d} \tag{3.9}$$

onde,

$f$ = fator de atrito (interno ou anular);

$\rho$ = massa especifica do fluido, em kg/m³;

$h$ = altura do trecho, em m;

$v$ = velocidade do fluido (região interna, da linha de choke ou anular), em m/s;

$d$ = diâmetro (interno da coluna, interno da linha de choke ou hidráulico), em m.

### 3.4.2 Cálculo da Perda de Carga na Região Bifásica

Para a região bifásica do anular, foi adotada uma formulação diferente para calcular as perdas de carga. Para tal, era necessário o cálculo da fração de vazio na região bifásica (equação 3.10) e do *holdup* do líquido (equação 3.11).

$$\lambda = \frac{U_{gs}}{C_o(U_{gs}+U_{gl})+U_{sl}} \tag{3.10}$$

onde,

$U_{gs}$ = Velocidade superficial do gás, em m/s;

$U_{gl}$ = Velocidade superficial do líquido, em m/s;

$U_{sl}$ = Velocidade de escorregamento, em m/s.

$C_o$ = 1.2

$$H = 1 - \lambda \tag{3.11}$$

De posse desses valores, foi possível calcular a densidade (equação 3.12) e a velocidade de escoamento (equação 3.13) dessa região.

$$\rho_m = \left(\rho_f H\right) + (\rho_g \lambda) \tag{3.12}$$

onde,

$\rho_f$ = massa específica do fluido, em kg/m³;

$\rho_g$ = massa específica do gás, em kg/m³;

H = *holdup* do líquido

$\lambda$ = fração de vazio

$$v_m = (U_{sl} H) + (U_{sg} \lambda) \tag{3.13}$$

onde,

$U_{sl}$ = velocidade superficial do líquido, em m/s;

$U_{sg}$ = velocidade superficial do gás, em m/s;

H = *holdup* do líquido

$\lambda$ = fração de vazio

Com isso, foi calculado, através da equação 3.14, o fator de atrito para a região bifásica.

$$f_{bi} = fe^{s} \tag{3.14}$$

Onde, o fator de *skin* também pode ser calculado pela equação 3.15.

$$s = \frac{\ln(\frac{\lambda}{H^2})}{-0.0523+\left(3.182\ln\left(\frac{\lambda}{H^2}\right)\right)-\left(0.8725\left(\ln\left(\frac{\lambda}{H^2}\right)\right)^2\right)+\left(0.01853\left(\ln\left(\frac{\lambda}{H^2}\right)\right)^4\right)} \tag{3.15}$$

onde,

H = *holdup* do líquido

$\lambda$ = fração de vazio

Por fim, é possível calcular a perda de carga total dentro da região bifásica do anular pela equação 3.16.

$$\Delta P_{bi} = \frac{f_{bi}\rho_m v_m^2 h_{bi}}{2gd_{bi}} \tag{3.16}$$

onde,

$f_{bi}$ = fator de atrito bifásico

$\rho_m$ = massa específica da mistura, em kg/m³;

$v_m$ = velocidade da mistura, em m/s;

$h_{bi}$ = comprimento do trecho da região bifásica, em m;

g = gravidade, em m/s²;

$d_{bi}$ = diâmetro da região bifásica, em m

## 3.5 Modelagem do Reservatório

Com intuito de calcular a vazão de entrada do *kick* no poço, foi criado um simples modelo de reservatório. Para tal, foi considerado um reservatório com fluxo radial em um regime transiente. A equação 3.17 explicita o modelo utilizado.

$$q_w = \left(P_i^2 - {P_w}^2\right) \frac{2kh_e\pi}{\ln\left(\frac{4k\Delta t}{\gamma \mu_g \varphi c_g^2 r_w}\right) * \mu * P_w} \quad (3.17)$$

Onde

$q_w$ = vazão de entrada do kick, em m³/s;

k = permeabilidade, em m²;

$h_e$ = espessura, em m;

$\Delta t$ = passo de tempo, em s

$\gamma$ = constante de Euler

$\mu_g$= viscosidade do gás, em Pa.s;

$\varphi$ = porosidade;

$c_g$ = compressibilidade do gás, em $Pa^{-1}$;

$r_w$ = raio do poço, em m;

$P_w$ = Pressão no fundo do poço, em Pa;

$P_i$ = Pressão inicial, em Pa.

## 3.6 Cálculo da Velocidade de Ascensão da Bolha

Segundo DIMITRESCU (1943) e DAVIES e TAYLOR (1950), para um escoamento do tipo bolha, podemos considerar que a velocidade de escorregamento entre a fase gasosa e a fase líquida se da por:

$$v_{sl} = 0.35\sqrt{g.d_t} \quad (3.18)$$

onde,

g = gravidade, em m/s²;

$d_t$= diâmetro do tubo, em m.

NICKLIN *et al*. (1962) propuseram uma equação para a velocidade de ascensão da bolha considerando também a parcela devido a velocidade de escoamento do fluido:

$$v_g = C_o v_m + v_{sl} \tag{3.19}$$

onde,

$v_m$ = velocidade da mistura, em m/s;

$v_{sl}$ = velocidade de escorregamento, em m/s.

$C_o$ = parâmetro de distribuição, que para um padrão vertical ascendente em golfadas é igual a 1.2

## 3.7 Cálculo da Nova Lama para a Segunda Circulação

Após o *kick* ter sido completamente circulado, inicia-se o processo de substituição da lama antiga por uma nova, mais densa, para assim, ser possível prosseguir com a perfuração sem a ocorrência de novos *kicks*.

Como visto na literatura existente, a densidade da nova lama a ser utilizada pode ser calculada por:

$$\rho_{fn} = \frac{(\rho_f Z) + SIDPP}{0.052Z} \tag{3.20}$$

onde,

$\rho_f$ = massa específica da lama antiga, em kg/m³;

Z = profundidade, em m;

SIDPP = pressão de fechamento da coluna (*Shut-In DrillPipe Pressure*), em Pa.

## 3.8 Metodologia do Programa

O código foi dividido em duas etapas distintas. A primeira, correspondente a primeira circulação do Método do Sondador, e a segunda etapa, correspondente a segunda circulação do mesmo.

Na primeira etapa, o objetivo era de circular todo o influxo de gás presente no espaço anular do poço para fora, utilizando a lama presente no poço. Para tal, foi elaborada uma lógica em que a cada passo de tempo o gás migrava de acordo com a velocidade previamente calculada, e era possível, a partir da posição de entrada do *kick* e da posição da base da bolha, calcular a posição do topo da mesma.

Em seguida, os cálculos das pressões e perdas de carga eram realizados e calculava-se a pressão no fundo do poço através do espaço anular. Nesse momento, o programa entra em um *loop* iterativo, onde ele compara a pressão no fundo do poço que acabou de ser calculada com a pressão que deveríamos ter para evitar um novo *kick*. Caso essas pressões não sejam iguais, o programa interfere na pressão atual do *choke*, aumentando ou diminuindo a mesma de acordo com os resultados apresentados. Com isso, com uma pressão no *choke* diferente, é realizado um novo cálculo para o volume total do gás, uma vez que a capacidade de expansão do mesmo se altera com uma pressão maior ou menor no *choke*. Assim, como a posição da base da bolha é tida como constante para um mesmo passo de tempo, calcula-se a nova posição do topo e os cálculos de pressão e perda de carga são refeitos para gerar uma nova pressão de fundo.

Ao fim desse *loop*, quando a pressão de fundo calculada possui um valor desejado, os valores da pressão no *choke* do volume do *kick* são mantidos e transportados para o passo de tempo seguinte, para servir como condição inicial para todos os cálculos desse novo instante. Essa lógica se repete até que todo o gás seja retirado do poço.

Na segunda etapa, antes de se iniciar a segunda circulação, calcula-se a densidade da nova lama a ser utilizada, como demonstrado no item acima. E são calculadas todas as propriedades referentes a essa nova lama, para que os futuros cálculos de perda de carga possam ser realizados corretamente.

Nessa segunda circulação, até o momento em que a nova lama chegue à broca, a cada passo de tempo, calcula-se novamente a pressão no fundo do poço através do espaço

anular e verifica-se se a mesma se mantém constante. Como a novo fluido está totalmente dentro da coluna, e o cálculo da pressão de fundo é feito através da região anular, não há variação da mesma, não sendo necessária a criação de um *loop* iterativo.

Por fim, à medida que a novo fluido começa a migrar pela região anular, se faz necessário um pequeno cálculo iterativo para manter a pressão de fundo constante variando conforme necessidade a pressão no *choke*. Além disso, nesse momento, a pressão na sapata começou a ser monitorada de forma mais cuidadosa para se evitar o fraturamento das formações. Logo, foi criada uma janela de pressões máximas e mínimas que poderia haver no *choke* para que não ocorresse um novo *kick*, ao mesmo tempo em que não acontecesse a fratura na sapata. Novamente, essa lógica foi utilizada até que todo o poço estivesse completado com a nova lama de perfuração.

# 4. RESULTADOS

Nesse capítulo serão apresentados e discutidos os principais resultados obtidos pelo programa desenvolvido. Além do caso de estudo principal, um outro cenário teórico também será apresentado, com o intuito de validar as hipóteses apresentadas na literatura com o comportamento do simulador.

## 4.1 Caso Teórico

Ao longo do desenvolvimento do código, uma situação teórica foi criada e testada para validar o simulador:

- Migração do *kick* com o poço completamente fechado;

### 4.1.1 Migração do *Kick* com o Poço Completamente Fechado

Esse cenário tinha como objetivo testar o comportamento do simulador em um caso mais simples, com menos variáveis a serem consideradas, para assim, poder validar os resultados obtidos com os esperados de acordo com a literatura.

Nesse caso hipotético, ao se manter o poço completamente fechado durante todo o processo, foi considerado que:

- O volume do gás é sempre constante;
- Não existem perdas de carga a serem calculadas;
- O processo de migração do gás ocorre somente em função da velocidade de escorregamento entre as fases;
- Não existe nenhuma tentativa de controle das pressões no poço.

Os gráficos abaixo mostram os resultados obtidos para o monitoramento das pressões (figura 8), para o deslocamento da bolha de gás ao longo do tempo (figura 9) e para o comprimento do *kick* e volume do mesmo (figura 10).

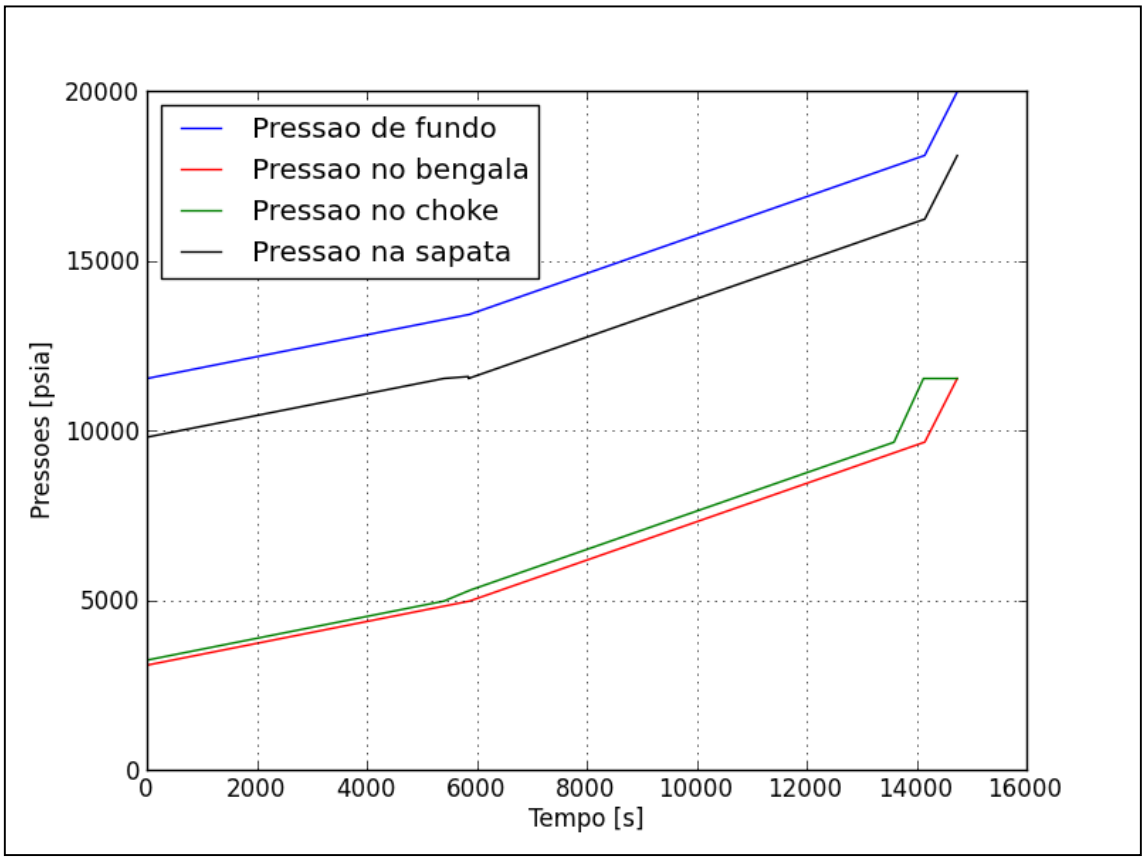


**Figura 8 - Comportamento das Pressões**

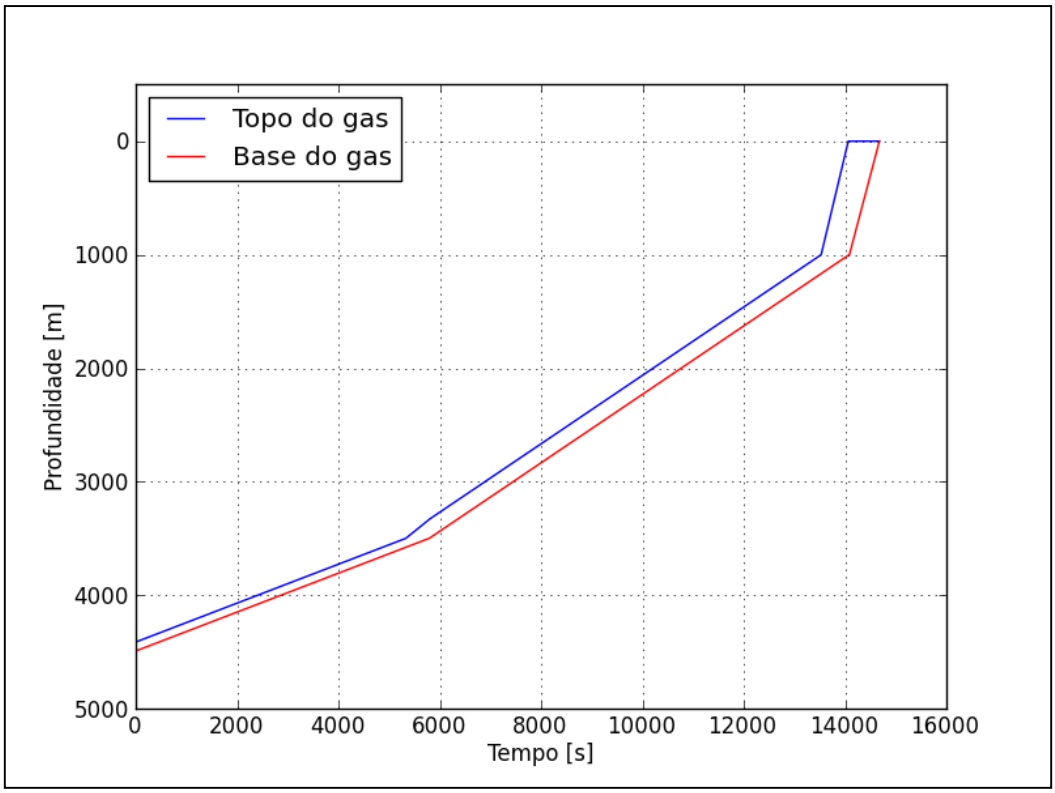


**Figura 9 - Movimentação do kick ao longo do poço**

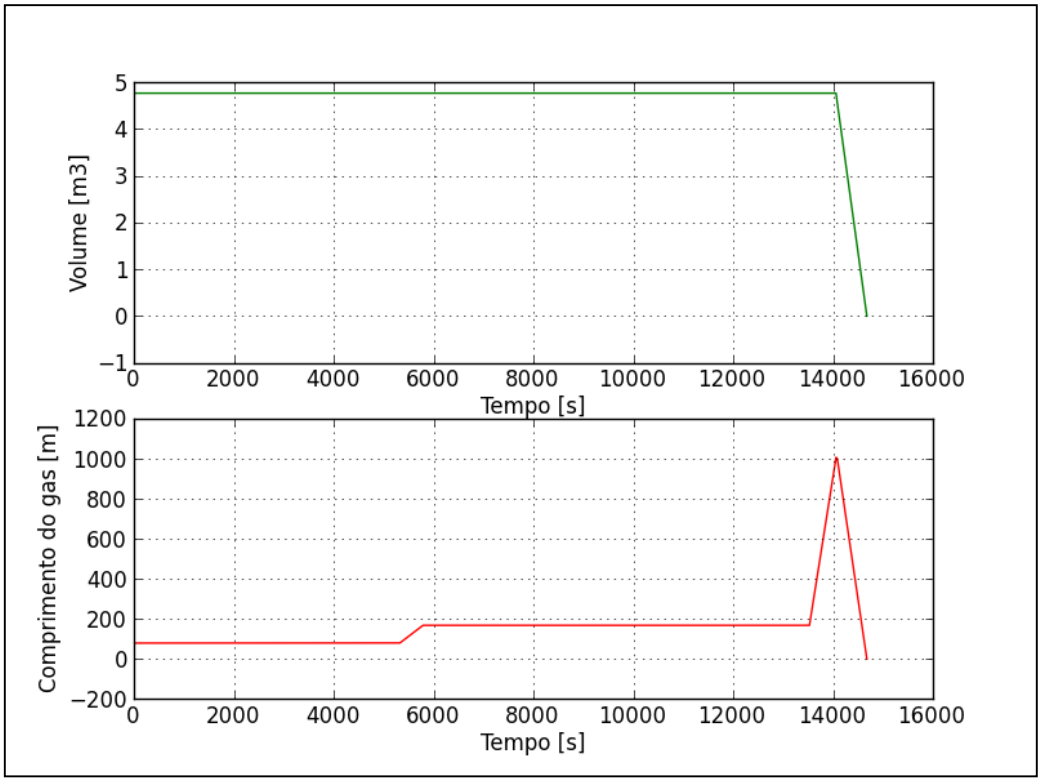


**Figura 10 - Volume do kick / Comprimento do kick**

Ao analisar os gráficos, podemos concluir que os resultados iniciais estão de acordo com o esperado, uma vez que por estar completamente fechado, o poço impede qualquer tentativa de expansão do gás, fazendo com que o mesmo transporte consigo sua pressão inicial à medida que vai migrando pelo poço, ocasionando assim, um aumento generalizado nas pressões em questão.

Também é interessante frisar que no final do gráfico da figura 8, é possível ver que as pressões no tubo bengala e no *choke* se igualam, mostrando assim que não existe mais nenhum outro fluido dentro do sistema.

E por fim, analisando o gráfico da figura 10, percebe-se que o volume do *kick* se mantém, como esperado, constante por todo o processo, somente diminuindo à medida que o gás começa a sair do poço. Com isso, concluímos que o aumento no comprimento do mesmo se dá em função da mudança na geometria do espaço anular do próprio poço.

