Monografia de Graduação

Desenvolvimento de um Simulador Computacional para Poços de Petróleo com Método de Elevação Artificial por *Plunger Lift*

Cristiano Gurgel de Castro

Natal, dezembro de 2009

UFRN - CT - NUPEG - Campus Universitário - CEP: 59070-970 - Natal-RN - Brasil
Fone-Fax: (84) 3215 3773 - www.nupeg.ufrn.br - prhanp14@nupeg.ufrn.br

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

Desenvolvimento de um Simulador Computacional para Poços de Petróleo com Método de Elevação Artificial por *Plunger Lift*

Cristiano Gurgel de Castro

Natal – RN
Dezembro / 2009

Desenvolvimento de um Simulador Computacional para Poços de Petróleo com Método de Elevação Artificial por *Plunger Lift*

Cristiano Gurgel de Castro

Orientador: Prof. Dr. André Laurindo Maitelli

Trabalho de Conclusão de Curso apresentado ao Corpo docente do Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Natal – RN
Dezembro / 2009

Desenvolvimento de um Simulador Computacional para Poços de Petróleo com Método de Elevação Artificial por *Plunger Lift*

Cristiano Gurgel de Castro

Trabalho de Conclusão de Curso aprovado em 14 de dezembro de 2009 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. André Laurindo Maitelli (Orientador) ..... DCA/UFRN

Prof. Dr. Andrés Ortiz Salazar ..... DCA/UFRN

Prof[a]. Ma. Carla Wilza Souza de Paula Maitelli ..... UFRN

“As dificuldades crescem à medida que nos aproximamos do nosso objetivo.”

Johann Goethe

Dedico esse trabalho primeiramente a Deus, por me dar o dom da vida, guiar todos os meus passos, reeguer-me nas horas difíceis e brindar comigo as horas felizes; obrigado por tudo. Também dedico à meus pais pela confiança na minha capacidade e por toda a dedicação que têm oferecido a mim durante esses meus anos de vida. A meus irmãos pela cumplicidade e simplesmente por estarem presentes. Ao restante da família, pelos momentos de felicidades quando estamos reunidos. Aos meus amigos, pelo ombro que oferecem nos momentos de desânimo e por todos os momentos divertidos que temos vividos ao longo dos anos. Sem vocês, esse trabalho e nenhum outro teria razão de ser.

AGRADECIMENTOS

Ao Professor André Laurindo Maitelli pela orientação

Aos colegas de trabalho do Laboratório de Automação em Petróleo pela ajuda com as dúvidas e pelo ótimo ambiente de trabalho

Aos professores do Departamento de Engenharia de Computação e Automação

Ao Programa de Recursos Humanos da Agência Nacional de Petróleo PRH-14 pelo apoio financeiro

Ao Eng. Edson Bolonhini, pelo apoio e paciência para o entendimento do objeto de trabalho

Aos colegas de curso pela ajuda mútua nas (muitas) horas de estudos

RESUMO

O Petróleo é a fonte energética mais utilizada no mundo. A sua indústria talvez possua processos mais complexos do que qualquer outra; contando com diversas etapas como a prospecção, a perfuração, completação, elevação, transporte, refino e distribuição. Algumas empresas, principalmente as grandes produtoras multinacionais, estão envolvidas em todas essas etapas. Em cada etapa há o investimento de quantias enormes. É importante salientar que em algumas etapas a quantia aplicada pode não ter um retorno garantido, como, por exemplo, na etapa de perfuração de um poço pioneiro, em que não há a garantia de que petróleo (ou gás) será encontrado. Para tornar mais eficientes os processos em questão e também para minimizar os riscos, tem sido dada um importância cada vez mais à tecnologia e à automação.

O presente trabalho preocupa-se com a etapa de Elevação, uma das últimas do *upstream*. O grande desafio na Elevação é fazer com que o óleo (ou gás), que pode estar a centenas ou a milhares de metros na subsuperfície, se eleve até alcançar as facilidades de produção para, então, ser transportado para tratamento. A escolha de um método de elevação depende de vários fatores, cabendo, essa decisão, a um engenheiro responsável. Portanto, é importante estimar o comportamento do poço com um método de elevação.

Um dos métodos de elevação é o *Plunger Lift*. É um método pneumático que utiliza um pistão na coluna de produção como interface entre óleo e gás. O seu funcionamento é regido por uma válvula presente na linha de produção. Quando está fechada, ocorre um acúmulo de pressão no sistema, ao se abrir, essa pressão anteriormente acumulada, é liberada e empurra o pistão em direção da superfície, fazendo com que o óleo acima deste se eleve. Apesar de já ser utilizado com sucesso em diversos poços, os algoritmos de controle desenvolvidos ainda são incipientes.

O trabalho apresentado no presente relatório descreve a implementação de um modelo desenvolvido para o método *Plunger Lift* com a finalidade de prover um Simulador Computacional. A simulação computacional se tornou uma ferramenta quase imprescindível para a análise de sistemas complexos e, no nosso caso, deverá prover suporte ao desenvolvimento de algoritmos eficientes para o controle do processo, ao treinamento de pessoal e também a um melhor entendimento a respeito do sistema em questão.

Palavras-chave: Petróleo, Elevação Artificial, *Plunger Lift*, Simulação Computacional.

ABSTRACT

Oil is the most used energetic source all around the globe. Its industry maybe has more complex processes than any other, with several steps like Prospecting, Drilling, Completion, Lifting, Transport, Refining. Some companies, mainly giant multinationals producers, are involved in all of these steps. In each step, there are investments of huge amounts. It is important to remember that in some steps, the amount invested maybe not results in a guaranteed return, as, for example, happens in the drilling of the first well of a field, where we are not sure if there is oil or gas underneath surface. To improve efficiency of the process and to avoid risks, it has been given more importance day by day, to technology and automation.

The work presented here discuss about the Lifting, one of the upstream last steps. The great challenge of Lifting is how to make oil (or natural gas) rise up from hundreds or thousands of meters below surface to the production line, to be transported for later treatment. The choice of a lift method depends on several points and the decision is made by a responsible engineer. Therefore, it is important to estimate the behavior of an oil well with one or another Lift Method.

One of these Artificial Lift Methods is Plunger Lift. It is a pneumatic method which uses a plunger as interface between oil and gas. Its behavior is managed by a valve located in the beginning of the production line. When it is closed, there is a accumulation of pressure in the system; and when its opened, the previously accumulated pressure is freed and pushes the plunger to surface, making the oil above it also rise. Besides being used with success in many oil wells, the developed control algorithms for the process are incipient.

The work presented in this report relates the implementation of a model developed for the Plunger Lift, intending to provide a Computational Simulator. Simulators became an almost indispensable tool for complex systems analysis and, in this case, must provide support to the development of efficient control algorithms, to people training and also for a better understanding of the system.

Key-words: Oil, Artificial Lift, Plunger Lift, Simulation

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE SÍMBOLOS E ABREVIATURAS

BSW	*Basic Sediments and Water*, quociente entre a vazão
CLP	Controlador Lógico Programável
grau API	Escala idealizada pelo *American Petroleum Institute* - API, para medir a densidade relativa de líquidos
GUI	*Graphical User Interface*, Interface Gráfica do Usuário
IPR	*Inflow Performance Relationship*. Ferramenta matemática para determinar a produção do reservatório de acordo com a pressão de fluxo de fundo
PETROBRAS	Petróleo Brasileiro S/A
PL	*Plunger Lift*
RGL	Razão Gás-Líquido
RGO	Razão Gás-Óleo
UN-RNCE	Unidade de Negócios do Rio Grande do Norte e do Ceará

CAPÍTULO 1

INTRODUÇÃO

O Petróleo (do latim *petro*: pedra + *oleum*: óleo) é a fonte energética mais utilizada no mundo e sua utilização retoma a tempos antigos. O fluido, que era encontrado em exsudações na superfícies, foi utilizado por várias civilizações como os egípcios, que utilizavam o betume para pavimentar estradas, ou os fenícios, que utilizavam o fluido para calafetar as embarcações, os romanos, por exemplo, também o utilizava para fins bélicos. A sua exploração moderna, porém, iniciou-se em meados do século XIX, principalmente após 1859, quando Edwin Laurentine Drake, também conhecido como Coronel Drake, perfurou o primeiro poço utilizando técnicas de perfuração de minas de sal.

O produto foi utilizado primeiramente na indústria da iluminação, já que o querosene estava substituindo o óleo de baleia anteriormente utilizado. Posteriormente, deu-se um dos fatos mais marcantes que favoreceu o crescimento do consumo desse combustível fóssil: a invenção dos motores a gasolina. A gasolina era considerada anteriormente um rejeito descartado no refino. É interessante notar que até os dias atuais ainda não há o aproveitamento de todos os produtos resultantes do refino do óleo, podemos estar desperdiçando produtos preciosos como a gasolina o foi no passado. No início do século XX, a demanda cresceu vertiginosamente devido a popularização do automóvel, graças a figuras como Henry Ford, o inventor da "linha de montagem" (Figura 1.1). Nesse mesmo tempo imensas jazidas foram descobertas nos EUA (Texas). No decorrer do século passado, o produto tornou-se muito estratégico; na Segunda Grande Guerra, por exemplo, a mobilidade dos exércitos era totalmente dependente desse combustível. Logo, começaram os conflitos motivados por essa riqueza em muitas partes do mundo, tendo o Oriente Médio como uma região bastante representativa desse fato, tanto pelo estoque de óleo na região quanto pelo violência dos conflitos. No Brasil, as pesquisas de petróleo iniciaram verdadeiramente com a criação, em 1953, da Petrobras no governo de Getúlio Vargas [Thomas 2001]. Atualmente, somos um país com relativa importância no cenário internacional e dominamos especialmente a tecnologia de exploração *offshore*.

Grande parte da exploração de petróleo está nas mãos de grandes produtoras multinacionais, dentre as quais podemos citar *Shell*, *Petronas*, a brasileira *Petrobras* ou a

Figura 1.1: Linha de Montagem

Saudi Aramco (maior produtora mundial), porém nesse mercado há também espaço para produtoras independentes menores. Desde o século XIX, as empresas produtoras de óleo obtêm um destaque mundial. Podemos citar o exemplo da empresa *Standard Oil*, a primeira companhia petrolífera americana e a maior empresa de seu tempo, até ser dividida (pela suprema corte dos EUA) em 34 companhias menores, em uma primeiras ações antitruste da história. Entre essas empresas, emergiriam a *Exxon*, *Chevron*, *Atlantic*, *Mobil* e a *Amoco*. Com tantas empresas poderosas no mercado, o jogo político na área do petróleo torna-se bastante complexo e a busca de novas tecnologias na produção torna-se imprescindível para uma competição aberta.

O petróleo é formado basicamente de hidrocarbonetos. Uma das teorias aceitas da formação do, assim chamado, Ouro Negro é a que ele se origina a partir da matéria orgânica depositada no fundo de antigos mares e lagos que foi sobreposta por camadas de sedimentos ao longo de alguns milhões de anos. Em um processo que alia elevadas temperaturas, elevadas pressões, ações de bactérias, entre outros fatores conhecidos ou não, ele aparece na chamada rocha geradora ou rocha matriz. Posteriormente, buscando sempre pontos de menor pressão, o óleo migra através de um processo ainda alvo de discussões da rocha geradora para outra chamada rocha reservatório, ou simplesmente Reservatório, e, caso ocorram armadilhas[1]. A Figura 1.2 apresenta um tipo de armadilha chamada de anticlinal, o fluido fica aí acumulado em porosidades e fissuras presentes na rocha. Pelas razões apresentadas, o fluido somente pode ser encontrado em bacias sedimentares.