## 4.2 Caso de Estudo

Nesse caso, o objetivo simular a circulação de todo o poço através do método do sondador. Como citados no capítulo 3, os dados utilizados como *inputs* foram:

**Tabela 1 - *Input* dos dados do poço**

| Dados do Poço | Valores |
|---|---|
| **Profundidade** | 4500 m |
| **Profundidade da linha de *choke*** | 1000 m |
| **Profundidade da sapata** | 3500 m |
| **Diâmetro externo da coluna** | 5 in |
| **Diâmetro interno da coluna** | 4 in |
| **Diâmetro interno do revestimento** | 9 in |

| | |
|---|---|
| **Diâmetro do poço aberto** | 12 in |
| **Diâmetro da linha de *choke*** | 3 in |

**Tabela 2 - *Input* dos dados do fluido**

| Dados Reológicos do Fluido | Valores |
|---|---|
| ***K*** | 1,8 $Pa.s^N$ |
| ***N*** | 0,6 |
| **Densidade do fluido** | 11 ppg |
| **Vazão de injeção** | 100 gpm |

**Tabela 3 - *Input* dos dados do kick**

| Dados do Kick | Valores |
|---|---|
| **Volume do kick** | 30 bbl |
| **Profundidade de entrada do kick** | 4500 m |
| **Densidade do gás** | 2 lb/Gal |
| **Viscosidade do gás** | 0,0000174 Pa.s |

**Tabela 4 - *Input* dos dados operacionais**

| Dados Operacionais | Valores |
|---|---|
| **Gravidade** | 9,8 $m^2/s$ |
| **Temperatura da superfície** | 298 k |

| | |
|---|---|
| **Gradiente de pressão de poros** | 15 lb/Gal |
| **Gradiente de pressão de fratura** | 17 lb/Gal |
| **Constante dos gases** | 8.134 |
| **Peso molecular do gás** | 16 (metano) |

**Tabela 5 - *Input* dos dados do reservatório**

| Dados do Reservatório | Valores |
|---|---|
| **Permeabilidade** | 50 md |
| **Porosidade** | 0,14 |
| **Espessura** | 20 m |

Com os dados das tabelas acima, o programa foi rodado e os resultados para a primeira circulação do Método do Sondador podem ser vistos a seguir nas figuras 11 (comportamento das pressões), 12 (movimentação da bolha de gás) e 13 (Volume de gás e comprimento da bolha).

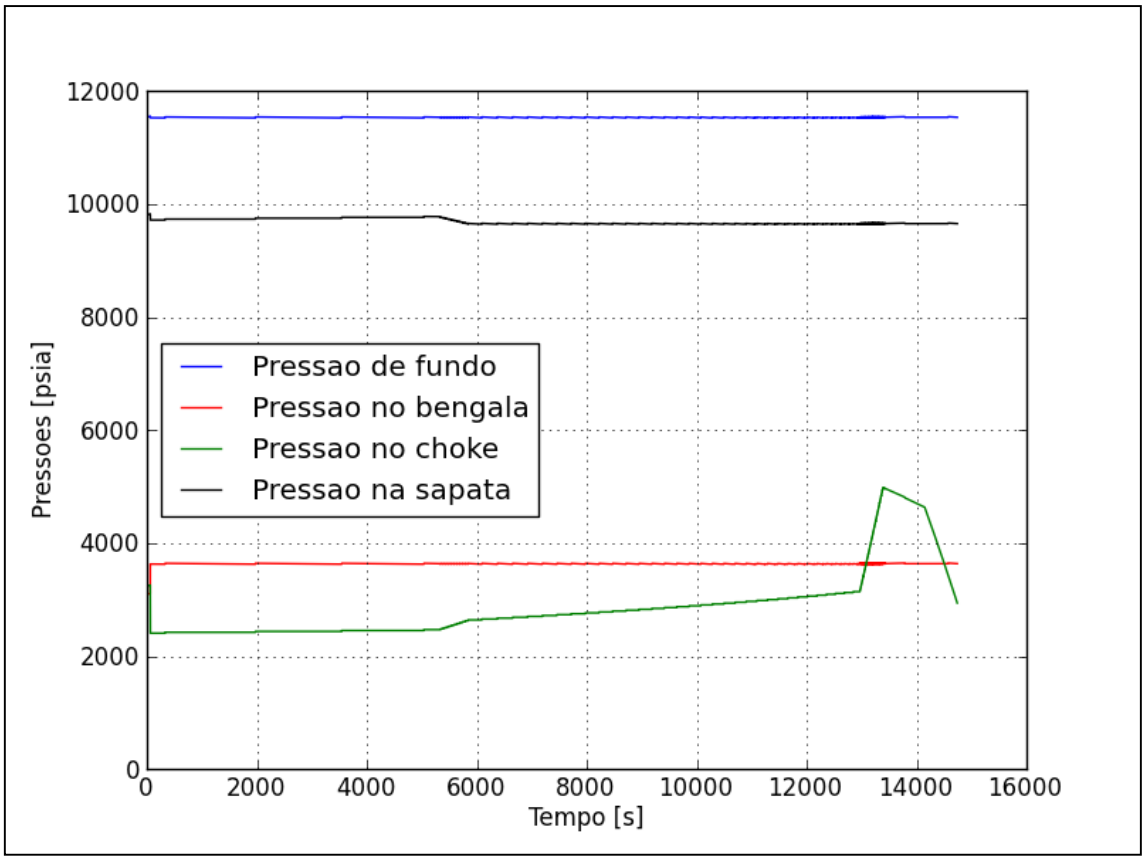


**Figura 11 - Comportamento das Pressões Durante a 1ª Circulação**

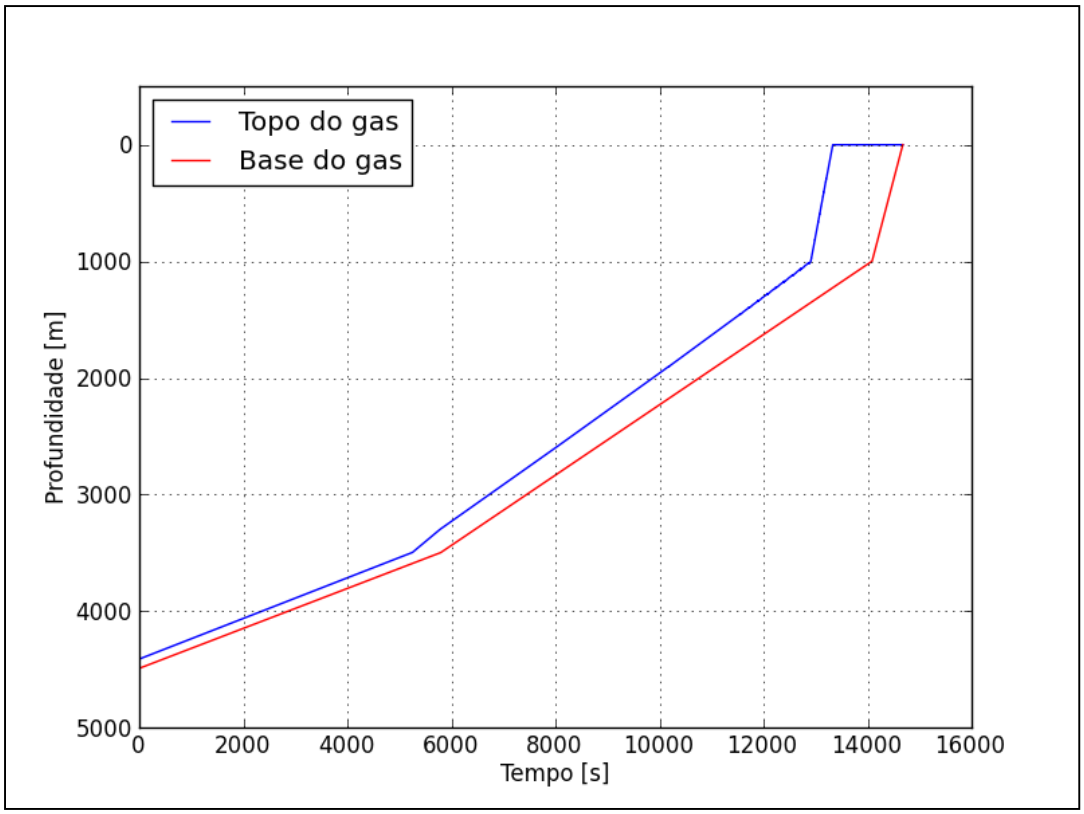


**Figura 112 - Movimentação do kick ao longo do poço**

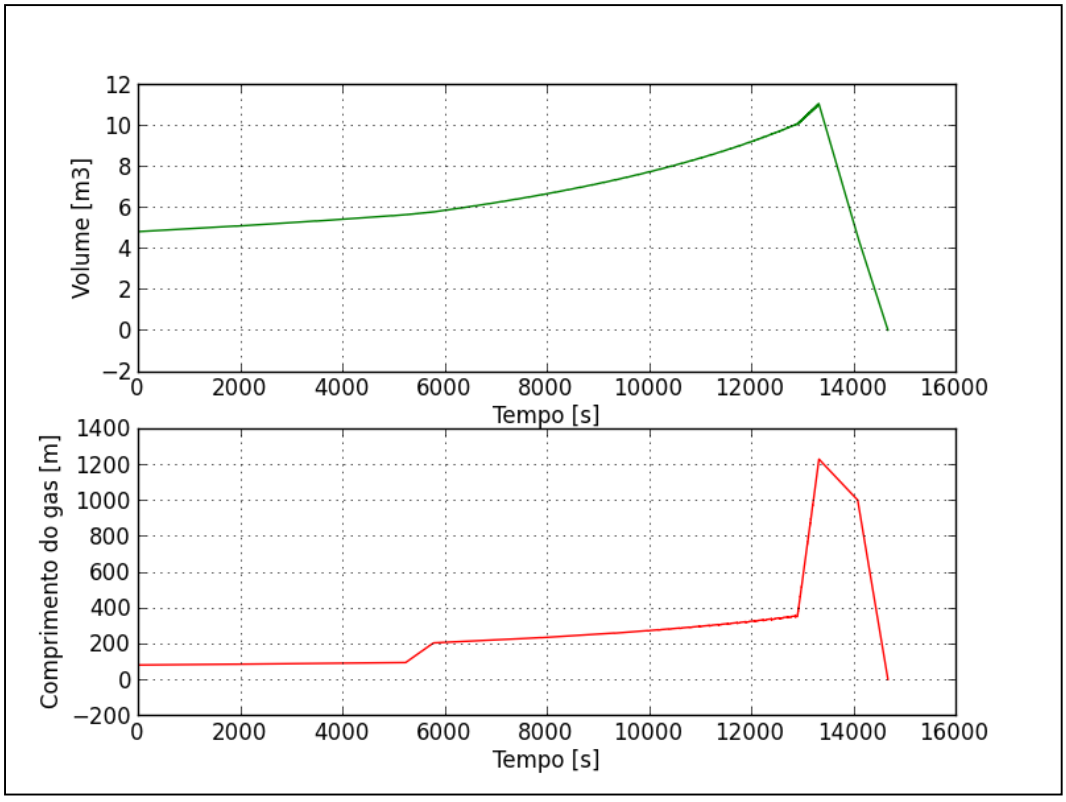


**Figura 123 - Volume do kick / Comprimento do kick**

Ao analisar o gráfico da figura 11, é possível perceber que o programa é bem sucedido em manter a pressão de fundo constante controlando a pressão no *choke*. É interessante ressaltar também que a mudança de inclinação na curva durante o momento de queda de pressão é devido ao gás começar a sair do poço não estando completamente na dentro da linha do *choke*. O que mostra mais uma vez o efeito do aumento do volume da bolha de gás.

Nas figuras 12 e 13, o efeito da expansão do volume do gás também pode ser notado, onde ao compara a figura 18 com a figura 10, é possível ver que o topo da bolha de gás chega a superfície em menos tempo, condizendo com a expansão volumétrica mostrada na figura 13.

Após todo o gás ser retirado do poço e a nova lama ser calculada, inicia-se a segunda circulação do Método do Sondador. Os resultados obtidos para tal podem ser vistos nas figuras 14 (comportamento das pressões) e 15 (janela de pressões permissíveis no *choke*).

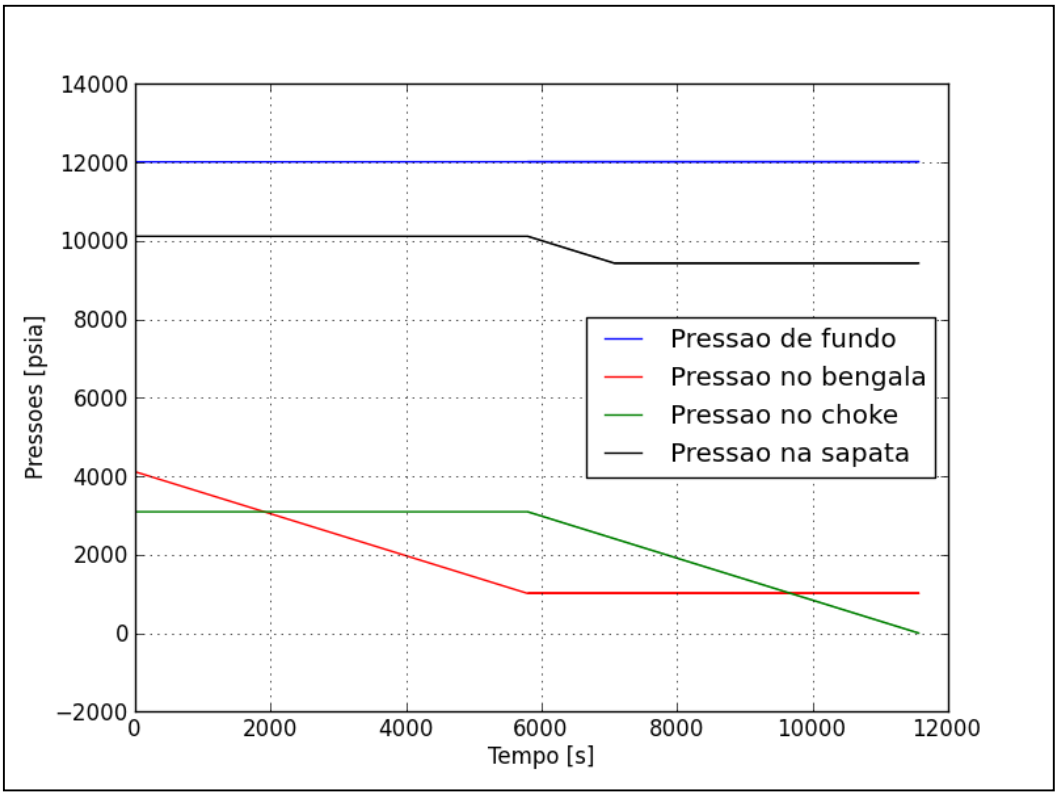


**Figura 14 - - Comportamento das Pressões Durante a 2ª Circulação**

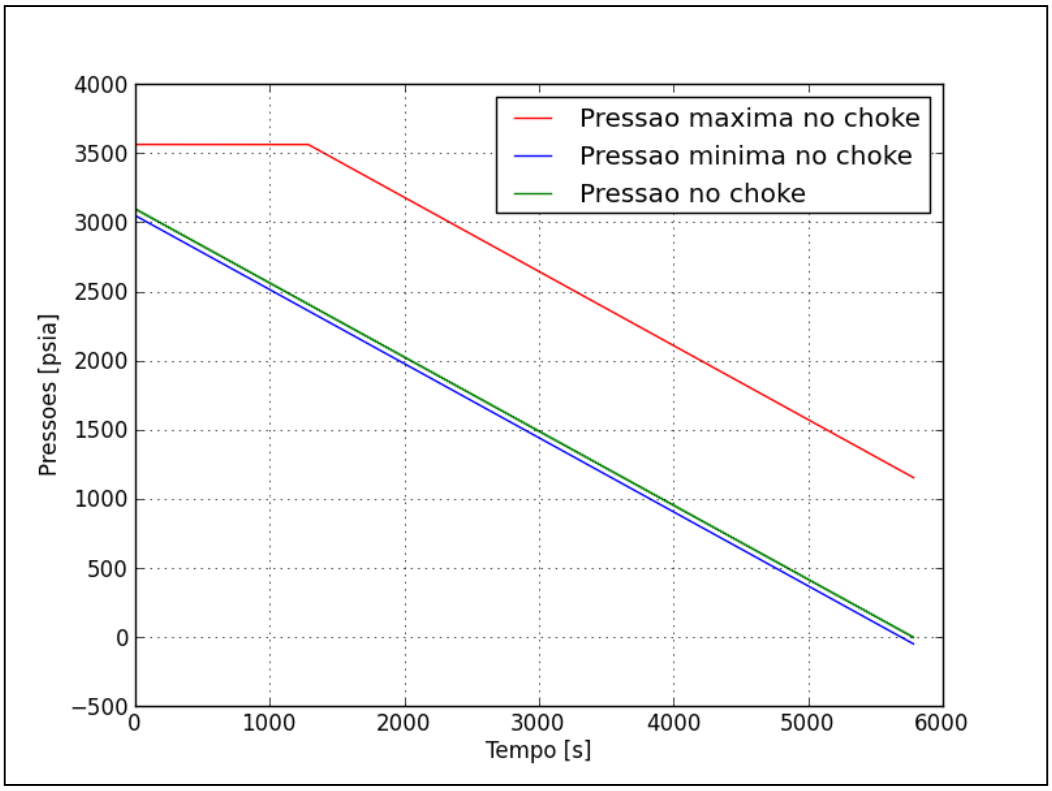


**Figura 15 - Janela de Pressões Permissíveis no Choke**

Pela figura 14 é possível perceber 2 etapas distintas dentro da segunda circulação. Inicialmente, enquanto a nova lama mais densa ainda não chegou à broca, ocorre uma queda constante na pressão no tubo bengala. Isso ocorre devido ao aumento da hidrostática dentro da coluna enquanto que a pressão no fundo do poço é mantida constante. Da mesma forma, podemos ver que a pressão no *choke* é mantida constante, uma vez que enquanto a nova lama estiver somente dentro da coluna, o espaço anular tem suas condições mantidas, não sendo afetado pela troca do fluido.

Em seguida, quando a nova lama começa a preencher o espaço anular, ocorre uma troca, a pressão no tubo bengala passa a ficar constante, uma vez que toda a coluna já está preenchida com a nova lama, e a pressão no *choke* começa a cair. Essa queda ocorre devido principalmente a necessidade de se manter a pressão de fundo constante e de evitar o fraturamento da sapata. Pois a medida que a nova lama vai sendo deslocada pelo espaço anular, as pressões no mesmo vão crescendo, fazendo com que seja necessária a redução da pressão que existia no *choke*.

Já no gráfico da figura 15, podemos ver a janela de pressões máximas e mínimas permissíveis no *choke* para se evitar tanto um novo *kick* quanto a fratura da sapata. Esse intervalo de pressões foi gerado somente a partir do momento em que o novo fluido sai da broca e começa a se deslocar pelo espaço anular. Pois, enquanto o fluido está dentro da coluna, a pressão mínima para se evitar um *kick* é sempre constante e igual a SIDPP.

Nota-se que a pressão no *choke* fica sempre dentro desta janela, na parte inferior, próxima a pressão mínima permitida. Isso ocorre porque o programa controla a pressão no *choke* sempre tentando manter a pressão de fundo constante em um valor muito próximo do calculado no início da segunda circulação.

Por fim, na figura 16 podemos ver o gráfico completo para todo o tempo de simulação, englobando tanto a 1ª circulação quanto a 2ª.