Nos reservatórios, ocorre o acúmulo de líquido em fissuras e/ou poros presentes no mesmo. Caso esses poros estejam interconectados, o óleo neles contido escoa dependendo do diferencial de pressão entre dois pontos e, caso um poço seja construído no

[1] Diz-se armadilha uma configuração geológica que impede o óleo de continuar o caminho da migração

local, pode, portanto, ser extraído. Dado o seu grande consumo nos dias atuais e por ser uma fonte de energia não renovável, algum dia o petróleo irá se extinguir, muito embora descobertas de novos campos ainda aconteçam, como a descoberta do campo de Tupi[2] aqui no Brasil. Estimativas apontam que, na situação em que não haja descoberta de jazidas, em apenas 15 anos, poucos países terão reservas suficientes para manter a exportação. Dessa forma, grande parte das pesquisas na área se preocupa em encontrar maneiras de aperfeiçoar as várias etapas da produção, refino e transporte do óleo até os centros de consumo. A busca para minimizar desperdícios e maximizar a produção está em contínua ascenção.

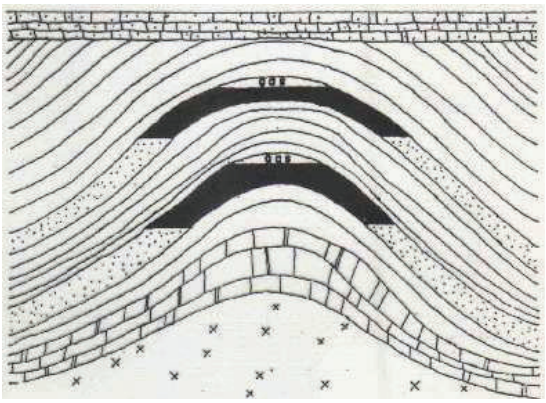


Figura 1.2: Armadilha. Fonte: [Vidal 2008]

A exploração comercial dessa fonte de energia é complexa. O primeiro passo é explorar áreas de bacias sedimentares. Através de um dispendioso estudo através de dados geológicos e geofísicos captados pela área, procura-se indícios da ocorrência de petróleo nesses locais e, caso se tenha evidências de um local com grande probabilidade de ocorrência dos fluidos propõe-se a perfuração de um poço no local: a parte que mais demanda investimento. Essa fase, chamada de Prospecção, foi alvo de uma evolução constante desde quando o petróleo começou a ser explorado comercialmente; nessa época a descoberta de um campo era muito dependente do fator sorte; hoje através da fotografia aérea por aviões ou satélites e da realização de testes sismológicos (na Figura 1.3 é mostrado a aplicação da Sísmica de Reflexão, as vibrações causadas pelo caminhão são refletidas pelas interfaces entre as camadas geológicas e são captadas pelos geofones instalados na superfície) e também pelos recursos computacionais utilizados na interpretação e apresentação dos dados obtidos, consegue-se obter uma melhor proporção nos "acertos" de áreas com petróleo. Após a Perfuração e Completação do poço pioneiro pode-se, efetivamente, comprovar a existência ou a não-existência de uma bacia petrolífera no local. Caso se verifique a existência do fluido, procede-se uma da Avaliação de Formação, e caso se verifique que o campo é comercialmente explorável, ou em outras palavras, que o retorno obtido com o óleo recuperável do campo seja maior que o investimento a ser feito, mais poços de produção são construídos, dependendo da extensão

[2] As reservas do campo de Tupi são estimadas entre 5 e 8 bilhões de barris, ou seja, algo em torno de um trilhão de litros

do campo, estimada a partir dos estudos prévios. Todos esses processos antecedem a efetiva exploração comercial e são (bastante) dispendiosos. Há ainda outros processos que antecedem o consumo: Transporte, Refino, Distribuição etc. Todas essas etapas envolvem imensos desafios científicos, tecnológicos, logísticos entre outros, e pode-se dizer que a indústria petrolífera detém (talvez como nenhuma outra) os processos industriais mais variados e complexos.

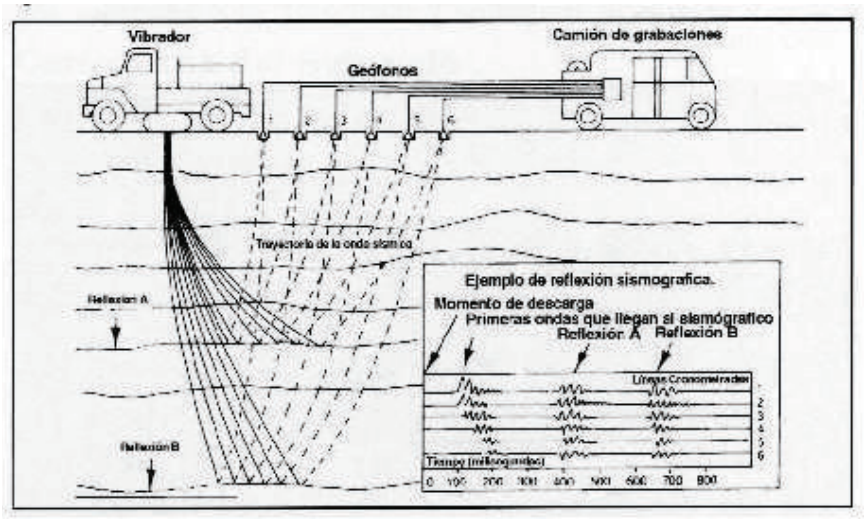


Figura 1.3: Aplicação do Método Sísmico de Reflexão

O trabalho em questão irá tratar especificamente da etapa que ocorre após a perfuração e completação do poço. Nessa etapa, a tarefa a ser resolvida do processo de produção é o de como fazer o óleo (ou gás), que pode estar localizado a centenas ou até milhares de metros abaixo da superfície, se elevar através da coluna de produção até a superfície, vencendo, desse modo, forças direcionadas para o centro da terra, como o atrito decorrente do contato do fluido com as paredes do *tubing* ou também o peso do fluido em ascensão. A etapa da produção onde essa tarefa é cumprida é chamada de Elevação. Existem dois métodos básicos de elevação: a natural e a artificial. A elevação natura é mais simples e demanda menos custos com manutenção. A elevação artificial se torna necessária quando a elevação natural não pode, ou não convém, ser aplicada. Nesse caso,é preciso recorrer a um dos vários métodos de elevação artificial. A etapa de elevação será mais adequadamente detalhada no capítulo posterior.

Uma escolha de um método de elevação artificial depende de vários fatores, como a disponibilidade de energia, experiência do pessoal envolvido, grau API do óleo e BSW do fluido, e visa à economia na produção. O método PL, por exemplo, é utilizado para poços produtores com baixa pressão estática e poços com alta RGO, entre outros fatores. As primeiras instalações de *Plunger Lift* que se tem notícia no Brasil ocorreram na década de 60 e na UN-RNCE, o primeiro conjunto foi instalado em 1994, no campo de Livramento (LV-13). Buscando projetar algoritmos eficientes de controle para maximizar a produção desses poços e também obter um melhor entendimento/modelagem desse método de elevação artificial, a Petrobras tem investido em projetos de parceria com a Universidade

Federal do Rio Grande do Norte. Há, por exemplo, o projeto PETRELEV, com objetivo principal de desenvolver um modelo dinâmico mais preciso para poços de petróleo com esse tipo de elevação artificial, para esse projeto há também a proposta de uma Planta Piloto Para Poços com *Plunger Lift* a ser construída nas dependências do Campus Universitário de Natal da UFRN. No projeto AUTOPOC, tem-se, entre outros, o objetivo de elaboração de um Simulador de Poço de Petróleo com método de Elevação Artificial por PL, desenvolvimento esse que será abordado no presente relatório. É evidente a necessidade de integração entre os projetos, já que no simulador a ser desenvolvido será utilizado o modelo dinâmico obtido a partir de observações da planta piloto, o que levará a um software de simulação mais preciso. Esse simulador deverá ser posteriormente ser utilizado para o projeto e testes dos novos algoritmos de controle.

No capítulo 2, é discutido a respeito da literatura utilizada para o aprendizado do método e também para obter conhecimentos sobre simulação computacional. Após no Capítulo 3, aborda-se a ferramenta construída no projeto utilizada na execução do projeto. Os resultados obtidos são mostrados no Capítulo 4. E no Capítulo 5, finalmente são apresentas as conclusões.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Para o projeto e desenvolvimento de um simulador é preciso obter certo domínio sobre o processo (sistema) abordado. Portanto, desde as primeiras etapas do projeto esforços foram concentrados no estudo sobre o método PL, seu princípio de funcionamento e, mais especificamente, seu modelo dinâmico. Além disso, foram estudados aspectos relacionados à indústria de petróleo, desde formação e composição química do fluido, até prospecção, perfuração de poços e métodos de elevação. Em outra vertente, foram conduzidos estudos sobre simulação computacional, para se saber as diretrizes de um bom projeto. Mais recentemente, alguns meses após o começo da implementação do método, pesquisas e estudos sobre as características/modo de operação da linguagem de programação C++ se tornaram fundamentais. Esses foram basicamente os tópicos de estudo abordados durante o projeto.

Na Seção 2.1 é aprofundada um pouco mais a discussão sobre as etapas de produção do petróleo até a etapa de elevaçao que será abordada no projeto. na Seção 2.2 explica-se o funcionamento do método de Elevaçao Artificial *Plunger Lift*. Finalmente, na Seção 2.3 é abordado o assunto Simulação Computacional.

2.1 Uma visão geral do Petróleo

No capítulo anterior foram apresentadas várias etapas necessárias para que a exploração do petróleo seja possível. Nas próximas subseções, essas etapas são discutidas mais profundamente, com ênfase na etapa de elevação.

2.1.1 Prospecção

Para que se de possa extrair comercialmente óleo na subsuperfície uma gama de condições singulares devem ser satisfeitas, são elas: a existência de uma rocha matriz onde o petróleo será criado; a migração primária, isto é a saída do fluida da rocha matriz

buscando pontos de menor pressão; a existência de uma rocha reservatório que permita o armazenamento do óleo nos seus poros; armadilhas que impedem o óleo de continuar seu processo de migração em direção à superfície (na qual apareceria com as exsudações); a migração secundária, que nada mais é do que o caminho do fluido pela rocha reservatório: entre outras condições. Todos esses fatores fazem com que a ocorrência de óleo na subsuperfície seja algo raro e probabilidade de que um poço perfurado "ao acaso" encontre fluido é muito baixa. Portanto, estudos prévios de uma área na qual deseja-se procurar petróleo são indispensáveis. Esses estudos são feitos na etapa chamada de Prospecção.

Nessa etapa, há um exaustivo estudo geológico e geofísico da área em questão para se determinar se é alta a probabilidade de óleo (e/ou gás) estar armazenado no subsolo, nas Rochas Reservatório [Thomas 2001]. Nesses estudos utilizam-se várias técnicas como análise de relatórios de perfurações de poços pela região; fotografias aéreas, levantamento e processamento de dados sísmicos (a Figura 2.1 mostra um exemplo de dados obtidos através desses estudos), etc. e tem-se a finalidade de testar se a área contém fatores essenciais para o armazenamento de óleo (ou gás). Somente após esses estudos propõe-se a perfuração de um poço no local mais provável de acúmulo do fluido.

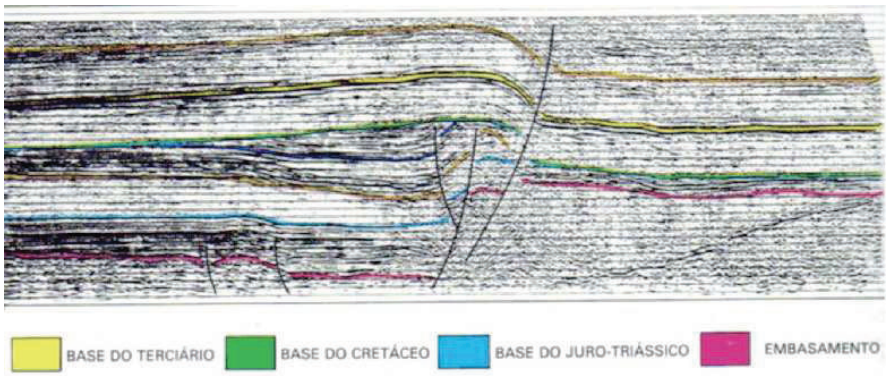


Figura 2.1: Exemplo de dados obtidos através de sísmica. Fonte: [Vidal 2008]

A rocha reservatório é um meio poroso com características únicas de armazenamento e de fluxo [Vidal 2005]. Configura-se através das propriedades básicas da rocha e dos fluidos nela presentes. Entre as características da rocha podemos citar a sua porosidade, razão entre o volume poroso da rocha e seu volume total, sua permeabilidade, que define a facilidade de um fluido escoar através de seus poros ou a saturação de um determinado fluido (óleo, gás ou água), que é a relação entre o volume do fluido em questão pelo volume poroso total da rocha [Thomas 2001]. Outra característica do reservatório bastante importante é a Razão Gás-Óleo, definida como a razão entre sua vazão de gás e de óleo produzidos pelo reservatório, medidas nas condições de superfície. Outra característica do fluido produzido pelo reservatório é o BSW, o quociente entre a vazão de água e sedimentos que estão sendo produzidos e a vazão total do líquido. [Thomas 2001].

2.1.2 Perfuração e Completação

A Perfuração é a etapa que mais demanda investimento. É feita em várias fases e cada uma delas é caracterizada pelo diâmetro da broca utilizada. Ao final de cada fase de perfuração, um anel de revestimento é posto no poço e é também aplicado um cimento especial para fixá-lo ao solo lateral. Essa cimentação também é útil para isolar diferentes zonas de produção do reservatório, impedindo o fluxo de fluidos entre diferentes zonas pelo espaço que se formaria entre o revestimento e o solo. O primeiro revestimento a ser colocado em um poço em perfuração também serve de suporte para a instalação dos equipamentos de segurança na superfície (os chamados BOPs). As brocas utilizadas na perfuração podem ser de vários tipos dependendo do tipo de rocha perfurada, elas trabalham juntamente com o fluido de perfuração, que remove os cascalhos provenientes da perfuração e também promove o resfriamento do equipamento, entre outras funções.

Hoje em dia, com a tecnologia utilizada na perfuração de poços, é possível ainda a perfuração de poços chamados de direcionais, isto é, poços em que há desvios no subsolo de forma a atingir áreas que não aquelas exatamente abaixo da cabeça do poço. Essa tecnologia de poços direcionais permite, por exemplo, aumentar o contato entre o poço perfurado e a área produtora do reservatório, entre outras vantagens. A Figura 2.2 apresenta uma visão de alguns equipamentos utilizados na perfuração, o mais "visível" é o mastro, que serve de suporte aos comandos e à broca.

Figura 2.2: Mastro de um Poço

A Completação trabalha juntamente com a fase de Perfuração, tendo o objetivo de preparar o poço, deixando-o em condições de operação ao longo de toda a sua vida

produtiva. Nesta fase, são efetuadas operações destinadas a colocar o poço para produzir óleo ou gás. Na Figura 2.3 são mostradas as principais características de sua estrutura.

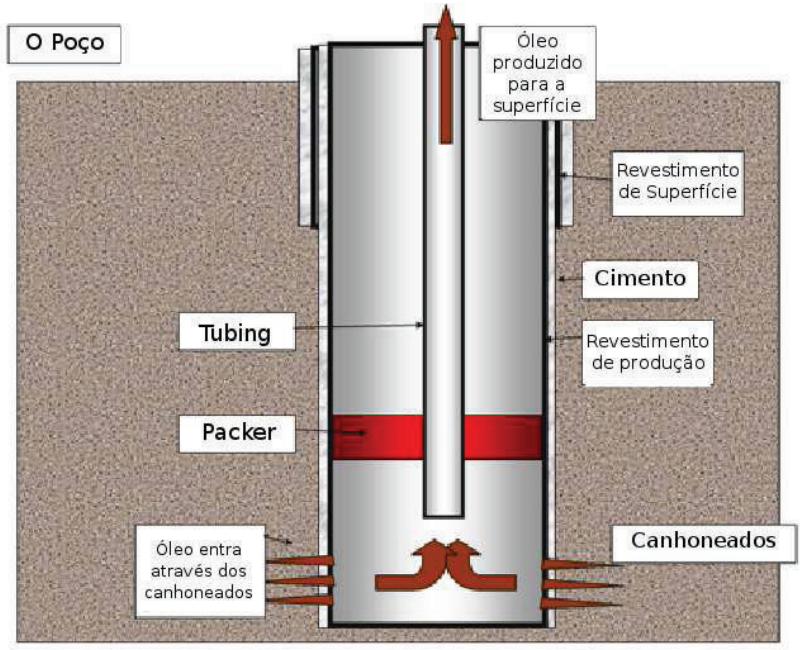


Figura 2.3: Um Esquema de Poço de Petróleo

O poço é constituído de uma coluna de produção (*tubing*), por onde os fluidos emergem até a linha de produção, localizada na superfície; o anular (*casing*) envolve a coluna de produção. Em uma determinada profundidade do *casing* há os chamados os canhoneados (*perforations*), que são perfurações no revestimento feitas geralmente por explosões controladas por onde os fluidos provenientes do reservatório penetram no poço. O *packer* é opcional e tem a função básica de promover a vedação do espaço anular entre o revestimento do poço e a coluna de produção, numa determinada profundidade [Thomas 2001]. A utilização do *packer* diminui o espaço do anular para o acúmulo de gás proveniente da produção do reservatório.

2.1.3 Elevação

Além do reservatório e do poço, outros sistemas fazem parte da produção de petróleo e estão intimamente conectados. Entre o equipamentos que há na cabeça do poço podemos ter as válvulas para o controle da produção, sensores (de pressão), a linha de produção. Todo esse aparato dá suporte à produção de Petróleo. Podemos ter além disso outros equipamentos que dão suporte à Elevação, ou seja, ao transporte do fluido do reservatório até a superfície através da coluna de produção. O trabalho apresentado nesse documento aborda essa etapa do *upstream*. Existem dois tipos básicos de elevação: natural e artificial.

A elevação natural geralmente acontece no início da produção do poço, quando a pressão exercida pelo reservatório é suficientemente grande para vencer as perdas de carga pelo peso do líquido, peso do gás, atrito do gás e do líquido na coluna de produção e demais forças direcionadas para o centro da Terra que atuem sobre o fluido em ascensão. Os poços que produzem por esse método são chamados de Poços Surgentes. Segundo [Thomas 2001], os poços surgentes produzem com os menores problemas de operação devido à simplicidade dos equipamentos de superfície e sub-superfície, com maiores vazões de líquido e com menor custo por unidade de volume.

A elevação artificial é utilizada caso a energia do reservatório não seja suficiente para manter o poço produzindo por surgência, ou ainda, apesar de o poço ser surgente, o fluxo obtido na superfície é baixo e pouco vantajoso comercialmente. Nesse caso, é preciso (ou preferível) fornecer uma energia extra para suprir as quedas de pressão entre o reservatório e as facilidades de produção. A escolha de um ou outro método de elevação depende de vários e diferentes fatores como o número de poços, a razão gás óleo (RGO) da produção do reservatório, a disponibilidade de energia, o pessoal treinado, e outros. Entre os principais métodos de elevação podemos destacar métodos essencialmente mecânicos, o Bombeio Mecânico [do Nascimento 2005], o mais utilizado (Figura 2.4), o Bombeio Centrífugo Submerso e o Bombeio por Cavidades Progressivas [Vidal 2005], e métodos pneumáticos como o *Gas Lift* contínuo e intermitente, e o *Plunger Lift* [Bolonhini 2005], o alvo desse projeto.

Figura 2.4: Unidade de Bombeio Mecânico

2.2 O método de Elevação *Plunger Lift*

O *Plunger Lift* é um método intermitente de elevação artificial, ou seja, o petróleo não é produzido de maneira contínua, mas através de golfadas cíclicas de óleo através da coluna de produção. [Bolonhini 2005] mostra três tipos de instalação de PL: PL com *Gas Lift* intermitente; PL com *packer* e PL sem *packer*, também chamado de PL convencional. As principais aplicações do PL são em poços produtores com reservatórios de baixa pressão estática e/ou pequena transmissibilidade; poços de óleo com alta razão gás-óleo, requer uma RGO mínima normalmente maior que $70\ m^3/m^3$ para cada 300 m; poços que operam com *Gas Lift* intermitente para a redução do escorregamento (*Fallback*) do líquido, que se trata do retorno do líquido ao fundo da coluna de produção após ter começado a se elevar, e/ou a redução da RGL de injeção; poços de gás para remoção de condensado ou água; poços de óleo ou gás para prevenção de deposição de parafina e/ou incrustações (*scale*) na coluna de produção. O alvo principal do estudo é o PL convencional, cuja instalação típica é mostrada na Figura 2.5, muito embora as outras variantes do método já estejam sendo estudadas.

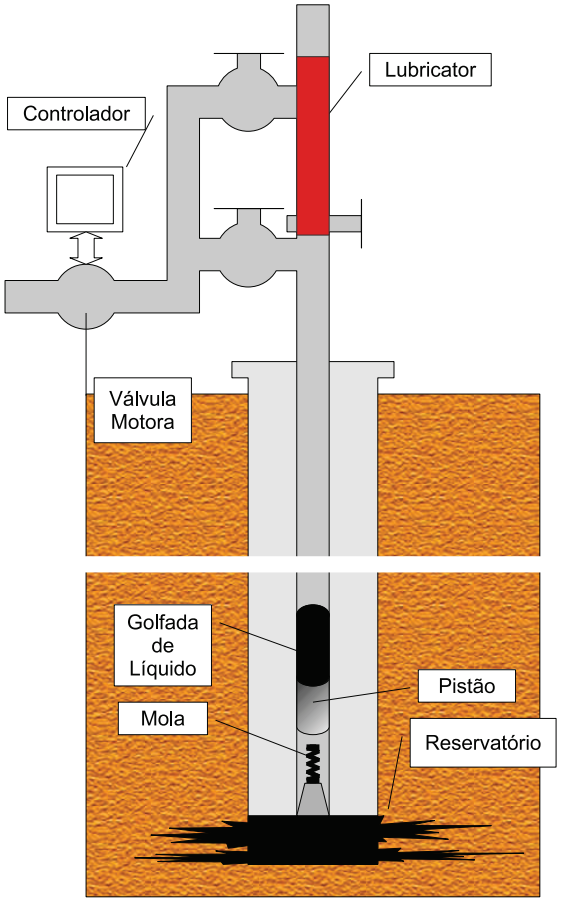
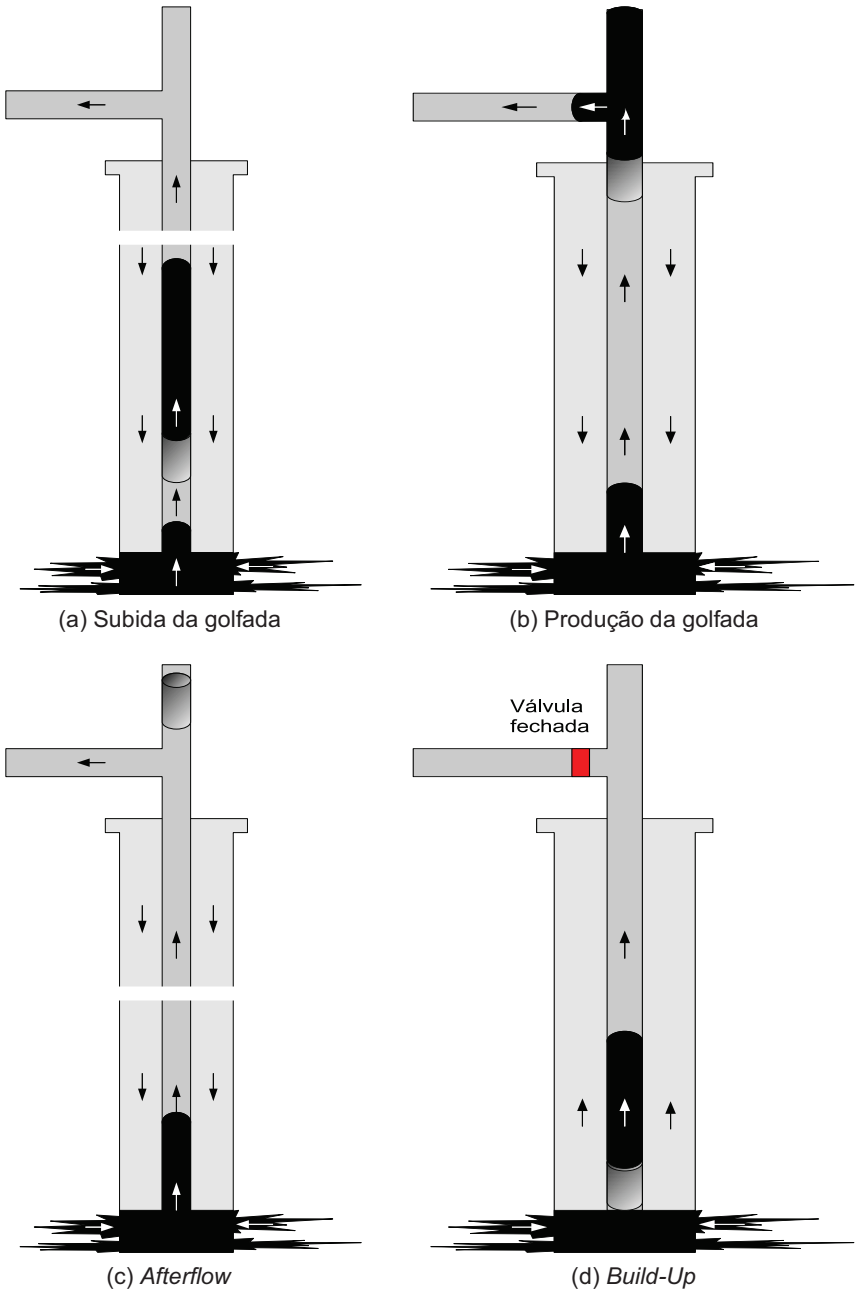