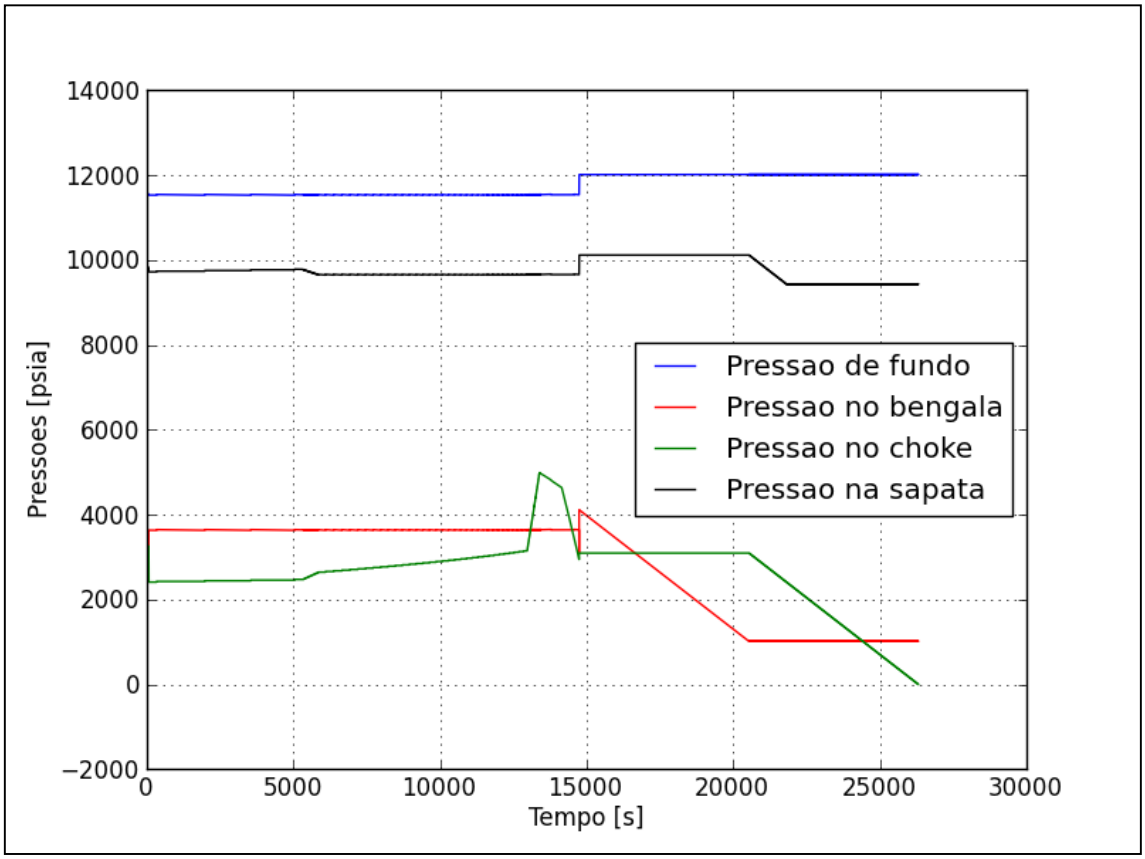


**Figura 16 - Comportamento das Pressões Durante o Método do Sondador**

## 4.3 Comparação com Modelo da Literatura

Visando a verificar a validade do modelo de controle de poço proposto neste trabalho, os dados apresentados no caso de estudo supracitado foram comparados aos do modelo matemático de kicks de gás em cenários de águas profundas apresentado por NUNES, 2002.

As Figuras 17 e 18 mostram a pressão no choke ao longo do tempo.

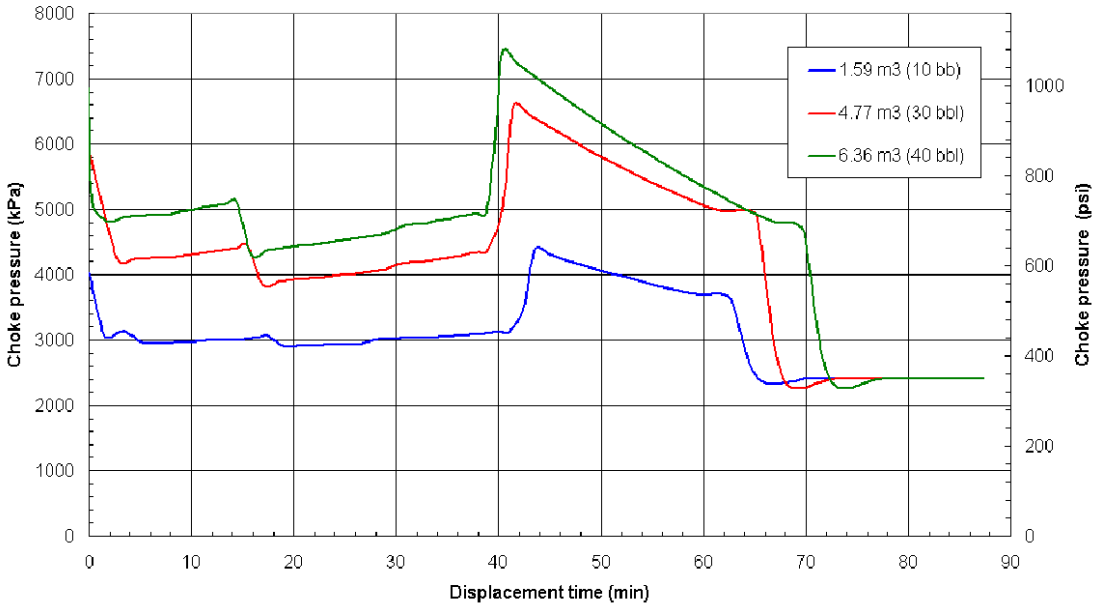


**Figure 17 - Pressão no choke (NUNES,2002)**

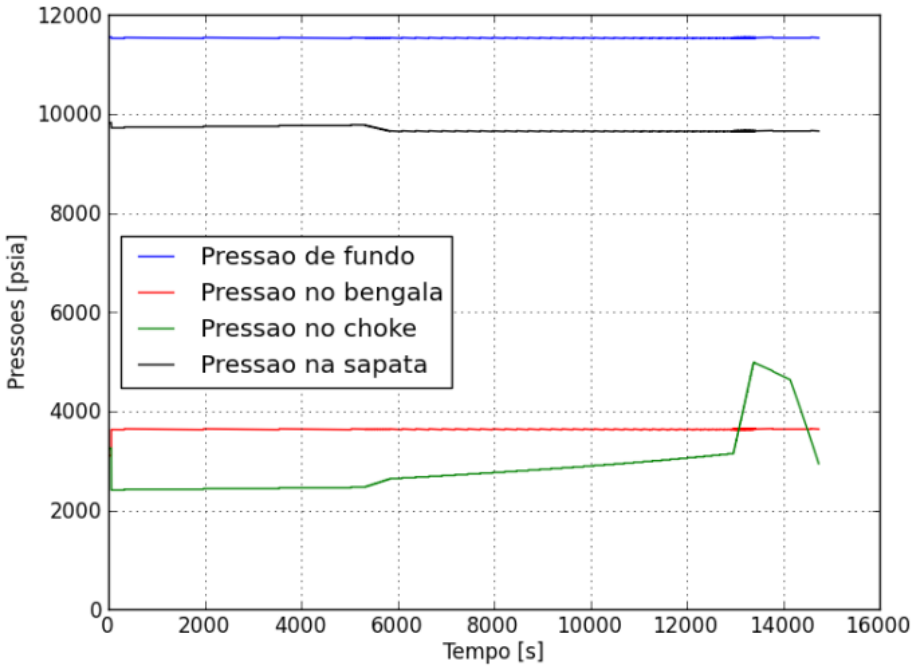


**Figure 18 - – Pressão no Choke para o caso estudado**

Ao se comparar os dois casos, é importante ressaltar que as configurações utilizadas foram diferentes e que o foco principal é comparar qualitativamente os dois casos. Como é possível verificar através das Figuras 17 e 18, a pressão no *choke* apresenta uma tendência comportamental muito semelhante, o que mostra uma consistência lógica do programa. Porém, é necessário que, no futuro, os mesmos dados utilizados pelo autor sejam replicados no modelo, para que uma análise quantitativa possa ser realizada e assim, melhor comprovar os resultados obtidos.

# 5. CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este trabalho apresentou o desenvolvimento de um código elaborado em *Python* para a simulação de um poço sofrendo um *kick* de gás e o processo de circulação do mesmo através do Método do Sondador, onde o principal objetivo era calcular as pressões no *choke* necessárias ao longo do tempo para concluir de forma segura a circulação completa do kick.

Uma revisão bibliográfica foi elaborada com o intuito de apresentar alguns conceitos básicos da indústria aos leitores e demonstrar de forma resumida alguns trabalhos feitos no passado por diversos autores dentro do mesmo tema.

Em seguida, o processo de criação do programa foi introduzido ao leitor, esclarecendo as principais formas de cálculos utilizadas, assim como o equacionamento utilizado e a metodologia criada para efetuar o controle do poço durante a circulação do *kick*. Tanto a primeira quanto a segunda circulação do método utilizado tiveram suas lógicas de raciocínio explicadas para facilitar a entendimento do código escrito.

Um Caso teórico foi testado e apresentado para calibrar o programa com os resultados da literatura e assim, conseguir simular o caso de interesse de forma mais segura e com resultados mais precisos.

O caso de interesse foi simulado e seus resultados apresentados e explicados, sendo possível analisar o comportamento das pressões durante todas as etapas da circulação. Além disso, outros parâmetros como o volume do *kick* e seu comprimento são monitorados e apresentados, assim como à medida que uma lama mais pesada entra no poço durante a segunda circulação, as pressões no *choke* também são calculadas para evitar, além de um novo *kick*, uma possível fratura na sapata do último revestimento.

Entretanto, diversas considerações foram feitas visando facilitar os métodos de cálculo e a elaboração como um todo do programa. O fato do gás ter sido estudado como ideal e o gradiente de temperatura do poço ser isotérmico, são fatores que possuem influência sobre os resultados e que podem facilmente ser aplicados em futuros trabalhos.

Por fim, como outras sugestões, a utilização de um escoamento do tipo bolha, uma vez que o mesmo possui resultados mais próximos com a realidade, da mesma forma que a utilização de um fluido a base óleo, para poder levar em consideração a solubilidade do gás no mesmo.

# 6. REFERÊNCIAS BIBLIOGRÁFICAS


AIRD, P.: **Drilling & Engineering: Introduction to Well Control**, 2009;

BEGGS, H. D.; BRILL, J. P.: **A Study of Two-phase Flow in Inclined Pipes**, Journal of Petroleum Technology, 607 – 617, Maio de 1973;

CABRAL, C. A. O.; TEIXEIRA, L. da S. P.; MOURA, R. N.; CHAMBRIARD, M.: **Investigação do Incidente de Vazamento de Petróleo no Campo de Frade**, Relatório nº 48610.003638/2012-92, Julho de 2012;

DAVIS, R. M.; TAYLOR, G. I.: **The Mechanism of Large Bubbles Rising Through Liquids in Tubes**, Proceedings of Royal Society (London), 200, Ser. A, 375, 1950;

DIMITRESCU, D. T.: **Strömung an Einer Luftblase im Senkrechten Rohr**, Z.angew, Math, Mech, 23, 139-149, 1943;

EKRANN, S., ROMMETVEIT, R., Rogaland Research Inst.: **A Simulator for Gas Kicks in Oil-Based Drilling Muds**, SPE Annual Technical Conference and Exhibition, Las Vegas, Nevada, 14182-MS, 22-26, Setembro de 1985;

ERCO/ENERGY RESOURCES CO. INC.: **Ixtoc Oil Spill Assessment, Final Report, Executive Summary Prepared for the US Bureau of Land Management**, Contract No. AA851-CTO-71; Março de 1982;

FAKHRO, K.M.: **Environment Protection, Ministry of Health, Review on the National and Regional Response to Oil Spill in the Arabian Gulf**, Middle East Oil Show, Bahrain, 21450-MS, 16-19, Novembro de 1991;

GAINES, T.H., Union Oil Co. of California: **Oil Pollution Control Efforts, Santa Barbara, California**, Journal of Petroleum Technology, 2752-PA, Dezembro de 2012;

HAEGH, T.; ROSSEMYR, I. L., Continental Shelf Institute.: **A Comparison of Weathering Processes of Oil from the Bravo And the Ixtoc Blowouts**, Offshore Technology Conference, 5-8, 3702-MS , Houston, Texas, Maio de 1980;

HOBEROCK, L. L.; STANBERY, S. R.: **Pressure Dynamics in Well During Gas Kicks: Part 1 – Fluid Lines Dynamics**, Journal of Petroleum Technology, 1357 – 1366, Agosto de 1981;

HOBEROCK, L. L.; STANBERY, S. R.: **Pressure Dynamics in Well During Gas Kicks: Part 2 – Component Models and Results**, Journal of Petroleum Technology, 1367 – 1378, Agosto de 1981;

HUTCHFUL, E.: **Texaco's Funiwa-5 Oil Well Blow-Out**, Journal of African Marxists, Rivers State, Nigeria, Março de 1985;

LEBLANC, J. L; LEWIS, R. L. A Mathematical Model of a Gas Kick. Journal of Petroleum Technology, p. 888-889, 1968.

MCANDREWS, K.L., The University of Texas at Austin.: **Consequences of Macondo: A Summary of Recently Proposed and Enacted Changes to US Offshore Drilling Safety and Environmental Regulation**; SPE Americas E&P Health, Safety, Security, and Environmental Conference, 21-23, 143718-MS, Houston, Texas, Março de 2011;

NEGRÃO, A. F.; MAIDLA, E. E.: **Optimization of Flow Rate Selection for Kick Control**, 64th Annual Conference and Exhibition of the Society of Petroleum Engineers, San Antonio, Texas: [s.n.]. 1989;

NICKENS, H. V.: **A Dynamic Computer Model of a Kicking Well**, SPE Drilling engineering, 158 – 173, Junho 1987;

NICKLIN, D.J., WILKES, J. O.; GREGORY, G. A.: **A Intermittent Two-Phase Flow in Vertical Tubes**, Trans. Inst. Chem. Engrs, 40, 61-68, 1962;

NUNES, J. O. L.: **Estudo do Controle de Poços em operações de Perfuração em Águas Profundas e Ultra Profundas**, UNICAMP, Dissertação de mestrado, 2002;

OHARA, S.: **Improved Method for Selecting Kick Tolerance During Deepwater Drilling Operations**, Louisiana State University, Doctor Thesis, Maio de 1996;

ORKIZEWSKI, J.: **Prediction Two-phase Pressure Drops in Vertical Pipes**, Journal of Petroleum Technology, 829 – 838, 1967;

PODIO, A. L.; YANG, A. P.: **Well Control Simulator for IBM Personal Computer**, IADC/SPE 14737. [S.I.]: [s.n.], 1986;

ROCHA, L.; AZEVEDO, C.: **Projeto de Poços de Petróleo**, 2ª Ed. Rio de Janeiro: Interciência, 2009;

SANTOS, O. L. A.: **A Mathematical Model of a Gas kick When Drilling in Deep Waters**, Colorado School of Mines, MS Thesis. [S.I.], 1982;

WHITE, D.B., WALTON, I.C., Schlumberger Cambridge Research.: **Computer Model for Kicks in Water- and Oil-Based Muds**, SPE/IADC Drilling Conference, 19975-MS, Houston, Texas, 1990.

# Anexo

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 07 12:29:05 2012

@author: rsantos
"""

# -*- coding: utf-8 -*-
"""
Created on Tue Oct 16 08:40:12 2012

@author: rsantos
"""

"""================================================================================
======
Método do Sondador v1.0
================================================================================
==="""

from numpy import pi
from numpy import array
import pylab
import numpy
import math


# Leitura dos inputs do arquivo txt e conversão de unidades
input = open('input.txt','r')
while True:
    linha = input.readline()
    if len(linha) == 0:
        break
    linha = linha.split()

    if linha[0] == 'profundidade':
        profundidade = float(linha[2])
    elif linha[0] == 'profundidade_linha_choke':
        profundidade_linha_choke = float(linha[2])
    elif linha[0] == 'profundidade_sapata':
        profundidade_sapata = float(linha[2])
    elif linha[0] == 'altura_trecho':
        altura_trecho = float(linha[2])
    elif linha[0] == 'diametro_externo_coluna':
        diametro_externo_coluna = float(linha[2]) * 0.0254 # m
    elif linha[0] == 'diametro_interno_coluna':
        diametro_interno_coluna = float(linha[2]) * 0.0254 # m
    elif linha[0] == 'diametro_interno_revestimento':
        diametro_interno_revestimento = float(linha[2]) * 0.0254 # m
    elif linha[0] == 'diametro_poco_aberto':
        diametro_poco_aberto = float(linha[2]) * 0.0254 # m
```

```
    elif linha[0] == 'diametro_linha_choke':
        diametro_linha_choke = float(linha[2]) * 0.0254 # m
    elif linha[0] == 'vazao':
        vazao = float(linha[2]) * 0.0000630901964 # m³/s
    elif linha[0] == 'k':
        k = float(linha[2])
    elif linha[0] == 'n':
        n = float(linha[2])
    elif linha[0] == 'densidade_fluido':
        densidade_fluido = float(linha[2]) * 1000 / 8.33 # kg/m³
    elif linha[0] == 'volume_kick':
        volume_kick = float(linha[2]) * 0.1589873 # m³
    elif linha[0] == 'profundidade_entrada_kick':
        profundidade_entrada_kick = float(linha[2])
    elif linha[0] == 'densidade_gas':
        densidade_gas = float(linha[2]) * 119.8264 # kg/m³
    elif linha[0] == 'vazao_gas':
        vazao_gas = float(linha[2]) * 0.002649788 # m³/s
    elif linha[0] == 'velocidade_gas':
        velocidade_gas = float(linha[2]) * 0.00508 # m/s
    elif linha[0] == 'viscosidade_gas':
        viscosidade_gas = float(linha[2])
    elif linha[0] == 'gravidade':
        gravidade = float(linha[2])
    elif linha[0] == 'temp_superficie':
        temp_superficie = float(linha[2])
    elif linha[0] == 'trechos_usados':
        trechos_usados = float(linha[2])
    elif linha[0] == 'vazao_controlada':
        vazao_controlada = float(linha[2]) * 0.002649788 # m³/s
    elif linha[0] == 'gradiente_pressao_poros':
        gradiente_pressao_poros = float(linha[2]) * 119.8264 # kg/m³
    elif linha[0] == 'gradiente_pressao_fratura':
        gradiente_pressao_fratura = float(linha[2]) * 119.8264 # kg/m³
    elif linha[0] == 'constante_gases':
        constante_gases = float(linha[2])
    elif linha[0] == 'peso_molecular':
        peso_molecular = float(linha[2]) / 1000 # kg/mol
    elif linha[0] == 'permeabilidade':
        permeabilidade = float(linha[2]) * 0.0000000000009869 / 1000 # m²
    elif linha[0] == 'porosidade':
        porosidade = float(linha[2])
    elif linha[0] == 'espessura':
        espessura = float(linha[2])
    else:
        pass
input.close()


# Criação dos vetores com relação a geometria do poço
# Lógica - pos0 = linha de choke; pos1 = altura da sapata; pos3 = final
do poço
diametro_interno = [diametro_linha_choke, diametro_interno_coluna,
diametro_interno_coluna]
```