Figura 2.5: Esquema de um poço convencional de *Plunger Lift*

A ideia para o funcionamento do método é bastante simples, sendo baseada na expan-

são de um gás anteriormente pressurizado. É dividido em etapas [Baruzzi 1994]: subida do pistão, subdividida em subida e produção da golfada; pós-fluxo (*Afterflow*), caracterizado pela produção de gás após o pistão chegar à superfície e crescimento da pressão no poço (*Build-Up*) (ver Figura 2.6). A queda do pistão acontece no começo da etapa de *Build-Up*, esse processo deve ser alvo de estudo ainda, já que o modelo não trata esse movimento descendente do pistão.



(a) Subida da golfada (b) Produção da golfada

(c) *Afterflow* (d) *Build-Up*

Figura 2.6: Etapas do *plunger lift* convencional

Segundo [Baruzzi 1994], o papel primário do pistão é garantir que todo líquido acima dele alcance a linha de produção, ou seja, evitar o retorno (*fallback*) de líquido. Mas em poços onde há deposição de parafinas, incrustações ou hidratos, o pistão atua como um limpador das paredes do poço, o que poupa imensos custos com a limpeza da coluna de produção. Para modificar a razão gás-oleo dos fluidos do reservatório, em busca da RGL ótima para sua produção, o poço ainda pode contar com uma válvula controlada no topo do anular, para injeção ou dreno de gás. O PL assim modificado é chamado de Assistido (com injeção de gás ou com dreno), em oposição ao PL Autônomo, que opera apenas com a energia do reservatório. O presente trabalho trata apenas deste último tipo.

2.2.1 Funcionamento

Sabe-se que o *Plunger Lift* é um método intermitente, ele opera em ciclos e cada um deles contem as etapas citadas anteriormente. Quando a linha de produção está fechada e o pistão está no fundo da coluna de produção, o gás produzido pelo reservatório se acumula no *casing*. Com o acúmulo desse gás, a pressão sobe de forma contínua. Uma parte do líquido produzido pelo reservatório se acumula na coluna de produção acima do pistão (de fato, no modelo utilizado, é considerado que todo o líquido produzido pelo reservatório se acumula no *tubing*, como será apresentado posteriormente), formando uma coluna de líquido a ser produzida. Essa etapa, onde ocorre o acúmulo de pressões é chamada de *Build-Up* (ver Figura 2.6d).

Após, a comunicação com a linha de produção é liberada, ou seja, a *motor valve* é aberta, então ocorre uma brusca queda na pressão no topo da coluna de produção. Essa queda de pressão é transmitida até o gás acumulado abaixo do pistão. Esse gás rapidamente tende a expandir e a escapar pela coluna de produção, que possui o pistão no seu interior funcionando como o obstáculo para o gás. A energia acumulada do gás pressurizado do anular é transmitida ao pistão, que adquire energia cinética e eleva o óleo acumulado por sobre ele durante a etapa anterior, vencendo forças contrárias como o peso do pistão e da coluna de líquido, as forças de atrito, etc. Esse comportamento é similar a uma arma de ar comprimido logo após ser disparada. Configura-se, então, a etapa de subida do pistão. Enquanto o topo da golfada de líquido não alcança a linha de produção, ocorre a subida da golfada (Figura 2.6a), após temos a produção da golfada (Figura 2.6b).

Após toda a coluna de líquido ter sido produzida, a força do gás abaixo do pistão eleva o mesmo até o Lubrificador (*Lubricator*), onde o pistão é amortecido por uma mola (o algoritmo de controle deve ser cuidadoso o suficiente de modo que a velocidade de chegada do pistão não ser muito alta para evitar danos a essa estrutura) e então, o gás abaixo do pistão começa a ser, também, produzido. Assim sendo, o gás permanece em produção enquanto possuir a pressão necessária para vencer a força peso do pistão e mantê-lo no *Lubricator*. Essa etapa é chamada de *Afterflow*(ver Figura 2.6c).

O fim do *Afterflow* é determinado pelo fechamento da *Motor Valve*. O gás para de fluir pela coluna de produção e não há mais pressão suficiente forçando o pistão a se manter no topo da coluna de produção. Assim começa o processo de queda do pistão pela coluna de produção, primeiramente na coluna de gás e depois na coluna de líquido

que foi se formou no fundo do *tubing* devido a produção do reservatório durante as etapas anteriores. O pistão é então também amortecido por uma mola posicionada no fundo da coluna e um novo ciclo tem início.

Em alguns ciclos, a pressão (energia) acumulada na etapa de *Build-Up* não é suficiente para elevar o pistão e a coluna de líquido à superfície, nesses ciclos não há as etapas de produção de líquido e de *afterflow*. Se essa situação persiste indefinidamente, é dito que o poço afogou. Os algoritmos de controle devem ser desenvolvidos para que esse tipo de situação seja evitada, no caso do PL convencional, o seu único atuador é a *motor valve*, no caso de PL assistido, o algoritmo também pode controlar a vazão de gás injetado ou drenado do anular. Os sensores de pressão no topo do *tubing* no topo do *casing* bem como o sensor de magnético de chegada do pistão na superfície. Uma ideia da posição dos sensores no poço de PL pode ser obtida na Figura 2.7

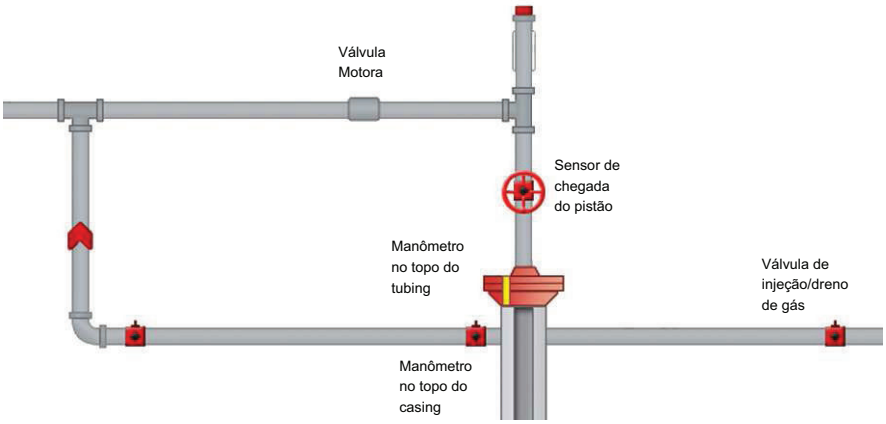


Figura 2.7: Representação da Cabeça de um Poço com PL

2.2.2 Modelagens

Vários modelos foram desenvolvidos para o método de elevação artificial *Plunger Lift*. [L. Marcano 1994] apresentam uma revisão da literatura e apontam que todos os modelos publicados até então se preocupavam apenas com o estágio de subida do pistão, deixando de lado todos os outros aspectos físicos relevantes do ciclo de produção: acúmulo de fluidos do reservatório durante o estágio de subida do pistão, etapa de acúmulo de pressão, formação da coluna de líquido. Seu modelo desenvolvido propunha, então, a tratar esses outros estágios, incluindo a influência do reservatório e o *Fallback*. Dividia então o processo em três etapas básicas:

Subida do pistão: começa quando a linha de produção é aberta. Subdividida em descida do gás do anular, ascensão do gás, golfada na coluna de produção e produção da golfada.

Queda do pistão: o pistão rapidamente se acelera e atinge uma velocidade constante.

Acúmulo de pressão: o pistão está no fundo da coluna de produção, líquido se acumula aumentando o tamanho da golfada e o gás produzido pressuriza o anular.

Também relatam que um programa de computador, desenvolvido em *Fortran*, executa o modelo e que foram obtidos resultados que concordam qualitativamente com o comportamento de poços de *Plunger Lift* convencional.

Já [J. Chacin 1994] propõem uma modelagem matemática para o tipo de *Plunger Lift* com *Gas Lift* intermitente. Relataram o fato de que esse tipo de PL, embora pareça bastante atrativo, tem recebido pouca atenção da literatura publicada. Fizeram também uma revisão da literatura e divide o processo nas etapas de:

- Subida do pistão;
- Produção da golfada;
- Produção de gás;
- Regeneração da golfada/acúmulo de pressão do gás.

Também trataram o *fallback* do líquido e fala-se do objetivo de se obter critérios para descobrir a produção ótima desses poços. Por final, Também é relatado o desenvolvimento de um programa de computador para teste do modelo que concordou qualitativamente com o comportamento observado nos poços reais.

Em [Baruzzi 1994] e [Baruzzi J.O.A. 1995] há a proposta de uma novo modelo hidrodinâmico para o PL convencional. Esse modelo considera a produção contínua do reservatório de acordo com a curva IPR do mesmo. Para o desenvolvimento do modelo, o ciclo do PL foi dividido em três etapas (rever Figura 2.6):

- Subida do Pistão, subdividida em:
  - Subida da golfada;
  - Produção da golfada;
- Produção de gás após o pistão chegar à superfície (*Afterflow*);
- Crescimento de pressão no poço (*Build-Up*).

Cada etapa tem seu próprio grupo de equações diferenciais regentes (equações de balanço e de fechamento). Após a definição dessas equações, ele dá os passos para a construção do programa de computador para implementar o modelo, definindo quantitativamente inclusive os passos de tempo para a obtenção de uma convergência e buscando a eficiência computacional.

Algumas simplificações presentes no modelo são citadas. O modelo considera que o líquido produzido pelo reservatório se acumula apenas na coluna de produção, porém o autor apresenta um método para verificar se isto acontece em um poço real, através da definição de uma RGL mínima para termos esse caso. Considera que o pistão é estanque, isto é, não há *fallback* de líquido. Outra simplificação que pode ser apontada no modelo é que ele não abrange a etapa de queda do pistão, isto é, após da etapa de *Afterflow* quando o pistão está no topo, a passagem se dá diretamente para a etapa de *Build-Up* com o pistão na superfície.

Apesar dessas simplificações, os relatos dos resultados obtidos com o programa de computador apontam resultados satisfatórios. Esse último modelo guia o projeto do simulador do qual trata o presente relatório. A partir dele também pretendemos retirar simplificações ou desenvolver abordagens para o problema futuramente.

2.3 Simulação Computacional

Outro ramo de estudo foi o de Simulação Computacional. As diretrizes de um bom projeto de simulação bem como exemplos de simuladores foram abordados. Esses estudos aconteceram no começo do projeto e foram importantes para orientação nos passos a serem efetuados.

Segundo [Shannon 1998], uma simulação é o processo de criação de um modelo de um sistema real e de condução de experimentos com esse modelo com o propósito de entender o comportamento do sistema e/ou avaliar várias estratégias para a operação do sistema. O modelo, por sua vez, é uma abstração matemática de um processo, com simplificações inerentes. Com a computação moderna, uma boa simulação computacional se tornou uma ferramenta poderosa para a tomada de decisões com relação ao controle de um sistema, pois permite estudar e analisar situações que não seriam possíveis de outra maneira; também foi possível adicionar mais detalhes ao modelo sem comprometer o desempenho do sistema.

[Shannon 1998] ainda apresenta vantagens e desvantagens sobre simulações. Entre as vantagens temos: pode-se testar novas configurações do sistema, sem custos provindos da perturbação do sistema real; permite testar hipóteses sobre o porquê de certo fenômeno estar ocorrendo no sistema; permite o escalonamento do tempo, ou seja, podemos testar uma sequência de eventos com um tempo acima ou abaixo do que esses acontecimento demorariam na realidade (“avançar” ou “câmera lenta”). Entre os pontos negativos podemos citar: a grande dependência do modelo do sistema, que define a utilidade do simulador e requer um grande estudo para ser feito; ou a grande dependência dos dados de entrada, já que dados de entrada imprecisos podem levar a resultados insatisfatórios e/ou não significativos da simulação. Na literatura, a dependência da fidelidade dos dados de entrada é referida como *garbage In, garbage Out* (lixo entrado, lixo Saindo). O autor também divide o projeto de um simulador em várias etapas apresentadas em seguida juntamente com algumas ações apropriadas para cada uma.

Definição do problema: o objeto e o objetivo do estudo são definidos e devem ficar claros.

Plano de Projeto: deve verificar se há pessoal, suporte e recursos de *hardware* e *software* para executar a tarefa. Deve-se providenciá-los se for o caso.

Definição do Sistema: determinar os limites do sistema e investigar como ele funciona.

Formulação do Modelo Conceitual: modelagem do sistema, através de simples blocos gráficos ou em pseudocódigo.

Definição dos testes preliminares: seleção da política de avaliação de resultados a ser usada,

Preparação dos dados de entrada: identifica-se e coleta-se os dados de entrada necessários ao modelo

Tradução do modelo: programação do modelo em uma linguagem de simulação

Verificação e Validação: confirma-se se o modelo está operando da forma que o analista pretende (Verificação) e se a resposta obtida com o sistema reflete o que acontece no sistema real (validação).

Definição dos testes finais: projeta-se um experimento com a ferramente e verifica-se se a simulação retorna corretamente os resultados.

Experimento: execução da simulação para obter dados e proceder às análises sobre os resultados

Análise e Interpretação: estudo e inferência a respeito dos resultados obtidos com a simulação

Implementação e Documentação: relatar os resultados, relatar descobertas sobre o sistema, documentar o modelo utilizado.

O autor aponta também falhas comuns em projetos de simulação sem sucesso e virtudes comuns em projetos que alcançaram seus objetivos. Entre as graves falhas que o autor identificou como constantes estão: inexistência de um objeto claro para simulação; nível inapropriado de detalhes do sistema, o que pode ou tornar a simulação simples e não significativa ou comprometer o desempenho geral do sistema com muitos detalhes do processo; e falha na montagem de equipes. Entre as virtudes estão: canais de comunicação adequados entre a equipe e os futuros usuários da simulação; e garantia que o pessoal envolvido tem a habilidade necessária para a condução da simulação. Assim, pode-se obter sugestões para um bom projeto de simulação.

Para construir modelos válidos, a proposta apresentada por [Law 2008] é seguir sete passos principais. Também discute que se um modelo não é uma boa aproximação do fenômeno real, então qualquer conclusão gerada a partir desse vai ser errônea, daí a necessidade da validação, que deve ser aplicada antes da sua utilização. Os setes passos apresentados são mostrados a seguir e o fluxograma a ser seguido é mostrado na Figura 2.8.

Também apresenta várias técnicas para se obter um bom projeto de simulação. Entre elas: a entrevista com o pessoal experiente no processo simulado; a documentação do modelo conceitual, que deve ser preciso; e reuniões regulares da equipe.

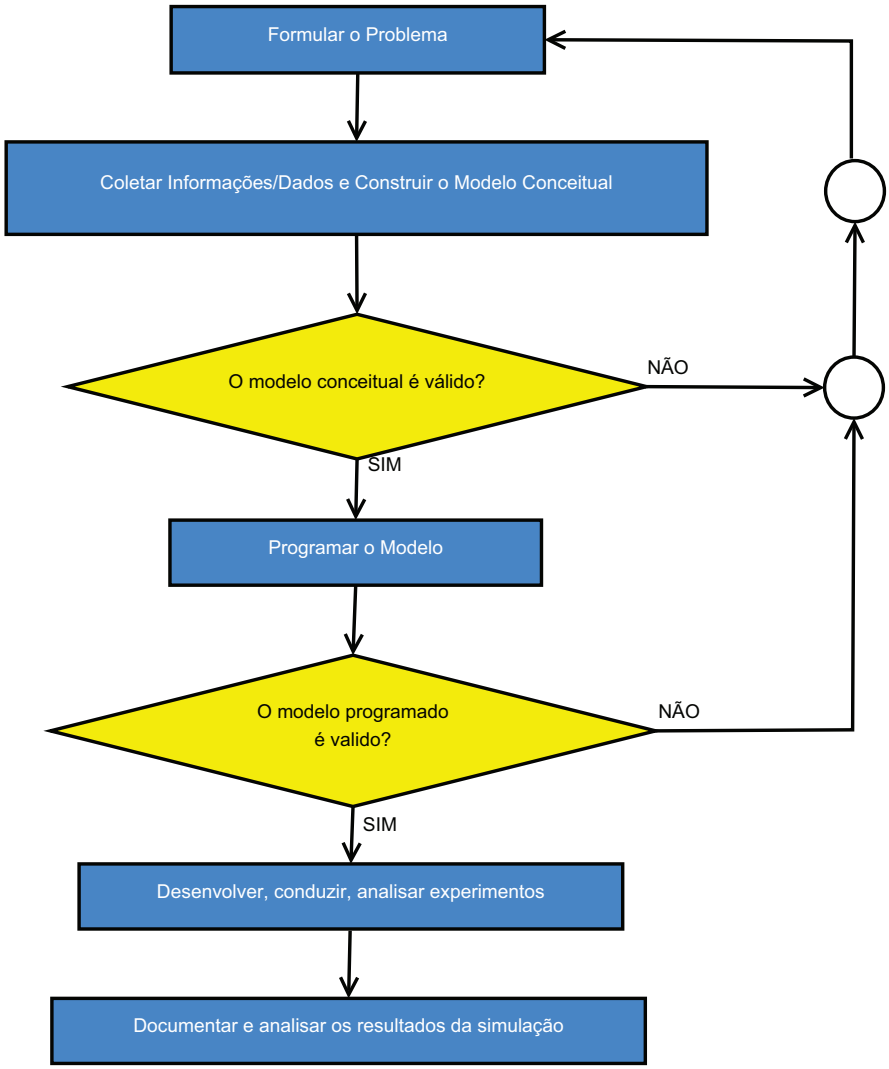


Figura 2.8: Fluxo dos Passos para Desenvolvimento de Simulação

CAPÍTULO 3

SIMULADOR

Nesse capítulo, é apresentado o objeto resultado do projeto, o *software* de simulação para o método PL. O projeto de atuação é “Desenvolvimento de Sistemas de Simulação, Controle e Supervisão para Elevação Artificial” (AUTOPOC). Na seção a seguir, é relatada a metodologia abordada na execução das fases; na seção 3.2 discute-se a respeito da arquitetura utilizada para a implementação do programa; e nas seções 3.3 e 3.4 é mostrado como o modelo da simulação foi implementado.

3.1 Metodologia

A etapa inicial se deu com o estudo e familiarização ao método de elevação artificial *Plunger Lift*. Com essa finalidade, primeiramente, buscou-se obter um conhecimento geral sobre os processos que envolvem a produção do petróleo, dando enfoque às partes constituintes de um poço de petróleo e o comportamento de um reservatório. Depois, foram estudados os métodos de elevação artificial, para isso, foi bastante importante a leitura das dissertações [do Nascimento 2005] e [Vidal 2005], não somente porque enfocam a elevação artificial, mas por descreverem como foram conduzidas as construções dos simuladores para poços com Método de Elevação Artificial por BM e por BCP, respectivamente. Assim pôde-se obter exemplos para a condução do nosso projeto.

Estudados brevemente os conceitos referentes à indústria do petróleo, pôde-se estudar efetivamente o *Plunger Lift*. Nessa etapa foram aprendidos como se dá o seu funcionamento: como o método efetivamente eleva o petróleo. Também se estudou os equipamentos necessários ao PL, procurando sempre entender a utilidade de cada um. Após esses estudos preliminares do método passou-se ao estudo das modelagens matemáticas propostas para o processo de Elevação Artificial PL convencional. Duas foram estudadas: o modelo proposto em [L. Marcano 1994] e o proposto em [Baruzzi 1994].

O modelo escolhido inicialmente para implementação é o apresentado em [Baruzzi 1994], que fez uma discussão sobre mecanismos para se ter acúmulo de líquido apenas

na coluna de produção (uma das premissas do seu modelo) e discutiu sobre o escoamento de fluidos pelo pistão. Posteriormente, pretende-se utilizar um modelo que reduza as simplificações que foram feitas. O projeto de pesquisa PETRELEV pretende construir uma melhora do modelo dinâmico existente, inclusive com a construção de uma planta piloto de dimensões reduzidas para reprodução do comportamento do poço. Os dados obtidos por esse projeto irão ser úteis ao desenvolvimento do simulador, já que um modelo mais exato pode ser incorporado, além disso, os dados de simulação podem ser comparados com resultados de experimentos obtidos com a planta piloto no futuro. Isso irá contribuir para a validação do simulador, permitindo a construção de uma ferramenta mais confiável para a tomada de decisões no futuro, e também para visualização de comportamentos do sistema para futura implementação no simulador.

Reuniões foram conduzidas para o estabelecimento de metas e direções para o andamento das atividades sobre o projeto:

20/05/2008: A reunião foi conduzida pelo eng. Edson Henrique Bolonhini que explicou o funcionamento do método além de apresentar o modelo [Baruzzi 1994], também sugeriu algumas funcionalidades para o simulador a ser desenvolvido, como a modelagem da queda do pistão e a possibilidade de definição das características do mesmo, por exemplo. Foi apresentado também o Simulador de PL desenvolvido por Kwon Il Choi (Petrobras).

11/06/2008: Conduzida pelo professor André Laurindo Maitelli (coordenador AUTOPOC), foi definido um cronograma para o projeto e definida a distribuição de atividades para o projeto: irei atuar principalmente nas rotinas numéricas para a solução das equações propostas no modelo.

19/06/2008: Em reunião conduzida na PETROBRAS pelo eng. Edson Henrique Bolonhini, foi sugerida a retirada de simplificações futuras do modelo, como considerar líquido no anular, novamente a modelagem da queda do pistão e a pressurização da linha de surgência.

30/07/2008: A reunião conduzida pelo Professor André Laurindo Maitelli, definiu os próximos passos para o projeto.

12/08/2008: Conduzida por Edson Bolonhini, foram mostrados exemplos de softwares já utilizados pela Petrobras para Supervisão do método *Plunger Lift* e Simulação do poços operando com *Gas Lift*.