```
diametro_externo = [0, diametro_externo_coluna, diametro_externo_coluna]
diametro_anular = [0, diametro_interno_revestimento,
diametro_poco_aberto]
area_interna = [(pi * (diametro_interno[0]**2)) / 4, (pi *
(diametro_interno[1]**2))/4, (pi * (diametro_interno[2]**2))/4]
area_anular = [0, (pi * ((diametro_anular[1]**2 -
diametro_externo[1]**2)))/4, (pi * ((diametro_anular[2]**2 -
diametro_externo[2]**2)))/4]
volume_interno = [area_interna[0] * altura_trecho, area_interna[1] *
altura_trecho, area_interna[2] * altura_trecho]
volume_anular = [0, area_anular[1] * altura_trecho, area_anular[2] *
altura_trecho]


# Criação dos vetores com relação as propriedades do fluido
# Lógica - Mesma utilizada acima
velocidade_fluido_interno = [vazao / area_interna[0], vazao /
area_interna[1], vazao / area_interna[2]]
taxa_deformacao_interno = [8 * velocidade_fluido_interno[0] /
diametro_interno[0], 8 * velocidade_fluido_interno[1] /
diametro_interno[1], 8 * velocidade_fluido_interno[2] /
diametro_interno[2]]
viscosidade_fluido_interno = [k * taxa_deformacao_interno[0]**( n - 1 ),
k * taxa_deformacao_interno[1]**( n - 1 ), k *
taxa_deformacao_interno[2]**( n - 1 )]
num_Reynolds_interno = [densidade_fluido / viscosidade_fluido_interno[0]
* diametro_interno[0] * velocidade_fluido_interno[0], densidade_fluido /
viscosidade_fluido_interno[1] * diametro_interno[1] *
velocidade_fluido_interno[1], densidade_fluido /
viscosidade_fluido_interno[2] * diametro_interno[2] *
velocidade_fluido_interno[2]]
fator_atrito_interno = [16/num_Reynolds_interno[0],
16/num_Reynolds_interno[1], 16/num_Reynolds_interno[2]]

velocidade_fluido_anular = [0, vazao / area_anular[1], vazao /
area_anular[2]]
diametro_hidraulico = [diametro_anular[0] - diametro_externo[0],
diametro_anular[1] - diametro_externo[1], diametro_anular[2] -
diametro_externo[2]]
taxa_deformacao_anular = [0, 12 * velocidade_fluido_anular[1] /
diametro_hidraulico[1], 12 * velocidade_fluido_anular[2] /
diametro_hidraulico[2]]
viscosidade_fluido_anular = [0, k * taxa_deformacao_anular[1]**( n - 1 ),
k * taxa_deformacao_anular[2]**( n - 1 )]
num_Reynolds_anular = [0, densidade_fluido / viscosidade_fluido_anular[1]
* diametro_hidraulico[1] * velocidade_fluido_anular[1], densidade_fluido
/ viscosidade_fluido_anular[2] * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
fator_atrito_anular = [0, 24 / num_Reynolds_anular[1], 24 /
num_Reynolds_anular[2]]

num_Reynolds_gas = [(densidade_gas / viscosidade_gas) *
diametro_linha_choke * velocidade_fluido_interno[0], (densidade_gas /
viscosidade_gas) * diametro_hidraulico[1] * velocidade_fluido_anular[1],
```

```
(densidade_gas / viscosidade_gas) * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
fator_atrito_gas = [24 / num_Reynolds_gas[0], 24 / num_Reynolds_gas[1],
24 / num_Reynolds_gas[2]]


# Criação dos vetores das pressões e perdas de carga
hidrostatica = []
perda_carga_interno = []
perda_carga_anular_revestido = []
perda_carga_anular_poco_aberto = []
perda_carga_anular = []
perda_carga_choke = []
for i in range(1,int(profundidade+1)):
    hidrostatica.append(densidade_fluido * gravidade * altura_trecho)

perda_carga_interno.append(2*densidade_fluido*fator_atrito_interno[1]*alt
ura_trecho*velocidade_fluido_interno[1]**2/diametro_interno[1])

perda_carga_anular_revestido.append(2*densidade_fluido*fator_atrito_anula
r[1]*altura_trecho*velocidade_fluido_anular[1]**2/diametro_hidraulico[1])

perda_carga_anular_poco_aberto.append(2*densidade_fluido*fator_atrito_anu
lar[2]*altura_trecho*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
    perda_carga_anular.append(0)

perda_carga_choke.append(2*densidade_fluido*fator_atrito_interno[0]*altur
a_trecho*velocidade_fluido_interno[0]**2/diametro_interno[0])
    if i ==1:
        hidrostatica[i-1] = hidrostatica[i-1]
        perda_carga_interno[i-1] = 0
        perda_carga_anular_revestido[i-1] = 0
        perda_carga_anular_poco_aberto[i-1] = 0
        perda_carga_anular[i-1] = 0
        perda_carga_choke[i-1] = perda_carga_choke[i-1]
    elif i > 1 and i <= (profundidade_linha_choke +1):
        hidrostatica[i-1] = hidrostatica[i-1] + hidrostatica[i-2]
        perda_carga_interno[i-1] = 0
        perda_carga_anular_revestido[i-1] = 0
        perda_carga_anular_poco_aberto[i-1] = 0
        perda_carga_anular[i-1] = 0
        perda_carga_choke[i-1] = perda_carga_choke[i-1] +
perda_carga_choke[i-2]
    elif i > (profundidade_linha_choke +1) and i <= (profundidade_sapata
+1):
        hidrostatica[i-1] = hidrostatica[i-1] + hidrostatica[i-2]
        perda_carga_interno[i-1] = perda_carga_interno[i-1] +
perda_carga_interno[i-2]
        perda_carga_anular_revestido[i-1] =
perda_carga_anular_revestido[i-1] + perda_carga_anular_revestido[i-2]
        perda_carga_anular_poco_aberto[i-1] = 0
        perda_carga_anular[i-1] = perda_carga_anular_revestido[i-1]
        perda_carga_choke[i-1] = 0
```

```
    else:
        hidrostatica[i-1] = hidrostatica[i-1] + hidrostatica[i-2]
        perda_carga_interno[i-1] = perda_carga_interno[i-1] +
perda_carga_interno[i-2]
        perda_carga_anular_revestido[i-1] = 0
        perda_carga_anular_poco_aberto[i-1] =
perda_carga_anular_poco_aberto[i-1] + perda_carga_anular_poco_aberto[i-2]
        perda_carga_anular[i-1] = perda_carga_anular_poco_aberto[i-1] +
perda_carga_anular[int(profundidade_sapata)]
        perda_carga_choke[i-1] = 0


# Pressão de fundo (pelo anular em Pa)
pressao_fundo = hidrostatica[int(profundidade-1)] +
perda_carga_anular[int(profundidade-1)] +
perda_carga_choke[int(profundidade_linha_choke-1)]


# Pressão na sapata (pelo anular em Pa)
pressao_sapata = hidrostatica[int(profundidade_sapata-1)] +
perda_carga_anular[int(profundidade_sapata-1)] +
perda_carga_choke[int(profundidade_linha_choke-1)]


# Pressão de bombeio (somente as perdas de carga em Pa)
pressao_bombeio = perda_carga_interno[int(profundidade-1)] +
perda_carga_anular[int(profundidade-1)] +
perda_carga_choke[int(profundidade_linha_choke-1)]


# Momento de entrada do gás
# Poço é fechado, sem circulação, vazão de entrada do gás constante
# Sistema mantido até atingir a Pressão de Poros
pressao_poros = gradiente_pressao_poros * gravidade * profundidade
pos_gas = []
volume_gas_trecho = []
volume_gas_extra_trecho = []
for i in range(1, int(profundidade_entrada_kick+1)):
    pos_gas.append(i)
    volume_gas_trecho.append(0)
    volume_gas_extra_trecho.append(0)
pos_gas.reverse()

for i in pos_gas:
    if i == profundidade_entrada_kick:
        volume_gas_trecho[i-1] = volume_anular[2]
        volume_gas_extra_trecho[i-1] = volume_kick - volume_anular[2]
    else:
        if volume_gas_extra_trecho[i] > volume_anular[2]:
            volume_gas_trecho[i-1] = volume_anular[2]
            volume_gas_extra_trecho[i-1] = volume_gas_extra_trecho[i] -
volume_gas_trecho[i-1]
        else:
            volume_gas_trecho[i-1] = volume_gas_extra_trecho[i]
```

```
            volume_gas_extra_trecho[i-1] = volume_gas_extra_trecho[i] -
volume_gas_trecho[i-1]


passo_tempo = 1 # segundos
altura_kick = (volume_kick / area_anular[2])
pressao_fundo_kick_vetor = []
pressao_fundo_kick = (pressao_fundo * volume_kick / (volume_kick))
altura_fluido = profundidade - altura_kick
hidrostatica_gas = densidade_gas * gravidade * altura_kick
pressao_bengala_kick_vetor = []
pressao_choke_kick_vetor = []
pressao_sapata_kick_vetor = []

# Modelagem do reservatório
# Regime Transiente
# Obter vazão de entrada do kick
constante_euler = 1.781
raio_poco = diametro_poco_aberto / 2
volume_controle = volume_kick
while pressao_poros > pressao_fundo_kick:

    compressibilidade_gas = 2 / (pressao_poros + pressao_fundo_kick)

    aux = (4 * permeabilidade * passo_tempo) / (viscosidade_gas *
constante_euler * porosidade * raio_poco * (compressibilidade_gas**2))
    aux_termo = numpy.log(aux)
    vazao_gas_reservatorio = ((pressao_poros**2) -
(pressao_fundo_kick**2)) * (1 / aux_termo) * ((2 * pi * permeabilidade *
espessura) / viscosidade_gas) * (1 / pressao_fundo_kick)

    pressao_fundo_kick = (pressao_fundo_kick* (volume_controle +
(vazao_gas_reservatorio*passo_tempo))) / (volume_controle)
    pressao_bengala_kick_vetor.append(pressao_fundo_kick -
hidrostatica[int(profundidade-1)])
    pressao_choke_kick_vetor.append(pressao_fundo_kick -
(hidrostatica[int(altura_fluido)] + hidrostatica_gas))

pressao_sapata_kick_vetor.append(pressao_choke_kick_vetor[passo_tempo-1]
+ hidrostatica[int(profundidade_sapata-1)])
    pressao_fundo_kick_vetor.append(pressao_fundo_kick)
    passo_tempo = passo_tempo + 1
    volume_controle = volume_controle + vazao_gas_reservatorio


#Cálculo da velocidade do gás
# Pos = choke, anular revestido, poço aberto
Co = 1.2
velocidade_escorregamento = [0.0345 * ((gravidade *
diametro_interno[0])**0.5), 0.0345 * ((gravidade *
diametro_hidraulico[1])**0.5), 0.0345 * ((gravidade *
diametro_hidraulico[2])**0.5)]
velocidade_gas = [(Co * velocidade_fluido_interno[0]) +
velocidade_escorregamento[0], (Co * velocidade_fluido_anular[1]) +
```

```
velocidade_escorregamento[1], (Co * velocidade_fluido_anular[2]) +
velocidade_escorregamento[2]]


# Início da migração do gás
# Poço fechado
# Duração da migração = 1 min
avanco = (1 * velocidade_gas[2])


for j in range(1,60):
    pos = profundidade_entrada_kick - avanco * (j-1) - altura_kick
    pos_base = profundidade_entrada_kick - avanco * (j-1)
    hidrostatica_fluido = densidade_fluido * gravidade * (profundidade -
pos_base)
    hidrostatica_fluido2 = (densidade_fluido * gravidade * (pos))
    pressao_fundo_kick_vetor.append(pressao_fundo_kick +
hidrostatica_fluido)

    pressao_choke_kick_vetor.append(pressao_fundo_kick -
hidrostatica_fluido2)

pressao_sapata_kick_vetor.append(pressao_choke_kick_vetor[passo_tempo+j-
2] + hidrostatica[int(profundidade_sapata-1)])

pressao_bengala_kick_vetor.append(pressao_fundo_kick_vetor[passo_tempo+j-
2] - hidrostatica[int(profundidade-1)])



# Momento da abertura do poço
# Expansão do gás controlada
pos_topo = pos
temperatura_gas = 333.15 # CONSTANTE POR ENQUANTO
velocidade_gas_poco_aberto = [velocidade_gas[0], velocidade_gas[1],
velocidade_gas[2]]
pos_topo_vetor = []
pos_base_vetor = []
tamanho_kick = []
volume_vetor = []
valor_convergencia = 69000 #345000
temperatura_superficie = 298 #K
temperatura_ponto = temperatura_superficie + 0.333*(pos_topo) # FAZER O
PERFIL DE TEMPERATURA!
volume_real = volume_kick
i = 0
volume_controle = volume_kick
avanco2_base = (velocidade_gas[2])
pressao_choke = 0
auxiliar = 0
while pos_base > 0.0:
    step = 6900000

    print j
```

```
    if pos_topo > profundidade_sapata:
        pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))
        pos_topo = pos_base - (volume_real / area_anular[2])



        if pos_topo <= profundidade_sapata:
            volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
            pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])



    elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
        pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))
        volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
        pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])


        aux = j
        avanco2_base_aux = avanco2_base



    elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
        pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
        pos_base = pos_base_aux - (avanco2_base * (j-aux))

        pos_topo = pos_base - (volume_real / area_anular[1])


        if pos_topo <= profundidade_linha_choke:
            volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
            pos_topo = (profundidade_linha_choke) - ((volume_real -
volume_antes_choke) / area_interna[0])


    elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:

        if pos_topo <= 0:
            pos_topo = 0
            pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
```

```
            pos_base = pos_base_aux - (avanco2_base * (j-aux))

        else:
            pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
            pos_base = pos_base_aux - (avanco2_base * (j-aux))
            if pos_base - (volume_real / area_anular[1]) >=
profundidade_linha_choke:
                pos_topo = pos_base - (volume_real / area_anular[1])
            else:
                volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                pos_topo = (profundidade_linha_choke) - ((volume_real -
volume_antes_choke) / area_interna[0])

            if pos_topo <= 0:
                pos_topo = 0



        aux2 = j
        avanco2_base_aux2 = avanco2_base


    elif pos_topo <= profundidade_linha_choke and pos_base <=
profundidade_linha_choke:
        if pos_topo <= 0:
            pos_topo = 0
            pos_base_aux = profundidade_entrada_kick - (avanco*58 +
avanco2_base_aux*(aux-58) + avanco2_base_aux2*(aux2-aux))
            pos_base = pos_base_aux - (avanco2_base * (j-aux2))
        else:
            pos_base_aux = profundidade_entrada_kick - (avanco*58 +
avanco2_base_aux*(aux-58) + avanco2_base_aux2*(aux2-aux))
            pos_base = pos_base_aux - (avanco2_base * (j-aux2))
            pos_topo = pos_base - (volume_real/area_interna[0])




    else:
        break



#    if pos_topo > profundidade_sapata:
#        taxa = vazao_controlada /  area_anular[2]
#        velocidade_fluido_gas = [velocidade_fluido_interno[0] + taxa,
velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] + taxa]
#    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
#        taxa = vazao_controlada /  area_anular[1]
#        velocidade_fluido_gas = [velocidade_fluido_interno[0] + taxa,
velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] + taxa]
```

```
#    else:
#        taxa = vazao_controlada /  area_interna[0]
#        velocidade_fluido_gas = [velocidade_fluido_interno[0] + taxa,
velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] + taxa]


#    print 'volume_kick=', volume_kick
#    print 'volume_real=', volume_real
#    print 'volume_controle[i]=', volume_controle[i]
#    print 'outpost 01'
#    raw_input()


#    if pos_topo > profundidade_sapata:
#
#        taxa = (volume_real - volume_controle) /  area_anular[2]
#        if taxa < 0:
#            velocidade_fluido_gas = [velocidade_fluido_interno[0]+
velocidade_gas[0], velocidade_fluido_anular[1] + velocidade_gas[1],
velocidade_fluido_anular[2] + velocidade_gas[2]]
#        else:
#            velocidade_fluido_gas = [velocidade_fluido_interno[0] +
velocidade_gas[0] + taxa, velocidade_fluido_anular[1] + velocidade_gas[1]
+ taxa, velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
#
#        taxa = (volume_real - volume_controle) /  area_anular[1]
#        if taxa < 0:
#            velocidade_fluido_gas = [velocidade_fluido_interno[0]+
velocidade_gas[0], velocidade_fluido_anular[1] + velocidade_gas[1],
velocidade_fluido_anular[2] + velocidade_gas[2]]
#        else:
#            velocidade_fluido_gas = [velocidade_fluido_interno[0] +
velocidade_gas[0] + taxa, velocidade_fluido_anular[1] + velocidade_gas[1]
+ taxa, velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#    else:
#
#        taxa = (volume_real - volume_controle) /  area_interna[0]
#        if taxa < 0:
#            velocidade_fluido_gas = [velocidade_fluido_interno[0]+
velocidade_gas[0], velocidade_fluido_anular[1] + velocidade_gas[1],
velocidade_fluido_anular[2] + velocidade_gas[2]]
#        else:
#            velocidade_fluido_gas = [velocidade_fluido_interno[0] +
velocidade_gas[0] + taxa, velocidade_fluido_anular[1] + velocidade_gas[1]
+ taxa, velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#


    taxa = 0
    velocidade_fluido_gas = [velocidade_fluido_interno[0] +
velocidade_gas[0] + taxa, velocidade_fluido_anular[1] + velocidade_gas[1]
+ taxa, velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
```

```
#    if j > 16433:
#        if pos_topo > profundidade_linha_choke:
#            taxa = 0
#            velocidade_fluido_gas = [velocidade_fluido_interno[0],
velocidade_fluido_anular[1], velocidade_fluido_anular[2]]
#        else:
#            taxa = (volume_real - volume_controle[i]) / area_interna[0]
#            if taxa <= 0:
#                velocidade_fluido_gas = [velocidade_fluido_interno[0],
velocidade_fluido_anular[1], velocidade_fluido_anular[2]]
#            else:
#                velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]

#    print 'taxa=', taxa
#    print 'outpost 02'
#    raw_input()



# TimeStep de 1 segundo
    if pos_topo > profundidade_sapata:
        avanco2_base = (velocidade_gas[2])
    elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
        avanco2_base = (velocidade_gas[2])
    elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
        avanco2_base = (velocidade_gas[1])
    elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
        avanco2_base = (velocidade_gas[1])
    elif pos_topo > 0 and pos_base <= profundidade_linha_choke:
        avanco2_base = (velocidade_gas[0])
    elif pos_topo <= 0 and pos_base <= profundidade_linha_choke:
        avanco2_base = (velocidade_gas[0])



# Cálculo das propriedades do fluido com a nova velocidade
    taxa_deformacao = [8 * velocidade_fluido_gas[0] /
diametro_interno[0], 12 * velocidade_fluido_gas[1] /
diametro_hidraulico[1], 12 * velocidade_fluido_gas[2] /
diametro_hidraulico[2]]
    viscosidade_fluido = [k * taxa_deformacao[0]**( n - 1 ), k *
taxa_deformacao[1]**( n - 1 ), k * taxa_deformacao[2]**( n - 1 )]
    num_Reynolds = [densidade_fluido / viscosidade_fluido[0] *
diametro_interno[0] * velocidade_fluido_gas[0], densidade_fluido /
viscosidade_fluido[1] * diametro_hidraulico[1] *
velocidade_fluido_gas[1], densidade_fluido / viscosidade_fluido[2] *
diametro_hidraulico[2] * velocidade_fluido_gas[2]]
```