Mais recentemente, reuniões para acompanhamento dos resultados e também para dirimir possíveis dúvidas foram conduzidas entre a equipe do projeto e o eng. Edson Bolonhini, que será o primeiro usuário do *software* desenvolvido:

07/05/2009: particularmente sobre a implementação do modelo, foi esclarecido que o controle do método na primeira implementação deveria ser manual

16/06/2009: O eng. Bolonhini propôs outras melhoras na implementação do modelo como a uma simulação inicial dos primeiros ciclos do processo antes da exibição de dados para o usuário de modo a obter valores confiáveis do processo a serem exibidos. Outra melhoria são os cálculos de algumas condições iniciais da simulação de acordo com certas características do poço a ser simulado, retirando-se dessa

forma a responsabilidades de configuração de alguns parâmetros iniciais por parte do usuário.

06/08/2009: uma nova reunião foi conduzida, para apresentar o primeiro protótipo do *software* desenvolvido até então.

A linguagem escolhida para a implementação do modelo foi C++ [Stroustrup 2000]. Na sua escolha pesaram as experiências com os projetos já finalizados, que utilizaram a mesma linguagem, e também a familiaridade dos integrantes do projeto com a mesma, a facilidade de comunicação com o *hardware* além do fator fundamental para sua escolha: o seu grande desempenho. O fator desempenho é particularmente importante de forma que a resolução das equações através de métodos numéricos, que podem envolver várias iterações e consumir recursos preciosos da máquina, não sobrecarregue o computador de modo que os resultados demorem a aparecer na interface gráfica, tornando-o um software lento e tedioso.

A linguagem C++ permite a orientação a objetos e, devido a seu comprovado benefício para modularização, manutenção e melhorias, nos utilizamos dessa para a organização do *software*. O padrão utilizado para a arquitetura do *software* foi o padrão MVC (*Model-View-Controler*) (MVC), cuja organização é mostrada na Figura 3.1.

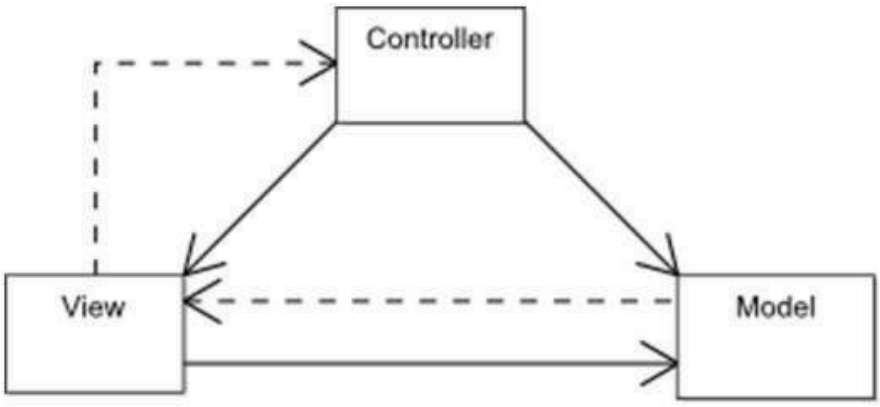


Figura 3.1: Esquema do padrão de projeto MVC

A arquitetura MVC é bastante utilizada. Especificamente no nosso projeto, essa arquitetura visa separar a estrutura da interface gráfica do usuário (*View*) da implementação das equações diferencias que tornam a simulação possível (*Model*). Assim sendo, é possível, por exemplo, que uma mesma interface funcione com mais de uma implementação de simulação e, reciprocamente, um mesmo simulador pode servir para mais de uma interface, por exemplo, para o caso em que se deseje proceder uma simulação sem uma interface gráfica associada, como em linha de comando.

A comunicação entre essas duas partes independentes é feita pelo gerenciador de dados do software (*Controller*), que é responsável também pelo armazenamento/recuperação dos dados gerados pelo modelo. Essa massa de dados gerados deve ser persistida em um Banco de Dados (BD), para permitir a abertura e fechamento de simulações e também possibilitar uma análise detalhada dos mesmos posteriormente, através de técnicas

como a mineração de dados, porém essa possível análise futura foge do escopo atual do projeto.

3.2 Arquitetura

Na Figura 3.2 é apresentado um desenho representativo da arquitetura do nosso sistema. Os blocos em azul claro representam os três pilares principais do sistema, cada um deles é executado em uma *thread*, fluxo de controle [Tanenbaum 2003], do programa.

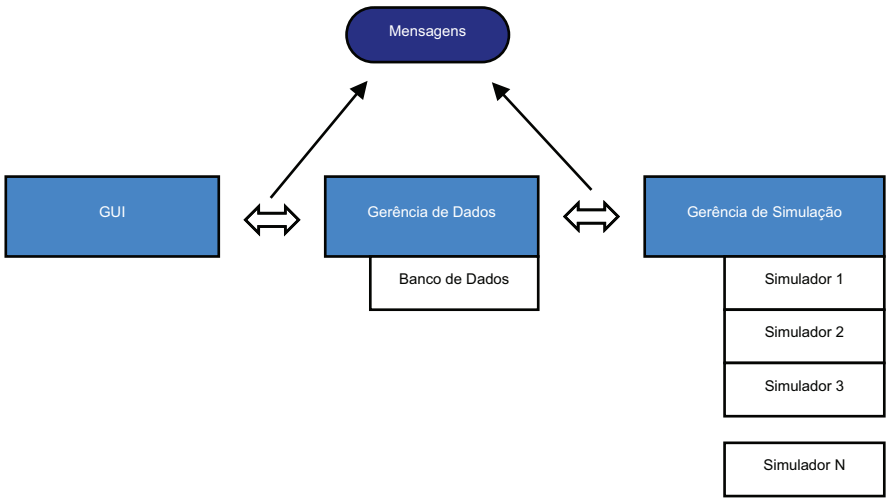


Figura 3.2: Arquitetura do Simulador *Plunger Lift*

Dessa forma pudemos implementar o padrão MVC. Percebe-se pela figura que tanto a interface quanto a simulação recebem/enviam requisições para a Gerência de Dados através de mensagens, possibilitando uma implementação distribuída do *software* com diferentes responsabilidades para cada componente do mesmo. A seguir são enumeramadas algumas da atribuições de cada uma dessas divisões (módulos).

- Interface Gráfica com o Usuário:
  - Supervisão do processo, através de gráficos e de animações e dados a respeito dos ciclos;
  - Nos gráficos deve ser permitido ao usuário mudar cores e nas animações é permitido ações como a rotação e o zoom;
  - Permitir a interferência do usuário no processo, como alterar os parâmetros do algoritmo de controle ou alguns parâmetros de configuração do sistema;
  - Permitir a edição, adição ou remoção de componentes do poço do Banco de Dados por parte do usuário;
- Gerência de Dados:
  - Atuar como intermediário na troca de mensagens entre a Interface com o usuário e o processo simulador;

  - Gerenciar a memória consumida pelo sistema para que os dados gerados pela simulação não sobrecarregue a máquina;
  - Permite o armazenamento de Dados (provindos de Simulações, perfil de usuário, componentes cadastrados etc.);

- Gerência de Simulação:
  - Tratar requisições de novas Simulações e disparar novas *threads* para os cálculos das equações;
  - Tratar requisições de mudança de parâmetros de simulação;
  - Parar simulações;
  - Enviar as amostras de variáveis provindas do processo para a Gerência de Dados.

Essa modularização do *software* foi pensada de modo a gerenciar a complexidade do sistema, facilitando a implementação de distintas características essenciais para o bom funcionamento do programa. Também é importante salientar a facilidade futura de atualizações, já que um módulo pode ser modificado sem afetar os outros, de fato, um módulo pode ser inteiramente substituído sem afetar os outros desde que se o novo módulo implemente uma interface predefinida para troca de mensagens entre eles.

Na Figura 3.3 é apresentado um esquema que representa a independências entre os componentes do sistema. Nela são apresentados dois subsistemas que se comunicam através de uma interface (no nosso caso, o *Pool* de Mensagens); há duas possibilidades de implementação do subsistema superior, representado pelos blocos azul e vermelho; mas para o bloco laranja (do subsistema inferior) a implementação do outro subsistema é transparente.

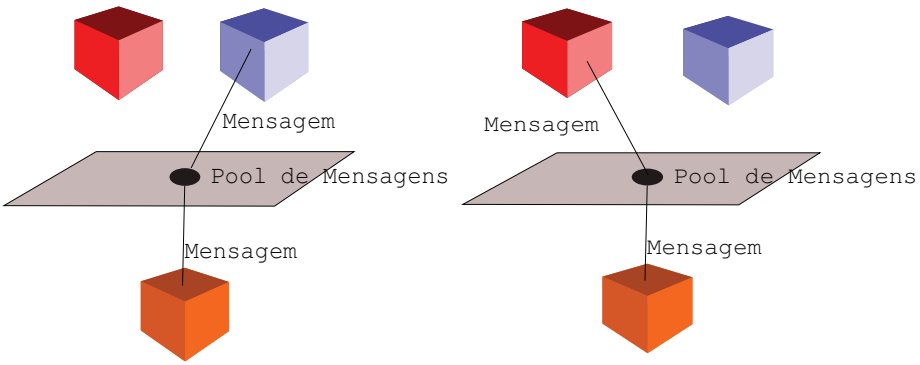


Figura 3.3: Esquema representativo da Modularização do Sistema

Percebe-se na Figura 3.3, as entidades **Mensagens** e **Pool de Mensagens**. As Mensagens são as informações trocadas entre os subsistemas do *software* e nada mais são que objetos de uma classe base chamada `PlMessage` cuja assinatura pode ser vista no trecho de código a seguir:

```
1  class PlMessage {

   public :
       explicit PlMessage( float time ) ;
       ~PlMessage() ;
6
       float getTime() const ;
       virtual int callHandle( class MessageHandler* o ) = 0 ;

       bool markAsHandled() ;
11     bool isHandled() ;

   protected :
       bool setTime( float newValue ) ;

16 private :

       bool init() ;
       bool init( float time ) ;

21     //! Tempo da mensagem
       float* mTime ;

       //! Indica quando uma mensagem já foi manipulada
       bool* handled ;
26
   } ;
```

Código 3.1: Classe `PlMessage`

Analisando essa classe, podemos perceber que não há atributos que possam servir de contêiner de dados ou informações, como uma lista de bytes (em C++, `char*`). Ao invés dessa abordagem mais tradicional e menos segura, optamos por explorar a capacidade de herança e fazer com que as mensagens específicas (com dados relevantes para a informação transmitida), herdem de `PlMessage`. Alguns exemplos de mensagens trocadas entre os módulos são citadas a seguir:

`PlMessage`: Classe base para as mensagens trocadas entre os módulos.

`InitSimulationMessage` Mensagem disparada pela GUI, quando se deseja criar uma nova simulação.

`StopMessage`: Mensagem disparada pela GUI, quando se deseja parar a simulação.

`ValueModificationMessage` Mensagem disparada pela GUI quando o usuário modifica algum parâmetro configurável em tempo de simulação.

`SimulationMessage` Classe base para mensagens geradas em uma simulação

`SampleMessage` Mensagem gerada pela simulação que possui amostras das variáveis relevantes do processo

`CycleVariableMessage` Disparada pela simulação com a informação de uma variável relevante para o ciclo do PL, como a velocidade média do pistão ou o tempo

de chegada.

Essas mensagens trocadas entre os módulos, precisam de um armazenamento intermediário entre a sua criação e o seu processamento por parte do “interlocutor”, esse é o papel do `MessagePool`. Essa classe atua como duas caixas de correios entre dois módulos (uma caixa para cada módulo), armazenando, assim, mensagens que ainda não foram processadas pelo módulo destinatário. Assim, em termos mais conhecidos de sistemas distribuídos, a caixa de correio nada mais é do que um *buffer*, em que irá atuar um produtor e um consumidor. Não nos preocupamos com nenhuma limitação de espaço desse *buffer*, pois o crescimento indefinido das mensagens armazenadas nesse contêiner é um evento pouco provável. Assim a comunicação entre dois módulos segue o esquema presente na Figura 3.4.