```
    fator_atrito = [16/num_Reynolds[0], 24 / num_Reynolds[1], 24 /
num_Reynolds[2]]
    num_Reynolds_gas = [(densidade_gas / viscosidade_gas) *
diametro_linha_choke * velocidade_fluido_interno[0], (densidade_gas /
viscosidade_gas) * diametro_hidraulico[1] * velocidade_fluido_anular[1],
(densidade_gas / viscosidade_gas) * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
    fator_atrito_gas = [16 / num_Reynolds_gas[0], 24 /
num_Reynolds_gas[1], 24 / num_Reynolds_gas[2]]



    hidrostatica_fluido = densidade_fluido * gravidade * (profundidade -
pos_base)
    if pos_topo > profundidade_sapata:
        hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
        hidrostatica_gas = densidade_gas * gravidade * (pos_base -
pos_topo)
    elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
        hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
        hidrostatica_gas = densidade_gas * gravidade * (pos_base -
pos_topo)
    else:
        hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
        hidrostatica_gas = densidade_gas * gravidade * (pos_base -
pos_topo)


# Cálculo das perdas de carga
    perda_carga_coluna_total =
2*densidade_fluido*fator_atrito_interno[1]*profundidade*velocidade_fluido
_interno[1]**2/diametro_interno[1]


    if pos_topo > profundidade_sapata:
        perda_carga_gas = (2 * densidade_gas * fator_atrito_gas[2] *
(pos_base - pos_topo) * (velocidade_gas_poco_aberto[2]**2)) /
diametro_hidraulico[2]
    elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
        perda_carga_gas = ((2 * densidade_gas * fator_atrito_gas[2] *
(pos_base - profundidade_sapata) * (velocidade_gas_poco_aberto[2]**2)) /
diametro_hidraulico[2]) + ((2 * densidade_gas * fator_atrito_gas[1] *
(profundidade_sapata - pos_topo) * (velocidade_gas_poco_aberto[1]**2)) /
diametro_hidraulico[1])
    elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
```

```
        perda_carga_gas = ((2 * densidade_gas * fator_atrito_gas[1] *
(pos_base - pos_topo) * (velocidade_gas_poco_aberto[1]**2)) /
diametro_hidraulico[1])
    elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
        perda_carga_gas = ((2 * densidade_gas * fator_atrito_gas[1] *
(pos_base - profundidade_linha_choke) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1]) + ((2 *
densidade_gas * fator_atrito_gas[0] * (profundidade_linha_choke -
pos_topo) * (velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])
    else:
        perda_carga_gas = ((2 * densidade_gas * fator_atrito_gas[0] *
(pos_base - pos_topo) * (velocidade_gas_poco_aberto[0]**2)) /
diametro_interno[0])




    if pos_topo > profundidade_linha_choke:

        perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*profundidade_linha_choke*velocidade_fl
uido_gas[0]**2/diametro_interno[0]
    else:

        perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*pos_topo*velocidade_fluido_gas[0]**2/d
iametro_interno[0]



    if pos_topo > profundidade_sapata:
        perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
    elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
        perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(pos_topo -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
    else:
        perda_carga_anular_revestido_total = 0




    if pos_topo > profundidade_sapata:
        perda_carga_anular_poco_aberto_total =
2*densidade_fluido*fator_atrito[2]*(pos_topo -
profundidade_sapata)*velocidade_fluido_gas[2]**2/diametro_hidraulico[2]
    else:
        perda_carga_anular_poco_aberto_total = 0
```

```
    if pos_base > profundidade:
        perda_carga_abaixo_kick = 0
    elif pos_base > profundidade_sapata:
        perda_carga_abaixo_kick =
2*densidade_fluido*fator_atrito[2]*(profundidade -
pos_base)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2]
    elif pos_base <= profundidade_sapata and pos_base >
profundidade_linha_choke:
        perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
pos_base)*velocidade_fluido_anular[1]**2/diametro_hidraulico[1]) +
(2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
    elif pos_base <= profundidade_linha_choke and pos_base > 0:
        perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke -
pos_base)*velocidade_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
    else:
        perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke)*velocidade
_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])


# Cálculo da perda de carga na região bifásica

    if pos_topo > profundidade_sapata:
        velocidade_superficial_gas = velocidade_gas[2]
        velocidade_superficial_liquido = velocidade_fluido_gas[2]
        velocidade_escorregamento_bifasico = velocidade_escorregamento[2]
        fator_atrito_monofasico = fator_atrito[2]
        diametro_bifasico = diametro_hidraulico[2]
    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
        velocidade_superficial_gas = velocidade_gas[1]
        velocidade_superficial_liquido = velocidade_fluido_gas[1]
        velocidade_escorregamento_bifasico = velocidade_escorregamento[1]
        fator_atrito_monofasico = fator_atrito[1]
        diametro_bifasico = diametro_hidraulico[1]
    else:
        velocidade_superficial_gas = velocidade_gas[0]
```

```
        velocidade_superficial_liquido = velocidade_fluido_gas[0]
        velocidade_escorregamento_bifasico = velocidade_escorregamento[0]
        fator_atrito_monofasico = fator_atrito[0]
        diametro_bifasico = diametro_interno[0]


    fracao_vazio = velocidade_superficial_gas /
(Co*(velocidade_superficial_gas + velocidade_superficial_liquido) +
velocidade_escorregamento_bifasico)
    holdup_liquido = 1 - fracao_vazio


    aux_ln = (fracao_vazio / (holdup_liquido**2))
    aux_ln_termo = numpy.log(aux_ln)

    skin = aux_ln_termo / (-0.0523 + (3.182*aux_ln_termo) -
(0.8725*(aux_ln_termo**2)) + (0.01853*(aux_ln_termo**4)))
    aux_exponencial = math.exp(skin)
    fator_atrito_bifasico = fator_atrito_monofasico * aux_exponencial


    velocidade_mistura = (velocidade_superficial_liquido*holdup_liquido)
+ (velocidade_superficial_gas*fracao_vazio)
    densidade_mistura = (densidade_fluido*holdup_liquido) +
(densidade_gas*fracao_vazio)


    perda_carga_bifasico =
(fator_atrito_bifasico*densidade_mistura*(velocidade_mistura**2)*(pos_bas
e-pos_topo)) / (2*gravidade*diametro_bifasico)



# Bloco de controle da pressão de fundo através do choke
    pressao_fundo_calculado = hidrostatica_gas + hidrostatica_fluido2 +
hidrostatica_fluido + perda_carga_bifasico + perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total
+ perda_carga_abaixo_kick  + pressao_choke
#    print 'pressao=', pressao_fundo_calculado * 0.000145
#    print 'pressao_kick=', pressao_fundo_kick * 0.000145
#    print 'volume_real=', volume_real
#    print 'hidrostatica_fluido2=', hidrostatica_fluido2 * 0.000145
#    print 'hidrostatica_fluido=', hidrostatica_fluido * 0.000145
#    print 'hidrostatica_gas=', hidrostatica_gas * 0.000145
#    print 'perda_carga_gas=', perda_carga_gas * 0.000145
#    print 'perda_carga_choke_total=', perda_carga_choke_total * 0.000145
#    print 'perda_carga_anular_revestido_total=',
perda_carga_anular_revestido_total * 0.000145
#    print'perda_carga_anular_poco_aberto=',
perda_carga_anular_poco_aberto_total * 0.000145
#    print 'pressao_choke=', pressao_choke * 0.000145
#    print 'step=', step * 0.000145
#    print 'outpost 03'
```

```
#
#
#    if j > 1:
#        raw_input()


    while abs(pressao_fundo_calculado - pressao_fundo_kick) >
valor_convergencia:

        if pos_topo < profundidade_linha_choke:
            step = abs(pressao_fundo_calculado - pressao_fundo_kick)

        if pressao_fundo_calculado < pressao_fundo_kick -
valor_convergencia:
            if auxiliar == 0:
                pressao_choke = pressao_fundo_kick -
pressao_fundo_calculado
            else:
                pressao_choke = pressao_choke + step
            if pressao_choke < 0:
                pressao_choke = 0
#            volume_controle.append(volume_real)
#            i = i + 1
            volume_real = (pressao_fundo_kick * volume_kick ) /
(hidrostatica_fluido2+ perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total
+ pressao_choke)
            #volume_real = volume_real
            #volume_real = volume_real + vazao_controlada
            pressao_gas = (volume_kick * pressao_fundo_kick) /
volume_real
            densidade_gas = (pressao_gas * peso_molecular) /
(temperatura_gas * constante_gases)

            if pos_topo > profundidade_sapata:
                pos_topo = pos_base - (volume_real / area_anular[2])

                pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))

                if pos_topo <= profundidade_sapata:
                    volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
                    pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])



            elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:

                volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
```

```
                pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])

                pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))
                aux = j
                avanco2_base_aux = avanco2_base



            elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:

                pos_topo = pos_base - (volume_real / area_anular[1])
                pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
                pos_base = pos_base_aux - (avanco2_base * (j-aux))

                if pos_topo <= profundidade_linha_choke:
                    volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                    pos_topo = (profundidade_linha_choke) - ((volume_real
- volume_antes_choke) / area_interna[0])


            elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:

                if pos_topo <= 0:
                    pos_topo = 0

                else:
                    if pos_base - (volume_real / area_anular[1]) >=
profundidade_linha_choke:
                        pos_topo = pos_base - (volume_real /
area_anular[1])
                    else:
                        volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                        pos_topo = (profundidade_linha_choke) -
((volume_real - volume_antes_choke) / area_interna[0])

                    if pos_topo <= 0:
                        pos_topo = 0

                pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
                pos_base = pos_base_aux - (avanco2_base * (j-aux))

                aux2 = j
                avanco2_base_aux2 = avanco2_base

            elif pos_topo <= profundidade_linha_choke and pos_base <=
profundidade_linha_choke:
```

```
                if pos_topo <= 0:
                    pos_topo = 0
                else:
                    pos_topo = pos_base - (volume_real/area_interna[0])

                pos_base_aux = profundidade_entrada_kick - (avanco*58 +
avanco2_base_aux*(aux-58) + avanco2_base_aux2*(aux2-aux))
                pos_base = pos_base_aux - (avanco2_base * (j-aux2))


            else:
                break



    #    if pos_topo > profundidade_sapata:
    #        taxa = vazao_controlada /  area_anular[2]
    #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
    #    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
    #        taxa = vazao_controlada /  area_anular[1]
    #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
    #    else:
    #        taxa = vazao_controlada /  area_interna[0]
    #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]



#            if pos_topo > profundidade_sapata:
#
#                taxa = (volume_real - volume_controle[i]) /
area_anular[2]
#                if taxa < 0:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                else:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#            elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
#
#                taxa = (volume_real - volume_controle[i]) /
area_anular[1]
#                if taxa < 0:
```

```
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                 else:
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#             else:
#
#                 taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                 if taxa < 0:
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                 else:
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]


#             if pos_topo > profundidade_linha_choke:
#                 taxa = 0
#                 velocidade_fluido_gas = [velocidade_fluido_interno[0],
velocidade_fluido_anular[1], velocidade_fluido_anular[2]]
#             else:
#                 taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                 if taxa <= 0:
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                 else:
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0] + taxa, velocidade_fluido_anular[1] + taxa,
velocidade_fluido_anular[2] + taxa]
#
    # TimeStep de 1 segundo
            if pos_topo > profundidade_sapata:
                avanco2_topo = (velocidade_fluido_gas[2])
                avanco2_base = (velocidade_gas[2])
            elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                avanco2_topo = (velocidade_fluido_gas[1])
                avanco2_base = (velocidade_gas[2])
            elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                avanco2_topo = (velocidade_fluido_gas[1])
                avanco2_base = (velocidade_gas[1])
            elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                avanco2_topo = (velocidade_fluido_gas[0])
```

```
                avanco2_base = (velocidade_gas[1])
            elif pos_topo > 0 and pos_base <= profundidade_linha_choke:
                avanco2_topo = (velocidade_fluido_gas[0])
                avanco2_base = (velocidade_gas[0])
            elif pos_topo <= 0 and pos_base <= profundidade_linha_choke:
                avanco2_topo = 0
                avanco2_base = (velocidade_gas[0])



    # Cálculo das propriedades do fluido com a nova velocidade
            taxa_deformacao = [8 * velocidade_fluido_gas[0] /
diametro_interno[0], 12 * velocidade_fluido_gas[1] /
diametro_hidraulico[1], 12 * velocidade_fluido_gas[2] /
diametro_hidraulico[2]]
            viscosidade_fluido = [k * taxa_deformacao[0]**( n - 1 ), k *
taxa_deformacao[1]**( n - 1 ), k * taxa_deformacao[2]**( n - 1 )]
            num_Reynolds = [densidade_fluido / viscosidade_fluido[0] *
diametro_interno[0] * velocidade_fluido_gas[0], densidade_fluido /
viscosidade_fluido[1] * diametro_hidraulico[1] *
velocidade_fluido_gas[1], densidade_fluido / viscosidade_fluido[2] *
diametro_hidraulico[2] * velocidade_fluido_gas[2]]
            fator_atrito = [16/num_Reynolds[0], 24 / num_Reynolds[1], 24
/ num_Reynolds[2]]
            num_Reynolds_gas = [(densidade_gas / viscosidade_gas) *
diametro_linha_choke * velocidade_fluido_interno[0], (densidade_gas /
viscosidade_gas) * diametro_hidraulico[1] * velocidade_fluido_anular[1],
(densidade_gas / viscosidade_gas) * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
            fator_atrito_gas = [16 / num_Reynolds_gas[0], 24 /
num_Reynolds_gas[1], 24 / num_Reynolds_gas[2]]



            hidrostatica_fluido = densidade_fluido * gravidade *
(profundidade - pos_base)
            if pos_topo > profundidade_sapata:
                hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
                hidrostatica_gas = densidade_gas * gravidade * (pos_base
- pos_topo)
            elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
                hidrostatica_gas = densidade_gas * gravidade * (pos_base
- pos_topo)
            else:
                hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
                hidrostatica_gas = densidade_gas * gravidade * (pos_base
- pos_topo)
```

```
    # Cálculo das perdas de carga
            perda_carga_coluna_total =
2*densidade_fluido*fator_atrito_interno[1]*profundidade*velocidade_fluido
_interno[1]**2/diametro_interno[1]


            if pos_topo > profundidade_sapata:
                perda_carga_gas = (2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]
            elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - profundidade_sapata) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]) + ((2 *
densidade_gas * fator_atrito_gas[1] * (profundidade_sapata - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
            elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
            elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - profundidade_linha_choke) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1]) + ((2 *
densidade_gas * fator_atrito_gas[0] * (profundidade_linha_choke -
pos_topo) * (velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])
            else:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[0] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])




            if pos_topo > profundidade_linha_choke:

                perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*profundidade_linha_choke*velocidade_fl
uido_gas[0]**2/diametro_interno[0]
            else:

                perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*pos_topo*velocidade_fluido_gas[0]**2/d
iametro_interno[0]



            if pos_topo > profundidade_sapata:
                perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
```

```
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
            elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(pos_topo -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
            else:
                perda_carga_anular_revestido_total = 0



            if pos_topo > profundidade_sapata:
                perda_carga_anular_poco_aberto_total =
2*densidade_fluido*fator_atrito[2]*(pos_topo -
profundidade_sapata)*velocidade_fluido_gas[2]**2/diametro_hidraulico[2]
            else:
                perda_carga_anular_poco_aberto_total = 0



            if pos_base > profundidade:
                perda_carga_abaixo_kick = 0
            elif pos_base > profundidade_sapata:
                perda_carga_abaixo_kick =
2*densidade_fluido*fator_atrito[2]*(profundidade -
pos_base)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2]
            elif pos_base <= profundidade_sapata and pos_base >
profundidade_linha_choke:
                perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
pos_base)*velocidade_fluido_anular[1]**2/diametro_hidraulico[1]) +
(2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
            elif pos_base <= profundidade_linha_choke and pos_base > 0:
                perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke -
pos_base)*velocidade_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
            else:
                perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke)*velocidade
_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
```

```
# Cálculo da perda de carga na região bifásica

            if pos_topo > profundidade_sapata:
                velocidade_superficial_gas = velocidade_gas[2]
                velocidade_superficial_liquido = velocidade_fluido_gas[2]
                velocidade_escorregamento_bifasico =
velocidade_escorregamento[2]
                fator_atrito_monofasico = fator_atrito[2]
                diametro_bifasico = diametro_hidraulico[2]
            elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
                velocidade_superficial_gas = velocidade_gas[1]
                velocidade_superficial_liquido = velocidade_fluido_gas[1]
                velocidade_escorregamento_bifasico =
velocidade_escorregamento[1]
                fator_atrito_monofasico = fator_atrito[1]
                diametro_bifasico = diametro_hidraulico[1]
            else:
                velocidade_superficial_gas = velocidade_gas[0]
                velocidade_superficial_liquido = velocidade_fluido_gas[0]
                velocidade_escorregamento_bifasico =
velocidade_escorregamento[0]
                fator_atrito_monofasico = fator_atrito[0]
                diametro_bifasico = diametro_interno[0]

            fracao_vazio = velocidade_superficial_gas /
(Co*(velocidade_superficial_gas + velocidade_superficial_liquido) +
velocidade_escorregamento_bifasico)
            holdup_liquido = 1 - fracao_vazio

            aux_ln = (fracao_vazio / (holdup_liquido**2))
            aux_ln_termo = numpy.log(aux_ln)

            skin = aux_ln_termo / (-0.0523 + (3.182*aux_ln_termo) -
(0.8725*(aux_ln_termo**2)) + (0.01853*(aux_ln_termo**4)))
            aux_exponencial = math.exp(skin)
            fator_atrito_bifasico = fator_atrito_monofasico *
aux_exponencial


            velocidade_mistura =
(velocidade_superficial_liquido*holdup_liquido) +
(velocidade_superficial_gas*fracao_vazio)
            densidade_mistura = (densidade_fluido*holdup_liquido) +
(densidade_gas*fracao_vazio)


            perda_carga_bifasico =
(fator_atrito_bifasico*densidade_mistura*(velocidade_mistura**2)*(pos_bas
e-pos_topo)) / (2*gravidade*diametro_bifasico)
```

```
    # Bloco de controle da pressão de fundo através do choke
            pressao_fundo_calculado = hidrostatica_gas +
hidrostatica_fluido2 + hidrostatica_fluido + perda_carga_bifasico +
perda_carga_choke_total + perda_carga_anular_revestido_total +
perda_carga_anular_poco_aberto_total + perda_carga_abaixo_kick  +
pressao_choke
           # print 'pressao1=', pressao_fundo_calculado*0.000145


            if pressao_fundo_calculado > pressao_fundo_kick +
valor_convergencia:
                step = step/2
                if pos_topo < profundidade_linha_choke:
                    step = abs(pressao_fundo_calculado -
pressao_fundo_kick)
                pressao_choke = pressao_choke - step
                if pressao_choke < 0:
                    pressao_choke = 0

#                volume_controle.append(volume_real)
#                i = i + 1
                volume_real = (pressao_fundo_kick * volume_kick ) /
(hidrostatica_fluido2+ perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total
+ pressao_choke)
                #volume_real = volume_real
                #volume_real = volume_real + vazao_controlada
                pressao_gas = (volume_kick * pressao_fundo_kick) /
volume_real
                densidade_gas = (pressao_gas * peso_molecular) /
(temperatura_gas * constante_gases)

                if pos_topo > profundidade_sapata:
                    pos_topo = pos_base - (volume_real / area_anular[2])

                    pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))

                    if pos_topo <= profundidade_sapata:
                        volume_antes_sapata = area_anular[2] * (pos_base
- profundidade_sapata)
                        pos_topo = (profundidade_sapata) - ((volume_real
- volume_antes_sapata) / area_anular[1])



                elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:

                    volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
                    pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])
```