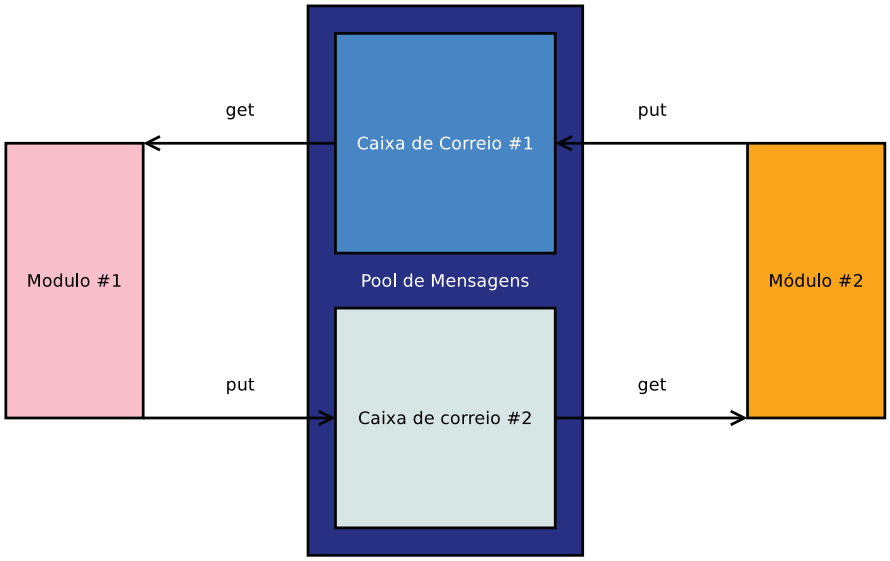


Figura 3.4: Comunicação entre dois módulos do sistema utilizando o esquema de caixa de correios

Os contêineres presentes no *pool* de mensagens são protegidos, cada um, com um *mutex* [Tanenbaum 2003] de modo que essa região crítica (contêineres) somente pode ser modificada, ou acessada, por um módulo por vez. Esse tipo de proteção de região crítica, chamado de compartilhamento de recursos, é bastante comum e uma Rede de Petri que exemplifica esse esquema é mostrado na Figura 3.5. Nessa rede, pode ser distinguidos o comportamento de dois módulos (*threads*), representados, cada um por três lugares. A definição de cada lugar é mostrada na Tabela 3.1. A marca no recurso compartilhado indica sua disponibilidades, enquanto, marcas nos lugares correspondentes aos módulos indica o estado das *threads*.

Observamos mais uma vez o código de `PlMessage` (ver Código 3.1) para focar no método virtual `callHandle(MessageHandler*)`. Esse método tem uma implementação diferente para cada tipo de mensagem, graças à característica chamada de **polimorfismo** presente em algumas linguagens de programação, a qual permite que dois objetos de

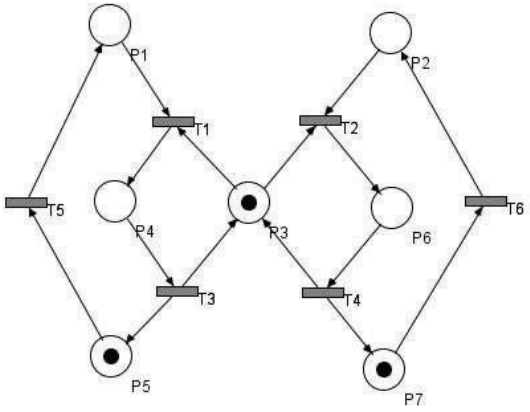


Figura 3.5: Rede de Petri simbolizando o compartilhamento de recursos

Lugar Módulo #1	Lugar Módulo #2	Descrição
P_1	P_2	Esperando recurso compartilhado
P_4	P_6	Utilizando recurso compartilhado
P_5	P_7	Execução normal
P_3		Recurso compartilhado

Tabela 3.1: Definição dos lugares da Rede de Petri que simboliza o compartilhamento de recursos

classes diferentes reajam diferentemente ao mesmo estímulo (chamada de método). Em C++ a característica de polimorfismo pode ser implementada através de métodos virtuais (que tem a palavra chave `virtual` em sua declaração). Assim sendo, uma mensagem, para ser tratada por uma classe *Handle*, deve ser requisitada que chame um método específico através da seu método `callHandle`. Cada *Handle* deve herdar da classe `MessageHandle`, cujo Código é mostrado a seguir:

```
//! Classe virtual responsavel pelo tratamento de mensagens
/*!
 * Os metodos sao implementados com comportamento <i>default</i>: nao fazer
 * nada. As classes que herdeiras devem reimplementar os metodos que tratam
 * de mensagems de interesse
 *
 * \author Cristiano Castro <crisgc@dca.ufrn.br>
 */
class MessageHandler {
public :
    virtual int handleInitSimulationMessage( class InitSimulationMessage* ) ;
    virtual int handleSampleMessage( class SampleMessage* ) ;
    virtual int handleStorageMessage( class StorageMessage* ) ;
    virtual int handleEndOfStageMessage( class EndOfStageMessage* ) ;
    virtual int handleEndOfCycleMessage( class EndOfCycleMessage* ) ;
```

```
    virtual int handleValueModificationMessage( class ValueModificationMessage*
        ) ;
    virtual int handleCycleVariableMessage( class CycleVariableMessage* ) ;
    virtual int handleFinishSimulationMessage( class FinishSimulationMessage* )
        ;
    virtual int handleStopMessage( class StopMessage* ) ;
    virtual int handleWarningSimulationMessage( class WarningSimulationMessage*
        ) ;
};
```

Código 3.2: Classe tratadora de mensagens

Nas próximas seções será discutido a Gerência de Simulação e a Implementação do Modelo *Plunger Lift*. As implementações da interface gráfica e do Banco de Dados será mostrada apenas no no capítulo 4, que trata dos resultados alcançados.

3.3 Gerência de Simulação

A gerência de simulação (`SimulatorManager`) é uma classe relativamente simples responsável por organizar a execução de simuladores computacionais e por intermediar a comunicação dos mesmos com a Gerência de Dados.

A única instância existente dessa classe recebe requisições para inicialização e término de uma simulação. A cada inicialização, uma nova *thread* é criada para o cálculo das equações diferenciais que implementam o modelo. As variáveis amostradas são relatados à gerência e repassados para os demais componentes do programa (Figura 3.2). O término também é gerenciado. Ao se receber uma mensagem de *stop*, a gerência imediatamente inviabiliza a comunicação de variáveis proveniente da simulação e, por fim, trata de que a *thread* que anteriormente estava executando a simulação computacional seja corretamente finalizada.

Esse tipo de organização irá facilitar posteriormente uma nova atualização da simulação para que várias simulações possam ser executadas em um mesma instância do simulador. Como também poderá dar suporte a outras melhorias, facilitando a escalabilidade do *software*.

3.4 Implementação do Modelo

A implementação atual baseia-se na orientação a objetos, isto é, o comportamento de cada componente do poço é refletido em uma entidade (classe) do programa. Este tipo de implementação foi adequado para uma primeira implementação da simulação. Embora, outra soluções de implementação possam ser mais adequadas futuramente, como uma abordagem matricial para resolução de equações diferenciais através do problema de valor de contorno.

Na implementação atual há entidades como Reservatório, Pistão, Tubulação, Poço de Petróleo etc., onde cada entidade possui um comportamento coerente com sua definição. Um exemplo é visto a seguir, na classe Reservatório temos a seguinte organização:

```
//! Classe que implementa o comportamento de uma rocha-reservatrio
/*!
 * \author Cristiano Castro
 */
class Reservoir : public Visitable {
public:
    Reservoir( const ReservoirData* data , Gas* , Liquid*  ) ;
    ~Reservoir() ;
    int getID() const ;
    bool setID(int id) ;
    bool produce( float pwf , float * liquid , float * gas ,
    float timeStep ) const;
    float getGlr() const ;
    int setGlr(const float newGlr);
    float getGor() const ;
    int setGor(const float newGor) ;
    float getTestFlowRate() const ;
    bool setTestFlowRate( float newTestFlowRate ) ;
    float getTestPressure() const ;
    bool setTestPressure( float newTestPressure ) ;
    float getStaticPressure() const ;
    int setStaticPressure( float newStaticPressure ) ;
    /* Vazes */
    const float getLiquidFlowRate() const ;
    const float getGasFlowRate() const ;
    int accept( Visitor* v ) ;
    const Gas* getGasInfo() const ;
    const Liquid* getLiquidInfo() const ;
protected :
    Gas* getGas() ;
    Liquid* getLiquid() ;
    int setGas( Gas* ) ;
    int setLiquid( Liquid* ) ;
private:
    //! Identificador
    int* id;
    //! Razo gs-lquido
    float* glr;
//! Taxa de teste de fluxo (em m3/s)
float* testFlowRate;
//! Pressao de teste
float* testPressure;
//! Presso esttica do reservatrio
float* staticPressure;
//! Vazo de lquido reservatrio
float* liquidFlowRate ;
// Vazo de gs do reservatrio
```

```
    float* gasFlowRate ;
49  //! Informaes do gs armazenado no respositrio
    Gas* gas ;
    //! Informaes do lquido armazenado no repositrio
    Liquid* liquid ;
    float getMaxFlowRate() const ;
54  void init( const ReservoirData* data , Gas* , Liquid* ) ;
    friend class WellFactory ;
    friend class ValueModificationVisitor ;
    };
```

Código 3.3: Assinatura da classe Reservatório

Através da análise do Código 3.3, pode-se visualizar que classe que o reservatório tem como variáveis suas características próprias, como a razão entre gás e líquido (`glr`) e a sua pressão estática (`staticPressure`), por exemplo. O único comportamento que o reservatório tem no simulador, a produção de líquido e gás, é implementado também nessa classe, através do método `produce`. A estrutura da implementação pode ser visualizada na Figura 3.6

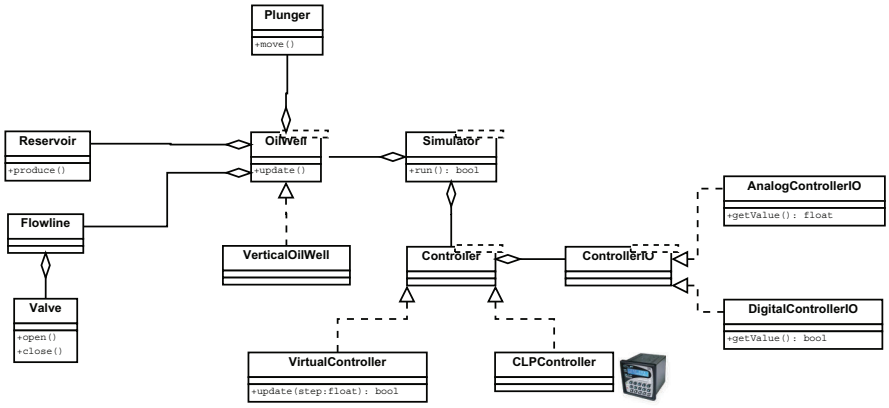


Figura 3.6: Diagrama de Classes Simplificado com a estrutura da implementação do Modelo

3.4.1 Análise das classes

A classe `Simulator` possui um método para executar ciclos de simulação do *Plunger Lift* (método `run`). Nessa classe é armazenado o tempo de simulação, os passos de integração para cada etapa do ciclo e também os intervalos de tempo para amostragem a cada ciclo. Para cada etapa de um ciclo PL, o simulador executa os cálculo numéricos necessários e, ao final, incrementa o tempo de simulação com o passo de integração correspondente. Procurando deixar espaço para posterior expansão, a classe que comanda a execução do método convencional de *Plunger Lift*, o único abordado até agora, é imple-

mentado na classe `ConventionalPLSimulator`, subclasse de `Simulator`. Assim, caso outra implementação seja sugerida, basta também utilizar de herança.

Associado a um simulador, há um controlador (instância da classe abstrata `Controller`) que é responsável por encapsular o algoritmo de controle do método. Uma das classes quem implementam o controlador é a `VirtualController` que nada mais é do que um controlador manual, em que o usuário define, através da GUI, os tempos de abertura e fechamento da *Motor Valve* a cada ciclo. Posteriormente, uma nova subclasse derivada de controle (`ClpControler`) irá se comunicar com um CLP para que o algoritmo de controle implementado neste possa ser usado para controlar o poço virtual. A estrutura é apresentada na Figura 3.7.