```
                    pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))
                    aux = j
                    avanco2_base_aux = avanco2_base



                elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:

                    pos_topo = pos_base - (volume_real / area_anular[1])
                    pos_base_aux = profundidade_entrada_kick - (avanco *
58 + avanco2_base_aux * (aux-58))
                    pos_base = pos_base_aux - (avanco2_base * (j-aux))

                    if pos_topo <= profundidade_linha_choke:
                        volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                        pos_topo = (profundidade_linha_choke) -
((volume_real - volume_antes_choke) / area_interna[0])


                elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:

                    if pos_topo <= 0:
                        pos_topo = 0

                    else:
                        if pos_base - (volume_real / area_anular[1]) >=
profundidade_linha_choke:
                            pos_topo = pos_base - (volume_real /
area_anular[1])
                        else:
                            volume_antes_choke = area_anular[1] *
(pos_base - profundidade_linha_choke)
                            pos_topo = (profundidade_linha_choke) -
((volume_real - volume_antes_choke) / area_interna[0])

                        if pos_topo <= 0:
                            pos_topo = 0

                    pos_base_aux = profundidade_entrada_kick - (avanco *
58 + avanco2_base_aux * (aux-58))
                    pos_base = pos_base_aux - (avanco2_base * (j-aux))

                    aux2 = j
                    avanco2_base_aux2 = avanco2_base


                elif pos_topo <= profundidade_linha_choke and pos_base <=
profundidade_linha_choke:
                    if pos_topo <= 0:
```

```
                    pos_topo = 0
                else:
                    pos_topo = pos_base -
(volume_real/area_interna[0])

                pos_base_aux = profundidade_entrada_kick - (avanco*58
+ avanco2_base_aux*(aux-58) + avanco2_base_aux2*(aux2-aux))
                pos_base = pos_base_aux - (avanco2_base * (j-aux2))


            else:
                break



        #    if pos_topo > profundidade_sapata:
        #        taxa = vazao_controlada /  area_anular[2]
        #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
        #    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
        #        taxa = vazao_controlada /  area_anular[1]
        #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
        #    else:
        #        taxa = vazao_controlada /  area_interna[0]
        #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]



#                if pos_topo > profundidade_sapata:
#
#                    taxa = (volume_real - volume_controle[i]) /
area_anular[2]
#                    if taxa < 0:
#                        velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                    else:
#                        velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#                elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
#
#                    taxa = (volume_real - volume_controle[i]) /
area_anular[1]
#                    if taxa < 0:
```

```
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                     else:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#                 else:
#
#                     taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                     if taxa < 0:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                     else:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#

#                 if pos_topo > profundidade_linha_choke:
#                     taxa = 0
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                 else:
#                     taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                     if taxa <= 0:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                     else:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0] + taxa, velocidade_fluido_anular[1] + taxa,
velocidade_fluido_anular[2] + taxa]
#
        # TimeStep de 1 segundo
                if pos_topo > profundidade_sapata:
                    avanco2_topo = (velocidade_fluido_gas[2])
                    avanco2_base = (velocidade_gas[2])
                elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                    avanco2_topo = (velocidade_fluido_gas[1])
                    avanco2_base = (velocidade_gas[2])
                elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                    avanco2_topo = (velocidade_fluido_gas[1])
                    avanco2_base = (velocidade_gas[1])
                elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
```

```
                    avanco2_topo = (velocidade_fluido_gas[0])
                    avanco2_base = (velocidade_gas[1])
                elif pos_topo > 0 and pos_base <=
profundidade_linha_choke:
                    avanco2_topo = (velocidade_fluido_gas[0])
                    avanco2_base = (velocidade_gas[0])
                elif pos_topo <= 0 and pos_base <=
profundidade_linha_choke:
                    avanco2_topo = 0
                    avanco2_base = (velocidade_gas[0])



        # Cálculo das propriedades do fluido com a nova velocidade
                taxa_deformacao = [8 * velocidade_fluido_gas[0] /
diametro_interno[0], 12 * velocidade_fluido_gas[1] /
diametro_hidraulico[1], 12 * velocidade_fluido_gas[2] /
diametro_hidraulico[2]]
                viscosidade_fluido = [k * taxa_deformacao[0]**( n - 1 ),
k * taxa_deformacao[1]**( n - 1 ), k * taxa_deformacao[2]**( n - 1 )]
                num_Reynolds = [densidade_fluido / viscosidade_fluido[0]
* diametro_interno[0] * velocidade_fluido_gas[0], densidade_fluido /
viscosidade_fluido[1] * diametro_hidraulico[1] *
velocidade_fluido_gas[1], densidade_fluido / viscosidade_fluido[2] *
diametro_hidraulico[2] * velocidade_fluido_gas[2]]
                fator_atrito = [16/num_Reynolds[0], 24 / num_Reynolds[1],
24 / num_Reynolds[2]]
                num_Reynolds_gas = [(densidade_gas / viscosidade_gas) *
diametro_linha_choke * velocidade_fluido_interno[0], (densidade_gas /
viscosidade_gas) * diametro_hidraulico[1] * velocidade_fluido_anular[1],
(densidade_gas / viscosidade_gas) * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
                fator_atrito_gas = [16 / num_Reynolds_gas[0], 24 /
num_Reynolds_gas[1], 24 / num_Reynolds_gas[2]]



                hidrostatica_fluido = densidade_fluido * gravidade *
(profundidade - pos_base)
                if pos_topo > profundidade_sapata:
                    hidrostatica_fluido2 = (densidade_fluido * gravidade
* (pos_topo))
                    hidrostatica_gas = densidade_gas * gravidade *
(pos_base - pos_topo)
                elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                    hidrostatica_fluido2 = (densidade_fluido * gravidade
* (pos_topo))
                    hidrostatica_gas = densidade_gas * gravidade *
(pos_base - pos_topo)
                else:
                    hidrostatica_fluido2 = (densidade_fluido * gravidade
* (pos_topo))
```

```
                    hidrostatica_gas = densidade_gas * gravidade *
(pos_base - pos_topo)


        # Cálculo das perdas de carga
                perda_carga_coluna_total =
2*densidade_fluido*fator_atrito_interno[1]*profundidade*velocidade_fluido
_interno[1]**2/diametro_interno[1]


                if pos_topo > profundidade_sapata:
                    perda_carga_gas = (2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]
                elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - profundidade_sapata) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]) + ((2 *
densidade_gas * fator_atrito_gas[1] * (profundidade_sapata - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
                elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
                elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - profundidade_linha_choke) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1]) + ((2 *
densidade_gas * fator_atrito_gas[0] * (profundidade_linha_choke -
pos_topo) * (velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])
                else:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[0] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])




                if pos_topo > profundidade_linha_choke:

                    perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*profundidade_linha_choke*velocidade_fl
uido_gas[0]**2/diametro_interno[0]
                else:

                    perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*pos_topo*velocidade_fluido_gas[0]**2/d
iametro_interno[0]
```

```
                if pos_topo > profundidade_sapata:
                    perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
                elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                    perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(pos_topo -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
                else:
                    perda_carga_anular_revestido_total = 0



                if pos_topo > profundidade_sapata:
                    perda_carga_anular_poco_aberto_total =
2*densidade_fluido*fator_atrito[2]*(pos_topo -
profundidade_sapata)*velocidade_fluido_gas[2]**2/diametro_hidraulico[2]
                else:
                    perda_carga_anular_poco_aberto_total = 0



                if pos_base > profundidade:
                    perda_carga_abaixo_kick = 0
                elif pos_base > profundidade_sapata:
                    perda_carga_abaixo_kick =
2*densidade_fluido*fator_atrito[2]*(profundidade -
pos_base)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2]
                elif pos_base <= profundidade_sapata and pos_base >
profundidade_linha_choke:
                    perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
pos_base)*velocidade_fluido_anular[1]**2/diametro_hidraulico[1]) +
(2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
                elif pos_base <= profundidade_linha_choke and pos_base >
0:
                    perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke -
pos_base)*velocidade_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
                else:
                    perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke)*velocidade
_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
```

```
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])



# Cálculo da perda de carga na região bifásica

                if pos_topo > profundidade_sapata:
                    velocidade_superficial_gas = velocidade_gas[2]
                    velocidade_superficial_liquido =
velocidade_fluido_gas[2]
                    velocidade_escorregamento_bifasico =
velocidade_escorregamento[2]
                    fator_atrito_monofasico = fator_atrito[2]
                    diametro_bifasico = diametro_hidraulico[2]
                elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
                    velocidade_superficial_gas = velocidade_gas[1]
                    velocidade_superficial_liquido =
velocidade_fluido_gas[1]
                    velocidade_escorregamento_bifasico =
velocidade_escorregamento[1]
                    fator_atrito_monofasico = fator_atrito[1]
                    diametro_bifasico = diametro_hidraulico[1]
                else:
                    velocidade_superficial_gas = velocidade_gas[0]
                    velocidade_superficial_liquido =
velocidade_fluido_gas[0]
                    velocidade_escorregamento_bifasico =
velocidade_escorregamento[0]
                    fator_atrito_monofasico = fator_atrito[0]
                    diametro_bifasico = diametro_interno[0]

                fracao_vazio = velocidade_superficial_gas /
(Co*(velocidade_superficial_gas + velocidade_superficial_liquido) +
velocidade_escorregamento_bifasico)
                holdup_liquido = 1 - fracao_vazio

                aux_ln = (fracao_vazio / (holdup_liquido**2))
                aux_ln_termo = numpy.log(aux_ln)

                skin = aux_ln_termo / (-0.0523 + (3.182*aux_ln_termo) -
(0.8725*(aux_ln_termo**2)) + (0.01853*(aux_ln_termo**4)))
                aux_exponencial = math.exp(skin)
                fator_atrito_bifasico = fator_atrito_monofasico *
aux_exponencial


                velocidade_mistura =
(velocidade_superficial_liquido*holdup_liquido) +
(velocidade_superficial_gas*fracao_vazio)
```

```
                densidade_mistura = (densidade_fluido*holdup_liquido) +
(densidade_gas*fracao_vazio)


                perda_carga_bifasico =
(fator_atrito_bifasico*densidade_mistura*(velocidade_mistura**2)*(pos_bas
e-pos_topo)) / (2*gravidade*diametro_bifasico)

        # Bloco de controle da pressão de fundo através do choke
                pressao_fundo_calculado = hidrostatica_gas +
hidrostatica_fluido2 + hidrostatica_fluido + perda_carga_bifasico +
perda_carga_choke_total + perda_carga_anular_revestido_total +
perda_carga_anular_poco_aberto_total + perda_carga_abaixo_kick  +
pressao_choke
                #print 'pressao2=', pressao_fundo_calculado*0.000145



        if pressao_fundo_calculado > pressao_fundo_kick +
valor_convergencia:
            if pos_topo < profundidade_linha_choke:
                step = abs(pressao_fundo_calculado - pressao_fundo_kick)
            pressao_choke = pressao_choke - step
            if pressao_choke < 0:
                pressao_choke = 0

#            volume_controle.append(volume_real)
#            i = i + 1
            volume_real = (pressao_fundo_kick * volume_kick ) /
(hidrostatica_fluido2+ perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total
+ pressao_choke)
            #volume_real = volume_real
            #volume_real = volume_real + vazao_controlada
            pressao_gas = (volume_kick * pressao_fundo_kick) /
volume_real
            densidade_gas = (pressao_gas * peso_molecular) /
(temperatura_gas * constante_gases)

            if pos_topo > profundidade_sapata:
                pos_topo = pos_base - (volume_real / area_anular[2])

                pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))

                if pos_topo <= profundidade_sapata:
                    volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
                    pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])
```

```
            elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:

                volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
                pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])

                pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))
                aux = j
                avanco2_base_aux = avanco2_base



            elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:

                pos_topo = pos_base - (volume_real / area_anular[1])
                pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
                pos_base = pos_base_aux - (avanco2_base * (j-aux))

                if pos_topo <= profundidade_linha_choke:
                    volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                    pos_topo = (profundidade_linha_choke) - ((volume_real
- volume_antes_choke) / area_interna[0])


            elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:

                if pos_topo <= 0:
                    pos_topo = 0

                else:
                    if pos_base - (volume_real / area_anular[1]) >=
profundidade_linha_choke:
                        pos_topo = pos_base - (volume_real /
area_anular[1])
                    else:
                        volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                        pos_topo = (profundidade_linha_choke) -
((volume_real - volume_antes_choke) / area_interna[0])

                    if pos_topo <= 0:
                        pos_topo = 0

                pos_base_aux = profundidade_entrada_kick - (avanco * 58 +
avanco2_base_aux * (aux-58))
                pos_base = pos_base_aux - (avanco2_base * (j-aux))
```

```
                aux2 = j
                avanco2_base_aux2 = avanco2_base


            elif pos_topo <= profundidade_linha_choke and pos_base <=
profundidade_linha_choke:
                if pos_topo <= 0:
                    pos_topo = 0
                else:
                    pos_topo = pos_base - (volume_real/area_interna[0])

                pos_base_aux = profundidade_entrada_kick - (avanco*58 +
avanco2_base_aux*(aux-58) + avanco2_base_aux2*(aux2-aux))
                pos_base = pos_base_aux - (avanco2_base * (j-aux2))


            else:
                break



    #    if pos_topo > profundidade_sapata:
    #        taxa = vazao_controlada /  area_anular[2]
    #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
    #    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
    #        taxa = vazao_controlada /  area_anular[1]
    #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
    #    else:
    #        taxa = vazao_controlada /  area_interna[0]
    #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]



#            if pos_topo > profundidade_sapata:
#
#                taxa = (volume_real - volume_controle[i]) /
area_anular[2]
#                if taxa < 0:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                else:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
```

```
#            elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
#
#                taxa = (volume_real - volume_controle[i]) /
area_anular[1]
#                if taxa < 0:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                else:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#            else:
#
#                taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                if taxa < 0:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                else:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#

#            if pos_topo > profundidade_linha_choke:
#                taxa = 0
#                velocidade_fluido_gas = [velocidade_fluido_interno[0],
velocidade_fluido_anular[1], velocidade_fluido_anular[2]]
#            else:
#                taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                if taxa <= 0:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                else:
#                    velocidade_fluido_gas =
[velocidade_fluido_interno[0] + taxa, velocidade_fluido_anular[1] + taxa,
velocidade_fluido_anular[2] + taxa]
#
    # TimeStep de 1 segundo
            if pos_topo > profundidade_sapata:
                avanco2_topo = (velocidade_fluido_gas[2])
                avanco2_base = (velocidade_gas[2])
            elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                avanco2_topo = (velocidade_fluido_gas[1])
                avanco2_base = (velocidade_gas[2])
```

```
            elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                avanco2_topo = (velocidade_fluido_gas[1])
                avanco2_base = (velocidade_gas[1])
            elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                avanco2_topo = (velocidade_fluido_gas[0])
                avanco2_base = (velocidade_gas[1])
            elif pos_topo > 0 and pos_base <= profundidade_linha_choke:
                avanco2_topo = (velocidade_fluido_gas[0])
                avanco2_base = (velocidade_gas[0])
            elif pos_topo <= 0 and pos_base <= profundidade_linha_choke:
                avanco2_topo = 0
                avanco2_base = (velocidade_gas[0])



    # Cálculo das propriedades do fluido com a nova velocidade
            taxa_deformacao = [8 * velocidade_fluido_gas[0] /
diametro_interno[0], 12 * velocidade_fluido_gas[1] /
diametro_hidraulico[1], 12 * velocidade_fluido_gas[2] /
diametro_hidraulico[2]]
            viscosidade_fluido = [k * taxa_deformacao[0]**( n - 1 ), k *
taxa_deformacao[1]**( n - 1 ), k * taxa_deformacao[2]**( n - 1 )]
            num_Reynolds = [densidade_fluido / viscosidade_fluido[0] *
diametro_interno[0] * velocidade_fluido_gas[0], densidade_fluido /
viscosidade_fluido[1] * diametro_hidraulico[1] *
velocidade_fluido_gas[1], densidade_fluido / viscosidade_fluido[2] *
diametro_hidraulico[2] * velocidade_fluido_gas[2]]







            fator_atrito = [16/num_Reynolds[0], 24 / num_Reynolds[1], 24
/ num_Reynolds[2]]
            num_Reynolds_gas = [(densidade_gas / viscosidade_gas) *
diametro_linha_choke * velocidade_fluido_interno[0], (densidade_gas /
viscosidade_gas) * diametro_hidraulico[1] * velocidade_fluido_anular[1],
(densidade_gas / viscosidade_gas) * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
            fator_atrito_gas = [16 / num_Reynolds_gas[0], 24 /
num_Reynolds_gas[1], 24 / num_Reynolds_gas[2]]



            hidrostatica_fluido = densidade_fluido * gravidade *
(profundidade - pos_base)
            if pos_topo > profundidade_sapata:
                hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
```

```
                hidrostatica_gas = densidade_gas * gravidade * (pos_base
- pos_topo)
            elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
                hidrostatica_gas = densidade_gas * gravidade * (pos_base
- pos_topo)
            else:
                hidrostatica_fluido2 = (densidade_fluido * gravidade *
(pos_topo))
                hidrostatica_gas = densidade_gas * gravidade * (pos_base
- pos_topo)


    # Cálculo das perdas de carga
            perda_carga_coluna_total =
2*densidade_fluido*fator_atrito_interno[1]*profundidade*velocidade_fluido
_interno[1]**2/diametro_interno[1]


            if pos_topo > profundidade_sapata:
                perda_carga_gas = (2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]
            elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - profundidade_sapata) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]) + ((2 *
densidade_gas * fator_atrito_gas[1] * (profundidade_sapata - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
            elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
            elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - profundidade_linha_choke) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1]) + ((2 *
densidade_gas * fator_atrito_gas[0] * (profundidade_linha_choke -
pos_topo) * (velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])
            else:
                perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[0] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])



            if pos_topo > profundidade_linha_choke:
```

```
                perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*profundidade_linha_choke*velocidade_fl
uido_gas[0]**2/diametro_interno[0]
            else:

                perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*pos_topo*velocidade_fluido_gas[0]**2/d
iametro_interno[0]



            if pos_topo > profundidade_sapata:
                perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
            elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(pos_topo -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
            else:
                perda_carga_anular_revestido_total = 0



            if pos_topo > profundidade_sapata:

                perda_carga_anular_poco_aberto_total =
2*densidade_fluido*fator_atrito[2]*(pos_topo -
profundidade_sapata)*velocidade_fluido_gas[2]**2/diametro_hidraulico[2]

            else:
                perda_carga_anular_poco_aberto_total = 0



            if pos_base > profundidade:
                perda_carga_abaixo_kick = 0
            elif pos_base > profundidade_sapata:
                perda_carga_abaixo_kick =
2*densidade_fluido*fator_atrito[2]*(profundidade -
pos_base)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2]
            elif pos_base <= profundidade_sapata and pos_base >
profundidade_linha_choke:
                perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
pos_base)*velocidade_fluido_anular[1]**2/diametro_hidraulico[1]) +
(2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
            elif pos_base <= profundidade_linha_choke and pos_base > 0:
```

```
                perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke -
pos_base)*velocidade_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
            else:
                perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke)*velocidade
_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])



# Cálculo da perda de carga na região bifásica

            if pos_topo > profundidade_sapata:
                velocidade_superficial_gas = velocidade_gas[2]
                velocidade_superficial_liquido = velocidade_fluido_gas[2]
                velocidade_escorregamento_bifasico =
velocidade_escorregamento[2]
                fator_atrito_monofasico = fator_atrito[2]
                diametro_bifasico = diametro_hidraulico[2]
            elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
                velocidade_superficial_gas = velocidade_gas[1]
                velocidade_superficial_liquido = velocidade_fluido_gas[1]
                velocidade_escorregamento_bifasico =
velocidade_escorregamento[1]
                fator_atrito_monofasico = fator_atrito[1]
                diametro_bifasico = diametro_hidraulico[1]
            else:
                velocidade_superficial_gas = velocidade_gas[0]
                velocidade_superficial_liquido = velocidade_fluido_gas[0]
                velocidade_escorregamento_bifasico =
velocidade_escorregamento[0]
                fator_atrito_monofasico = fator_atrito[0]
                diametro_bifasico = diametro_interno[0]

            fracao_vazio = velocidade_superficial_gas /
(Co*(velocidade_superficial_gas + velocidade_superficial_liquido) +
velocidade_escorregamento_bifasico)
            holdup_liquido = 1 - fracao_vazio

            aux_ln = (fracao_vazio / (holdup_liquido**2))
            aux_ln_termo = numpy.log(aux_ln)
```

```
            skin = aux_ln_termo / (-0.0523 + (3.182*aux_ln_termo) -
(0.8725*(aux_ln_termo**2)) + (0.01853*(aux_ln_termo**4)))
            aux_exponencial = math.exp(skin)
            fator_atrito_bifasico = fator_atrito_monofasico *
aux_exponencial


            velocidade_mistura =
(velocidade_superficial_liquido*holdup_liquido) +
(velocidade_superficial_gas*fracao_vazio)
            densidade_mistura = (densidade_fluido*holdup_liquido) +
(densidade_gas*fracao_vazio)


            perda_carga_bifasico =
(fator_atrito_bifasico*densidade_mistura*(velocidade_mistura**2)*(pos_bas
e-pos_topo)) / (2*gravidade*diametro_bifasico)


    # Bloco de controle da pressão de fundo através do choke
            pressao_fundo_calculado = hidrostatica_gas +
hidrostatica_fluido2 + hidrostatica_fluido + perda_carga_bifasico +
perda_carga_choke_total + perda_carga_anular_revestido_total +
perda_carga_anular_poco_aberto_total + perda_carga_abaixo_kick  +
pressao_choke
           # print 'pressao3=', pressao_fundo_calculado *0.000145


            if pressao_fundo_calculado < pressao_fundo_kick -
valor_convergencia:
                step = step/2
                if pos_topo < profundidade_linha_choke:
                    step = abs(pressao_fundo_calculado -
pressao_fundo_kick)
                pressao_choke = pressao_choke + step
                if pressao_choke < 0:
                    pressao_choke = 0
#                volume_controle.append(volume_real)
#                i = i + 1
                volume_real = (pressao_fundo_kick * volume_kick ) /
(hidrostatica_fluido2+ perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total
+ pressao_choke)
                #volume_real = volume_real
                #volume_real = volume_real + vazao_controlada
                pressao_gas = (volume_kick * pressao_fundo_kick) /
volume_real
                densidade_gas = (pressao_gas * peso_molecular) /
(temperatura_gas * constante_gases)

                if pos_topo > profundidade_sapata:
                    pos_topo = pos_base - (volume_real / area_anular[2])
```

```
                    pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))

                    if pos_topo <= profundidade_sapata:
                        volume_antes_sapata = area_anular[2] * (pos_base
- profundidade_sapata)
                        pos_topo = (profundidade_sapata) - ((volume_real
- volume_antes_sapata) / area_anular[1])



                elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:

                    volume_antes_sapata = area_anular[2] * (pos_base -
profundidade_sapata)
                    pos_topo = (profundidade_sapata) - ((volume_real -
volume_antes_sapata) / area_anular[1])

                    pos_base = profundidade_entrada_kick - (avanco * 58 +
avanco2_base * (j-58))
                    aux = j
                    avanco2_base_aux = avanco2_base



                elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:

                    pos_topo = pos_base - (volume_real / area_anular[1])
                    pos_base_aux = profundidade_entrada_kick - (avanco *
58 + avanco2_base_aux * (aux-58))
                    pos_base = pos_base_aux - (avanco2_base * (j-aux))

                    if pos_topo <= profundidade_linha_choke:
                        volume_antes_choke = area_anular[1] * (pos_base -
profundidade_linha_choke)
                        pos_topo = (profundidade_linha_choke) -
((volume_real - volume_antes_choke) / area_interna[0])


                elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:

                    if pos_topo <= 0:
                        pos_topo = 0

                    else:
                        if pos_base - (volume_real / area_anular[1]) >=
profundidade_linha_choke:
                            pos_topo = pos_base - (volume_real /
area_anular[1])
                        else:
```

```
                            volume_antes_choke = area_anular[1] *
(pos_base - profundidade_linha_choke)
                            pos_topo = (profundidade_linha_choke) -
((volume_real - volume_antes_choke) / area_interna[0])

                        if pos_topo <= 0:
                            pos_topo = 0

                    pos_base_aux = profundidade_entrada_kick - (avanco *
58 + avanco2_base_aux * (aux-58))
                    pos_base = pos_base_aux - (avanco2_base * (j-aux))

                    aux2 = j
                    avanco2_base_aux2 = avanco2_base


                elif pos_topo <= profundidade_linha_choke and pos_base <=
profundidade_linha_choke:
                    if pos_topo <= 0:
                        pos_topo = 0
                    else:
                        pos_topo = pos_base -
(volume_real/area_interna[0])

                    pos_base_aux = profundidade_entrada_kick - (avanco*58
+ avanco2_base_aux*(aux-58) + avanco2_base_aux2*(aux2-aux))
                    pos_base = pos_base_aux - (avanco2_base * (j-aux2))


                else:
                    break



        #    if pos_topo > profundidade_sapata:
        #        taxa = vazao_controlada /  area_anular[2]
        #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
        #    elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
        #        taxa = vazao_controlada /  area_anular[1]
        #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]
        #    else:
        #        taxa = vazao_controlada /  area_interna[0]
        #        velocidade_fluido_gas = [velocidade_fluido_interno[0] +
taxa, velocidade_fluido_anular[1] + taxa, velocidade_fluido_anular[2] +
taxa]



#                if pos_topo > profundidade_sapata:
```

```
#
#                     taxa = (volume_real - volume_controle[i]) /
area_anular[2]
#                     if taxa < 0:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                     else:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#                 elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
#
#                     taxa = (volume_real - volume_controle[i]) /
area_anular[1]
#                     if taxa < 0:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                     else:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#                 else:
#
#                     taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                     if taxa < 0:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                     else:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0] + velocidade_gas[0] + taxa,
velocidade_fluido_anular[1] + velocidade_gas[1] + taxa,
velocidade_fluido_anular[2] + velocidade_gas[2] + taxa]
#
#

#                 if pos_topo > profundidade_linha_choke:
#                     taxa = 0
#                     velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
#                 else:
#                     taxa = (volume_real - volume_controle[i]) /
area_interna[0]
#                     if taxa <= 0:
#                         velocidade_fluido_gas =
[velocidade_fluido_interno[0], velocidade_fluido_anular[1],
velocidade_fluido_anular[2]]
```

```
#                    else:
#                        velocidade_fluido_gas =
[velocidade_fluido_interno[0] + taxa, velocidade_fluido_anular[1] + taxa,
velocidade_fluido_anular[2] + taxa]
#
        # TimeStep de 1 segundo
                if pos_topo > profundidade_sapata:
                    avanco2_topo = (velocidade_fluido_gas[2])
                    avanco2_base = (velocidade_gas[2])
                elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                    avanco2_topo = (velocidade_fluido_gas[1])
                    avanco2_base = (velocidade_gas[2])
                elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                    avanco2_topo = (velocidade_fluido_gas[1])
                    avanco2_base = (velocidade_gas[1])
                elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                    avanco2_topo = (velocidade_fluido_gas[0])
                    avanco2_base = (velocidade_gas[1])
                elif pos_topo > 0 and pos_base <=
profundidade_linha_choke:
                    avanco2_topo = (velocidade_fluido_gas[0])
                    avanco2_base = (velocidade_gas[0])
                elif pos_topo <= 0 and pos_base <=
profundidade_linha_choke:
                    avanco2_topo = 0
                    avanco2_base = (velocidade_gas[0])



        # Cálculo das propriedades do fluido com a nova velocidade
                taxa_deformacao = [8 * velocidade_fluido_gas[0] /
diametro_interno[0], 12 * velocidade_fluido_gas[1] /
diametro_hidraulico[1], 12 * velocidade_fluido_gas[2] /
diametro_hidraulico[2]]
                viscosidade_fluido = [k * taxa_deformacao[0]**( n - 1 ),
k * taxa_deformacao[1]**( n - 1 ), k * taxa_deformacao[2]**( n - 1 )]
                num_Reynolds = [densidade_fluido / viscosidade_fluido[0]
* diametro_interno[0] * velocidade_fluido_gas[0], densidade_fluido /
viscosidade_fluido[1] * diametro_hidraulico[1] *
velocidade_fluido_gas[1], densidade_fluido / viscosidade_fluido[2] *
diametro_hidraulico[2] * velocidade_fluido_gas[2]]
                fator_atrito = [16/num_Reynolds[0], 24 / num_Reynolds[1],
24 / num_Reynolds[2]]

                num_Reynolds_gas = [(densidade_gas / viscosidade_gas) *
diametro_linha_choke * velocidade_fluido_interno[0], (densidade_gas /
viscosidade_gas) * diametro_hidraulico[1] * velocidade_fluido_anular[1],
(densidade_gas / viscosidade_gas) * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
                fator_atrito_gas = [16 / num_Reynolds_gas[0], 24 /
num_Reynolds_gas[1], 24 / num_Reynolds_gas[2]]
```

```
                hidrostatica_fluido = densidade_fluido * gravidade *
(profundidade - pos_base)
                if pos_topo > profundidade_sapata:
                    hidrostatica_fluido2 = (densidade_fluido * gravidade
* (pos_topo))
                    hidrostatica_gas = densidade_gas * gravidade *
(pos_base - pos_topo)
                elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                    hidrostatica_fluido2 = (densidade_fluido * gravidade
* (pos_topo))
                    hidrostatica_gas = densidade_gas * gravidade *
(pos_base - pos_topo)
                else:
                    hidrostatica_fluido2 = (densidade_fluido * gravidade
* (pos_topo))
                    hidrostatica_gas = densidade_gas * gravidade *
(pos_base - pos_topo)


        # Cálculo das perdas de carga
                perda_carga_coluna_total =
2*densidade_fluido*fator_atrito_interno[1]*profundidade*velocidade_fluido
_interno[1]**2/diametro_interno[1]


                if pos_topo > profundidade_sapata:
                    perda_carga_gas = (2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]
                elif pos_topo <= profundidade_sapata and pos_base >
profundidade_sapata:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[2] * (pos_base - profundidade_sapata) *
(velocidade_gas_poco_aberto[2]**2)) / diametro_hidraulico[2]) + ((2 *
densidade_gas * fator_atrito_gas[1] * (profundidade_sapata - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
                elif pos_topo > profundidade_linha_choke and pos_base <=
profundidade_sapata:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1])
                elif pos_topo <= profundidade_linha_choke and pos_base >
profundidade_linha_choke:
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[1] * (pos_base - profundidade_linha_choke) *
(velocidade_gas_poco_aberto[1]**2)) / diametro_hidraulico[1]) + ((2 *
densidade_gas * fator_atrito_gas[0] * (profundidade_linha_choke -
pos_topo) * (velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])
                else:
```

```
                    perda_carga_gas = ((2 * densidade_gas *
fator_atrito_gas[0] * (pos_base - pos_topo) *
(velocidade_gas_poco_aberto[0]**2)) / diametro_interno[0])



                if pos_topo > profundidade_linha_choke:

                    perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*profundidade_linha_choke*velocidade_fl
uido_gas[0]**2/diametro_interno[0]
                else:

                    perda_carga_choke_total =
2*densidade_fluido*fator_atrito[0]*pos_topo*velocidade_fluido_gas[0]**2/d
iametro_interno[0]



                if pos_topo > profundidade_sapata:
                    perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
                elif pos_topo <= profundidade_sapata and pos_topo >
profundidade_linha_choke:
                    perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito[1]*(pos_topo -
profundidade_linha_choke)*velocidade_fluido_gas[1]**2/diametro_hidraulico
[1]
                else:
                    perda_carga_anular_revestido_total = 0



                if pos_topo > profundidade_sapata:
                    perda_carga_anular_poco_aberto_total =
2*densidade_fluido*fator_atrito[2]*(pos_topo -
profundidade_sapata)*velocidade_fluido_gas[2]**2/diametro_hidraulico[2]
                else:
                    perda_carga_anular_poco_aberto_total = 0



                if pos_base > profundidade:
                    perda_carga_abaixo_kick = 0
                elif pos_base > profundidade_sapata:
                    perda_carga_abaixo_kick =
2*densidade_fluido*fator_atrito[2]*(profundidade -
pos_base)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2]
                elif pos_base <= profundidade_sapata and pos_base >
profundidade_linha_choke:
```

```
                    perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
pos_base)*velocidade_fluido_anular[1]**2/diametro_hidraulico[1]) +
(2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
                elif pos_base <= profundidade_linha_choke and pos_base >
0:
                    perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke -
pos_base)*velocidade_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])
                else:
                    perda_carga_abaixo_kick =
(2*densidade_fluido*fator_atrito[0]*(profundidade_linha_choke)*velocidade
_fluido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido*fator_atrito[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) + (2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
])



# Cálculo da perda de carga na região bifásica

                if pos_topo > profundidade_sapata:
                    velocidade_superficial_gas = velocidade_gas[2]
                    velocidade_superficial_liquido =
velocidade_fluido_gas[2]
                    velocidade_escorregamento_bifasico =
velocidade_escorregamento[2]
                    fator_atrito_monofasico = fator_atrito[2]
                    diametro_bifasico = diametro_hidraulico[2]
                elif pos_topo > profundidade_linha_choke and pos_topo <=
profundidade_sapata:
                    velocidade_superficial_gas = velocidade_gas[1]
                    velocidade_superficial_liquido =
velocidade_fluido_gas[1]
                    velocidade_escorregamento_bifasico =
velocidade_escorregamento[1]
                    fator_atrito_monofasico = fator_atrito[1]
                    diametro_bifasico = diametro_hidraulico[1]
                else:
                    velocidade_superficial_gas = velocidade_gas[0]
                    velocidade_superficial_liquido =
velocidade_fluido_gas[0]
                    velocidade_escorregamento_bifasico =
velocidade_escorregamento[0]
                    fator_atrito_monofasico = fator_atrito[0]
```

```
                    diametro_bifasico = diametro_interno[0]

                fracao_vazio = velocidade_superficial_gas /
(Co*(velocidade_superficial_gas + velocidade_superficial_liquido) +
velocidade_escorregamento_bifasico)
                holdup_liquido = 1 - fracao_vazio

                aux_ln = (fracao_vazio / (holdup_liquido**2))
                aux_ln_termo = numpy.log(aux_ln)

                skin = aux_ln_termo / (-0.0523 + (3.182*aux_ln_termo) -
(0.8725*(aux_ln_termo**2)) + (0.01853*(aux_ln_termo**4)))
                aux_exponencial = math.exp(skin)
                fator_atrito_bifasico = fator_atrito_monofasico *
aux_exponencial


                velocidade_mistura =
(velocidade_superficial_liquido*holdup_liquido) +
(velocidade_superficial_gas*fracao_vazio)
                densidade_mistura = (densidade_fluido*holdup_liquido) +
(densidade_gas*fracao_vazio)


                perda_carga_bifasico =
(fator_atrito_bifasico*densidade_mistura*(velocidade_mistura**2)*(pos_bas
e-pos_topo)) / (2*gravidade*diametro_bifasico)


        # Bloco de controle da pressão de fundo através do choke
                pressao_fundo_calculado = hidrostatica_gas +
hidrostatica_fluido2 + hidrostatica_fluido + perda_carga_bifasico +
perda_carga_choke_total + perda_carga_anular_revestido_total +
perda_carga_anular_poco_aberto_total + perda_carga_abaixo_kick  +
pressao_choke
               # print 'pressao4=', pressao_fundo_calculado*0.000145

        auxiliar = auxiliar + 1

#        print 'pressao_fundo_calculado=', pressao_fundo_calculado *
0.000145
#        print 'pressao_fundo_kick=', pressao_fundo_kick * 0.000145
#        print 'pressao_choke=', pressao_choke * 0.000145
#        print 'step=', step * 0.000145
#        print 'volume_real=', volume_real
#        print 'pos_topo=', pos_topo
#        print 'outpost 03.01', j
#        if j > 1:
#            raw_input()
```

```
    volume_controle = volume_real
    i = i + 1

    temperatura_ponto = temperatura_superficie + 0.333*(pos_topo)
    volume_real = (pressao_fundo_kick * volume_kick ) /
(hidrostatica_fluido2+ perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total
+ pressao_choke)

    if pos_topo <= 0:
        if pos_base > profundidade_linha_choke:
            volume_real = (profundidade_linha_choke*area_interna[0]) +
((pos_base-profundidade_linha_choke)*area_anular[1])
        else:
            volume_real = pos_base*area_interna[0]

#    if volume_real < volume_controle:
#        volume_real = volume_controle
#    else:
#        volume_real = volume_real
    #volume_real = volume_real
    #volume_real = volume_real + vazao_controlada
    pressao_gas = (volume_kick * pressao_fundo_kick) / volume_real
    densidade_gas = (pressao_gas * peso_molecular) / (temperatura_gas *
constante_gases)
    volume_total = (pressao_fundo_kick * volume_kick)/ 101352.9322
```

```
    tamanho_kick.append(pos_base - pos_topo)
    pos_topo_vetor.append(pos_topo)
    pos_base_vetor.append(pos_base)
    volume_vetor.append(volume_real)

    pressao_fundo_kick_vetor.append(pressao_fundo_calculado)
    pressao_choke_kick_vetor.append(pressao_choke)
    pressao_bengala_kick_vetor.append(pressao_fundo_calculado -
hidrostatica[int(profundidade-1)] + perda_carga_coluna_total)
    if pos_topo >= profundidade_sapata:
        pressao_sapata_kick_vetor.append(pressao_choke +
hidrostatica[int(profundidade_sapata-1)] + perda_carga_choke_total +
perda_carga_anular_revestido_total)
    elif pos_topo < profundidade_sapata and pos_base >=
profundidade_sapata:
        pressao_sapata_kick_vetor.append(pressao_choke +
hidrostatica_fluido2 + (densidade_gas*gravidade*(profundidade_sapata-
pos_topo)) + perda_carga_choke_total + perda_carga_anular_revestido_total
+ perda_carga_gas)
    else:
        pressao_sapata_kick_vetor.append(pressao_fundo_calculado -
(densidade_fluido*gravidade*(profundidade-profundidade_sapata)) -
(2*densidade_fluido*fator_atrito[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
]))




    j = j + 1


#    print 'volume_final_loop=', volume_real
#    print 'volume_total=', volume_total
#    print 'volume_comparacao=', (area_anular[2] * (pos_base - pos_topo))
    print 'pos_topo_final=', pos_topo
    print 'pos_base=', pos_base
#    print 'pressao_choke=', pressao_choke*0.000145
#    print 'pressao_fundo=', pressao_fundo_calculado*0.000145
#    print 'pressao_fundo_kick - convergencia=', (pressao_fundo_kick -
valor_convergencia) * 0.000145
#    print 'pressao_fundo_kick + convergencia=', (pressao_fundo_kick +
valor_convergencia) * 0.000145
    print 'pressao_choke=', pressao_choke * 0.000145
```

```
    print 'volume=', volume_real

#    if pos_topo < 1001:
#        raw_input()
#    print i
#    print "voltou"
#    print 'outpost 04'


# Término da primeira circulação
if pos_base < 0:
    pos_base = 0

if volume_real < 0:
    volume_real = 0


# Etapa de verificação dos resultados
step = 50 / 0.000145
pressao_fundo_2_circulacao = pressao_choke +
hidrostatica[int(profundidade-1)]
while pressao_fundo_2_circulacao <= pressao_poros:
    pressao_choke = pressao_choke + step
    pressao_fundo_2_circulacao = pressao_choke +
hidrostatica[int(profundidade-1)]


# Janela de pressões do choke
pressao_fratura = gradiente_pressao_fratura * gravidade *
profundidade_sapata

pressao_choke_minima = pressao_poros - hidrostatica[int(profundidade-1)]
pressao_choke_maxima = pressao_fratura -
hidrostatica[int(profundidade_sapata-1)]
pressao_choke_minima_vetor = []
pressao_choke_maxima_vetor = []


# Cálculo das pressões com o poço fechado
pressao_fundo_2_circulacao_vetor = []
pressao_choke_total_2_circulacao_vetor = []
pressao_bengala_2_circulacao_vetor = []
pressao_sapata_2_circulacao_vetor = []


pressao_fundo_kick_vetor.append(pressao_fundo_2_circulacao)
pressao_choke_kick_vetor.append(pressao_choke)
pressao_bengala_kick_vetor.append(pressao_fundo_2_circulacao -
hidrostatica[int(profundidade-1)])
pressao_sapata_kick_vetor.append(pressao_choke +
hidrostatica[int(profundidade_sapata -1)])

pressao_fundo_2_circulacao_vetor.append(pressao_fundo_2_circulacao)
pressao_choke_total_2_circulacao_vetor.append(pressao_choke)
```

```
pressao_bengala_2_circulacao_vetor.append(pressao_fundo_2_circulacao -
hidrostatica[int(profundidade-1)])
pressao_sapata_2_circulacao_vetor.append(pressao_choke +
hidrostatica[int(profundidade_sapata -1)])


# Início da segunda circulação
SIDPP = pressao_fundo_calculado - hidrostatica[int(profundidade-1)]

# Conversão das unidades - SI para Inglês
densidade_fluido_eq = densidade_fluido * 0.00043353 # psi/ft
profundidade_eq = profundidade * 3.28084 # ft
SIDPP_eq = SIDPP * 0.000145 # psia

# Cálculo da nova lama a ser utilizada
densidade_fluido_novo_eq = ((densidade_fluido_eq*profundidade_eq) +
SIDPP_eq) / (0.052*profundidade_eq) # psia/ft
densidade_fluido_novo = densidade_fluido_novo_eq * (1000/8.33) #2306.6587
# kg/m³

pos_fluido_novo = 0
pos_fluido_novo_vetor = []
passo_tempo = 1 # segundos
avanco_fluido_novo = velocidade_fluido_interno[1] * passo_tempo

# Propriedades do fluido novo
num_Reynolds_interno_fluido_novo = [densidade_fluido_novo /
viscosidade_fluido_interno[0] * diametro_interno[0] *
velocidade_fluido_interno[0],  densidade_fluido_novo /
viscosidade_fluido_interno[1] * diametro_interno[1] *
velocidade_fluido_interno[1], densidade_fluido_novo /
viscosidade_fluido_interno[2] * diametro_interno[2] *
velocidade_fluido_interno[2]]
fator_atrito_interno_fluido_novo =
[16/num_Reynolds_interno_fluido_novo[0],
16/num_Reynolds_interno_fluido_novo[1],
16/num_Reynolds_interno_fluido_novo[2]]
num_Reynolds_anular_fluido_novo = [0, densidade_fluido_novo /
viscosidade_fluido_anular[1] * diametro_hidraulico[1] *
velocidade_fluido_anular[1], densidade_fluido_novo /
viscosidade_fluido_anular[2] * diametro_hidraulico[2] *
velocidade_fluido_anular[2]]
fator_atrito_anular_fluido_novo = [0, 24 /
num_Reynolds_anular_fluido_novo[1], 24 /
num_Reynolds_anular_fluido_novo[2]]

# 1ª fase da segunda circulação - Fluido novo até a broca
# Manter pressão no choke constante
while pos_fluido_novo < profundidade:

# Cálculo da hidrostática
    hidrostatica_fluido_antigo = densidade_fluido * gravidade *
(profundidade - pos_fluido_novo)
```

```
    hidrostatica_fluido_novo = densidade_fluido_novo * gravidade *
pos_fluido_novo
    hidrostatica_anular = densidade_fluido * gravidade * profundidade
    hidrostatica_anular_sapata = densidade_fluido * gravidade *
profundidade_sapata

# Cálculo da perda de carga
    perda_carga_coluna_fluido_antigo =
2*densidade_fluido*fator_atrito_interno[1]*(profundidade -
pos_fluido_novo)*velocidade_fluido_interno[1]**2/diametro_interno[1]
    perda_carga_coluna_fluido_novo =
2*densidade_fluido_novo*fator_atrito_interno_fluido_novo[1]*pos_fluido_no
vo*velocidade_fluido_interno[1]**2/diametro_interno[1]
    perda_carga_choke_total =
2*densidade_fluido*fator_atrito_interno[0]*profundidade_linha_choke*veloc
idade_fluido_interno[0]**2/diametro_interno[0]
    perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito_anular[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]
    perda_carga_anular_poco_aberto_total =
2*densidade_fluido*fator_atrito_anular[2]*(profundidade -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
]


    pressao_fundo_2_circulacao = pressao_choke  +
hidrostatica[int(profundidade-1)] + perda_carga_choke_total +
perda_carga_anular_revestido_total + perda_carga_anular_poco_aberto_total


# Cálculo das pressões
    pressao_fundo_kick_vetor.append(pressao_fundo_2_circulacao)
    pressao_choke_kick_vetor.append(pressao_choke)
    pressao_bengala_kick_vetor.append(pressao_fundo_2_circulacao -
(hidrostatica_fluido_antigo + hidrostatica_fluido_novo) +
(perda_carga_coluna_fluido_antigo + perda_carga_coluna_fluido_novo))
    pressao_sapata_kick_vetor.append(pressao_choke +
hidrostatica_anular_sapata + perda_carga_choke_total +
perda_carga_anular_revestido_total)


    pressao_fundo_2_circulacao_vetor.append(pressao_fundo_2_circulacao)
    pressao_choke_total_2_circulacao_vetor.append(pressao_choke)
    pressao_bengala_2_circulacao_vetor.append(pressao_fundo_2_circulacao
- (hidrostatica_fluido_antigo + hidrostatica_fluido_novo) +
(perda_carga_coluna_fluido_antigo + perda_carga_coluna_fluido_novo))
    pressao_sapata_2_circulacao_vetor.append(pressao_choke +
hidrostatica_anular_sapata + perda_carga_choke_total +
perda_carga_anular_revestido_total)

# Controle do avanço do novo fluido
    pos_fluido_novo_vetor.append(pos_fluido_novo)
    pos_fluido_novo = pos_fluido_novo + avanco_fluido_novo
```

```
# 2ª fase da segunda circulação - Fluido pelo anular até preencher todo o
poço
# Manter pressão no bengala constante
pressao_choke_2_circulacao_vetor = []

if pos_fluido_novo > profundidade:
    pos_fluido_novo = profundidade

pressao_fundo_parametro = pressao_fundo_2_circulacao
while pos_fluido_novo > 0:

# Cálculo da hidrostatica
    hidrostatica_coluna = densidade_fluido_novo * gravidade *
profundidade
    hidrostatica_fluido_antigo_anular = densidade_fluido * gravidade *
pos_fluido_novo
    hidrostatica_fluido_novo_anular = densidade_fluido_novo * gravidade *
(profundidade - pos_fluido_novo)

    if pos_fluido_novo > profundidade_sapata:
        hidrostatica_sapata = (densidade_fluido * gravidade *
(pos_fluido_novo - profundidade_sapata)) + (densidade_fluido_novo *
gravidade * (profundidade - pos_fluido_novo))
    else:
        hidrostatica_sapata = densidade_fluido_novo * gravidade *
(profundidade - profundidade_sapata)

    if pos_fluido_novo > profundidade_sapata:
        hidrostatica_sapata_2 = (densidade_fluido * gravidade *
profundidade_sapata)
    else:
        hidrostatica_sapata_2 = (densidade_fluido * gravidade *
pos_fluido_novo) + (densidade_fluido_novo * gravidade *
(profundidade_sapata - pos_fluido_novo))

# Cálculo da perda de carga
    perda_carga_coluna_fluido_novo =
2*densidade_fluido_novo*fator_atrito_interno_fluido_novo[1]*profundidade*
velocidade_fluido_interno[1]**2/diametro_interno[1]

    if pos_fluido_novo > profundidade_linha_choke:
        perda_carga_choke_total =
2*densidade_fluido*fator_atrito_interno[0]*profundidade_linha_choke*veloc
idade_fluido_interno[0]**2/diametro_interno[0]
    else:
        perda_carga_choke_total =
(2*densidade_fluido*fator_atrito_interno[0]*pos_fluido_novo*velocidade_fl
uido_interno[0]**2/diametro_interno[0]) +
(2*densidade_fluido_novo*fator_atrito_interno_fluido_novo[0]*(profundidad
e_linha_choke-
pos_fluido_novo)*velocidade_fluido_interno[0]**2/diametro_interno[0])
```

```
    if pos_fluido_novo > profundidade_sapata:
        perda_carga_anular_revestido_total =
2*densidade_fluido*fator_atrito_anular[1]*(profundidade_sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]
    elif pos_fluido_novo <= profundidade_sapata and pos_fluido_novo >
profundidade_linha_choke:
        perda_carga_anular_revestido_total =
(2*densidade_fluido*fator_atrito_anular[1]*(pos_fluido_novo -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]) +
(2*densidade_fluido_novo*fator_atrito_anular_fluido_novo[1]*(profundidade
_sapata -
pos_fluido_novo)*velocidade_fluido_anular[1]**2/diametro_hidraulico[1])
    else:
        perda_carga_anular_revestido_total =
2*densidade_fluido_novo*fator_atrito_anular_fluido_novo[1]*(profundidade_
sapata -
profundidade_linha_choke)*velocidade_fluido_anular[1]**2/diametro_hidraul
ico[1]

    if pos_fluido_novo > profundidade_sapata:
        perda_carga_anular_poco_aberto_total =
(2*densidade_fluido*fator_atrito_anular[2]*(pos_fluido_novo -
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
]) +
(2*densidade_fluido_novo*fator_atrito_anular_fluido_novo[2]*(profundidade
- pos_fluido_novo)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2])
    else:
        perda_carga_anular_poco_aberto_total =
2*densidade_fluido_novo*fator_atrito_anular_fluido_novo[2]*(profundidade
-
profundidade_sapata)*velocidade_fluido_anular[2]**2/diametro_hidraulico[2
]


    pressao_fundo_2_circulacao = pressao_choke +
hidrostatica_fluido_antigo_anular + hidrostatica_fluido_novo_anular +
perda_carga_choke_total + perda_carga_anular_revestido_total +
perda_carga_anular_poco_aberto_total
    pressao_sapata = pressao_fundo_2_circulacao - hidrostatica_sapata -
perda_carga_anular_poco_aberto_total
    pressao_choke_minima = (pressao_poros -
hidrostatica_fluido_antigo_anular - hidrostatica_fluido_novo_anular)
    pressao_choke_maxima = (pressao_fratura - hidrostatica_sapata_2)
    pressao_choke = pressao_fundo_2_circulacao -
hidrostatica_fluido_antigo_anular - hidrostatica_fluido_novo_anular -
perda_carga_choke_total - perda_carga_anular_revestido_total -
perda_carga_anular_poco_aberto_total


    while abs(pressao_fundo_2_circulacao - pressao_fundo_parametro) >
valor_convergencia:
#    while abs(pressao_fratura - pressao_sapata) < valor_convergencia:
```

```
        step = pressao_fundo_2_circulacao - pressao_fundo_parametro
        pressao_choke = pressao_choke - step
        pressao_fundo_2_circulacao = pressao_choke +
hidrostatica_fluido_antigo_anular + hidrostatica_fluido_novo_anular +
perda_carga_choke_total + perda_carga_anular_revestido_total +
perda_carga_anular_poco_aberto_total
        pressao_sapata = pressao_fundo_2_circulacao - hidrostatica_sapata
- perda_carga_anular_poco_aberto_total




# Cálculo das pressões
    pressao_fundo_kick_vetor.append(pressao_fundo_2_circulacao)
    pressao_bengala_kick_vetor.append(pressao_fundo_2_circulacao -
hidrostatica_coluna + perda_carga_coluna_fluido_novo)
    pressao_choke_kick_vetor.append(pressao_choke)
#(pressao_fundo_calculado - hidrostatica_fluido_antigo_anular -
hidrostatica_fluido_novo_anular - perda_carga_choke_total -
perda_carga_anular_revestido_total -
perda_carga_anular_poco_aberto_total)
    pressao_sapata_kick_vetor.append(pressao_sapata)


    pressao_choke_minima_vetor.append(pressao_choke_minima)
    pressao_choke_maxima_vetor.append(pressao_choke_maxima)
    pressao_choke_2_circulacao_vetor.append(pressao_choke)


    pressao_fundo_2_circulacao_vetor.append(pressao_fundo_2_circulacao)
    pressao_bengala_2_circulacao_vetor.append(pressao_fundo_2_circulacao
- hidrostatica_coluna + perda_carga_coluna_fluido_novo)
    pressao_choke_total_2_circulacao_vetor.append(pressao_choke)
#(pressao_fundo_calculado - hidrostatica_fluido_antigo_anular -
hidrostatica_fluido_novo_anular - perda_carga_choke_total -
perda_carga_anular_revestido_total -
perda_carga_anular_poco_aberto_total)
    pressao_sapata_2_circulacao_vetor.append(pressao_sapata)
#    print pressao_fundo_calculado + 0.000145
#    print hidrostatica_coluna * 0.000145
#    print perda_carga_coluna_fluido_novo * 0.000145
#    raw_input()

# Controle do avanço do novo fluido
    pos_fluido_novo_vetor.append(pos_fluido_novo)
    pos_fluido_novo = pos_fluido_novo - avanco_fluido_novo




# Conversão das pressões de Pa para psia
```

```
pressao_fundo_kick_vetor_psia = array(pressao_fundo_kick_vetor)
pressao_fundo_kick_vetor_psia = pressao_fundo_kick_vetor_psia * 0.000145

pressao_bengala_kick_vetor_psia = array(pressao_bengala_kick_vetor)
pressao_bengala_kick_vetor_psia = pressao_bengala_kick_vetor_psia *
0.000145

pressao_choke_kick_vetor_psia = array(pressao_choke_kick_vetor)
pressao_choke_kick_vetor_psia = pressao_choke_kick_vetor_psia * 0.000145

pressao_sapata_kick_vetor_psia = array(pressao_sapata_kick_vetor)
pressao_sapata_kick_vetor_psia = pressao_sapata_kick_vetor_psia *
0.000145

pressao_choke_minima_vetor_psia = array(pressao_choke_minima_vetor)
pressao_choke_minima_vetor_psia = pressao_choke_minima_vetor_psia *
0.000145

pressao_choke_maxima_vetor_psia = array(pressao_choke_maxima_vetor)
pressao_choke_maxima_vetor_psia = pressao_choke_maxima_vetor_psia *
0.000145

pressao_choke_2_circulacao_vetor_psia =
array(pressao_choke_2_circulacao_vetor)
pressao_choke_2_circulacao_vetor_psia =
pressao_choke_2_circulacao_vetor_psia * 0.000145

pressao_fundo_2_circulacao_vetor_psia =
array(pressao_fundo_2_circulacao_vetor)
pressao_fundo_2_circulacao_vetor_psia =
pressao_fundo_2_circulacao_vetor_psia * 0.000145

pressao_bengala_2_circulacao_vetor_psia =
array(pressao_bengala_2_circulacao_vetor)
pressao_bengala_2_circulacao_vetor_psia =
pressao_bengala_2_circulacao_vetor_psia * 0.000145

pressao_choke_total_2_circulacao_vetor_psia =
array(pressao_choke_total_2_circulacao_vetor)
pressao_choke_total_2_circulacao_vetor_psia =
pressao_choke_total_2_circulacao_vetor_psia * 0.000145

pressao_sapata_2_circulacao_vetor_psia =
array(pressao_sapata_2_circulacao_vetor)
pressao_sapata_2_circulacao_vetor_psia =
pressao_sapata_2_circulacao_vetor_psia * 0.000145




##
pylab.plot(pressao_fundo_kick_vetor_psia,'b')
pylab.plot(pressao_bengala_kick_vetor_psia,'r')
pylab.plot(pressao_choke_kick_vetor_psia,'g')
```

```
pylab.plot(pressao_sapata_kick_vetor_psia,'k')
pylab.xlabel('Tempo [s]')
pylab.ylabel('Pressoes [psia]')
pylab.legend(('Pressao de fundo','Pressao no bengala','Pressao no
choke','Pressao na sapata'), loc=7)
pylab.grid()
pylab.show()
#
#
pylab.plot(pos_topo_vetor,'b')
pylab.plot(pos_base_vetor,'r')
pylab.xlabel('Tempo [s]')
pylab.ylabel('Profundidade [m]')
pylab.legend(('Topo do gas','Base do gas'), loc=0)
pylab.axis([0, 16000, 5000, -500])
pylab.grid()
pylab.show()
#
#
pylab.subplot(211)
pylab.plot(volume_vetor,'g')
pylab.xlabel('Tempo [s]')
pylab.ylabel('Volume [m3]')
pylab.grid()
pylab.subplot(212)
pylab.plot(tamanho_kick,'r')
pylab.xlabel('Tempo [s]')
pylab.ylabel('Comprimento do gas [m]')
pylab.grid()
pylab.show()
#
#
pylab.plot(pressao_fundo_2_circulacao_vetor_psia,'b')
pylab.plot(pressao_bengala_2_circulacao_vetor_psia,'r')
pylab.plot(pressao_choke_total_2_circulacao_vetor_psia,'g')
pylab.plot(pressao_sapata_2_circulacao_vetor_psia,'k')
pylab.xlabel('Tempo [s]')
pylab.ylabel('Pressoes [psia]')
pylab.legend(('Pressao de fundo','Pressao no bengala','Pressao no
choke','Pressao na sapata'), loc=7)
pylab.grid()
pylab.show()
#
#
pylab.plot(pressao_choke_maxima_vetor_psia, 'r')
pylab.plot(pressao_choke_minima_vetor_psia, 'b')
pylab.plot(pressao_choke_2_circulacao_vetor_psia, 'g')
pylab.xlabel('Tempo [s]')
pylab.ylabel('Pressoes [psia]')
pylab.legend(('Pressao maxima no choke', 'Pressao minima no choke',
'Pressao no choke'), loc=0)
pylab.grid()
pylab.show()
```

```
print "END OF SIMULATION"

"""
*********************************************************************
*****
*********************************************************************
*****

CASO DE MIGRAÇÃO DO GÁS COM O POÇO ABERTO COM CONTROLE DE POÇO VOLUME
VARIÁVEL -> DONE!


*********************************************************************
*****
*********************************************************************
*****
"""
```