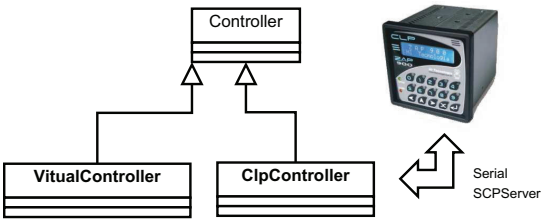


Figura 3.7: Controle do Processo

A classe `Controller`, se comunica com o processo através de entradas e saídas (*IO*), que nada mais são que variáveis (analógicas ou digitais) encapsuladas, que recebem dados dos "sensores" da simulação (pressão no topo do anular, pressão no topo do *casing*, sensor de chegada do pistão) e também nos atuadores (abertura ou fechamento da *Motor Valve* e também a quantidade de gás injetado ou retirado no caso de PL Assistido).

O simulador possui uma instância de um poço de petróleo (`OilWell`) , que é a peça fundamental do processo de simulação, já que nessa classe estão implementadas a maior parte da resolução numérica das equações diferenciais. A cada passo de integração, a classe `ConventionalPLSimulator` envia mensagens para o `OilWell`, indicando que um intervalo de tempo passou e informando o intervalo de tempo em questão (passo de integração), e este é responsável pelo seu próprio comportamento, ou seja, pelos equilíbrio de pressões, pelo balanço da quantidade de mols de gás nas colunas do anular e da coluna de produção e pelos volumes da golfada acima do pistão e no piso da coluna de produção, pela produção do reservatório, através da classe `Reservoir` apresentada no Código 3.3), e pelo movimento do pistão, através da classe `Plunger`. Funciona portanto como um "organizador" da resolução das equações. Pelos mesmos motivos apresentados anteriormente para a classe `Simulator`, a `OilWell` também é abstrata. `VerticalOilWell` é a entidade que é utilizada para a para poços verticais (a única tratada até agora pelo modelo até agora). Pretende-se posteriormente abordar os poços direcionais, já que as equações diferenciais devem ser modificadas para levarem em consideração a inclinação da parte da coluna de produção (ou anular) a qual elas se aplicam.

O poço de petróleo também possui um pistão (`Plunger`), cujo comportamento consiste em mover-se através da coluna de acordo com o diferencial de pressão na sua extremidade inferior. O pistão possui uma velocidade e uma posição associadas. O reservatório (`Reservoir`) que produz óleo e gás de acordo com a pressão de fluxo de fundo

e também há dados a respeito dos fluidos que nele contido. As propriedades dos fluidos gás, óleo, água e do líquido (mistura de água e óleo) presentes no reservatório estão nas classes `Gas`, `Oil`, `Water` e `Liquid`, respectivamente. Em `Gas`, por exemplo, temos armazenadas propriedades como a densidade do gás, em `Oil` é armazenado o grau API do mesmo, em `Liquid` tem-se a razão entre óleo e água (BSW) do mesmo.

Há também outras classes utilitárias, como `PressureUtils`, que armazena a pressão atmosférica e possui métodos para o cálculo pressão do topo (piso) de uma coluna de gás através da sua altura, das duas temperaturas do piso e do topo e também de sua pressão no piso (topo); ou `TemperatureUtils`, que armazena a temperatura de superfície, a temperatura padrão, o gradiente geotérmico da terra e possui métodos para o cálculo de temperatura a uma dada profundidade.

Essa estrutura está sujeita a modificações contínuas no decorrer do projeto. Pois desde o início da implementação, já houve algumas mudanças, algumas classes foram adicionadas e outras foram retiradas.

3.4.2 Padrões de Projeto

Também lançou-se mão no decorrer do projeto, de padrões de projetos para auxiliar na implementação do simulador. Os dois principais foram: *Visitor* e *Listener*.

O padrão de projeto *Visitor* permite que se defina uma nova operação para atuar em uma estrutura de dados, sem que se modifique a implementação das classes em que se opera. Nesse padrão temos duas classes principais: `Visitable` (Anfitriã), que encapsula os dados e `Visitor` (Visitante), que implementa a (nova) operação sobre os dados. Ele foi útil na implementação do modelo a medida em que operações, como a amostragem, que não influem na efetiva simulação do modelo, puderam ser implementadas em classes visitantes auxiliares. No Código 3.3, pode-se ver que a classe implementada herda de `Visitable`, ou seja, implementa o método `accept`, o qual manda uma mensagem para o visitor e este executa sua operação. Esse padrão de projeto também nos foi bastante útil para recuperar informações de tipo em C++, já que a linguagem não oferece um sistema dinâmico de verificação. Tal estratégia foi utilizada com as classes de mensagens explicado na Seção 3.2. As assinaturas das classes Anfitriã e Visitante são mostrados nos Códigos 3.4 e 3.5 respectivamente.

```
class Visitor ;
//! Classe base para todos os objetos que aceitam visitantes
/*!
 * Implementa apenas um mtodo para aceitar a visita
 *
 * \author Cristiano Gurgel de Castro <crisgc@dca.ufrn.br>
 */
class Visitable {
public:
    virtual int accept( Visitor* visitor ) = 0 ;
} ;
```

Código 3.4: Classe Anfitriã

```
//! Classe para visitar os objetos do modelo
class Visitor {
public:
    virtual int visit( Visitable* visitable ) ;
    // virtual bool visitConventionalPLSimulator( class ConventionalPLSimulator* ) ;
    virtual int visitController( class Controller* ) ;
    virtual int visitVirtualController( class VirtualController* ) ;
    virtual int visitOilWell( class OilWell* ) ;
    virtual int visitPipe( class Pipe* ) ;
    virtual int visitValve( class Valve* ) ;
    virtual int visitFlowLine( class FlowLine* ) ;
    virtual int visitGasInjectionLine( class GasInjectionLine* ) ;
    virtual int visitPlunger( class Plunger* ) ;
    virtual int visitReservoir( class Reservoir* ) ;
    virtual int visitLiquid( class Liquid* ) ;
    virtual int visitGas( class Gas* ) ;
    virtual int visitWater( class Water* ) ;
    virtual int visitOil( class Oil* ) ;
    virtual int visitTemperatureUtils( class TemperatureUtils* ) ;
    virtual int visitPressureUtils( class PressureUtils* ) ;
    virtual int visitVerticalOilWell( class VerticalOilWell* ) ;
    virtual int visitAnalogControllerIO( class AnalogControllerIO* ) ;
    virtual int visitDigitalControllerIO( class DigitalControllerIO* ) ;
    virtual int visitConventionalPLSimulator( class ConventionalPLSimulator* )
        ;
} ;
```

Código 3.5: Classe Visitante

O outro padrão de projeto amplamente utilizado foi o Observador (*Observer*) ou Ouvinte (*Listener*). Nesse padrão, como no *Visitor*, há duas classes principais: o `Observable` e o `Observer`. À uma entidade observável pode estar anexado vários observadores, estes por sua vez reagem a mudanças de estados indicados pelo primeiro. Para exemplificar o conceito no nosso caso, a classe `Simulator` implementa um *Observable* e as mudanças de estado disparados são “atualização do tempo de simulação”, “estouro do tempo de amostragem” , “fim de uma etapa de PL”, “fim de ciclo do PL”, etc., e cada ouvinte pode reagir de um modo diferente a um evento; no evento de “estouro do tempo de amostragem”, por exemplo, um dos ouvintes implementa a coleta de pontos referentes a pressões, altura da golfada, etc. para o envio à gerência de dados e à interface gráfica. Alguns eventos disparados pela simulação são mostrados a seguir:

`SampleTimeOverflowEvent` Evento disparado quando o tempo de amostragem estourou, indicando que se deve coletar os pontos.

`TimeUpdateEvent` O tempo da simulação foi atualizado, ou seja, indica mais um passo de integração.

`OilReachFlowLineEvent` Óleo começou a ser produzido no poço.

`PlungerArrivedEvent` Pistão chegou à superfície.

Esses padrões auxiliam também na implementação de outros comportamentos, que ainda precisam de uma implementação mais robusta, como o "retorno no tempo", em que o usuário volta o tempo de simulação passado para possivelmente alterar algum valor pertinente e retoma a simulação a partir desse ponto passado.

3.4.3 Documentação

Desde o início da implementação, um dos pontos que foram abordados é a documentação do programa. Dada a complexidade do *software*, este item é de fundamental importância tanto para o entendimento entre os membros da equipe, quanto para a posterior manutenção e adição de novas funcionalidades para o simulador. Para isso, estamos documentando o código no estilo da ferramenta *Doxygen* [van Heesch 2009], que pode gerar documentação nos formatos HTML e LaTeX entre outros. Mais recentemente, começou-se também a ser utilizado o *Trac* [Software 2009] para o gerenciamento das tarefas resultantes do projeto, possibilitando o monitoramento das atividades. Também é possível criar páginas Wiki, o que pode ser útil no intuito de se criar um *site* da ferramenta.

CAPÍTULO 4

RESULTADOS

Depois de pouco mais de um ano e meio de projeto. Teve-se, como resultado inicial, uma compreensão mais aprofundada dos fenômenos físicos, principalmente na área de Mecânica dos Fluidos, que regem a etapa de Elevação Artificial. O *software* já tem uma primeira versão, cuja tela principal é mostrada na Figura 4.1. Essa tela mostra as possibilidades que o usuário tem para controlar o processo de simulação.

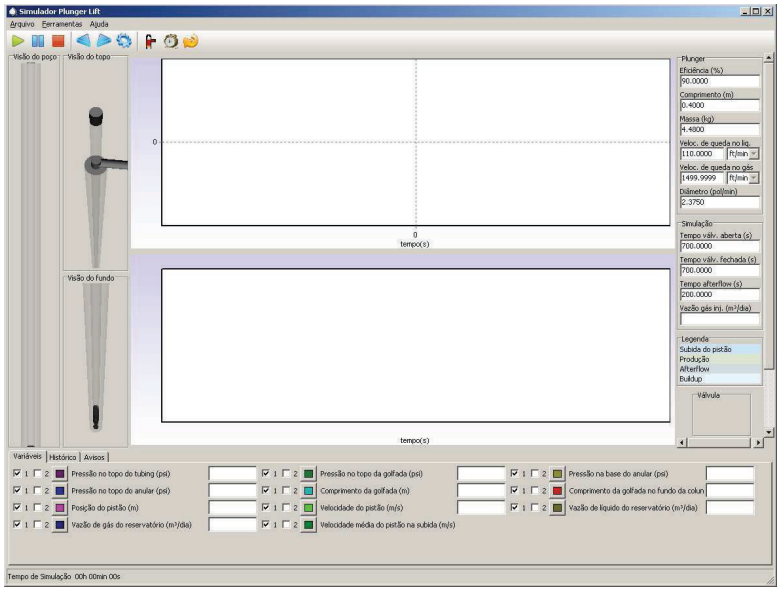


Figura 4.1: Tela Inicial da Simulação

4.1 Componentes da Interface

O desenvolvimento da interface não faz parte do escopo desse Relatório e, por isso, seus componentes serão explicados brevemente. Ela teve como base alguns casos de uso definidos pela equipe de programação nas etapas iniciais. A GUI (figura 4.1) é composta principalmente por gráficos atualizados periodicamente, contendo os resultados (amostras) obtidos com a simulação; um modelo 3D animado do poço; caixas de texto na parte direita e um painel na parte inferior da tela. Neste último pode-se escolher as variáveis a serem exibidas nos gráficos, e visualizar o seu valor instantâneo. Pode também visualizar nesse painel o histórico das variáveis importantes em todos os ciclos simulados. As caixas de texto na parte direita da interface permitem ao usuário interferir em parâmetros de simulação em tempo de execução. Há ainda uma janela de configuração (ver figura 4.2), para acesso às opções da simulação, e que permite inserir os dados do poço para que se dê início à simulação.

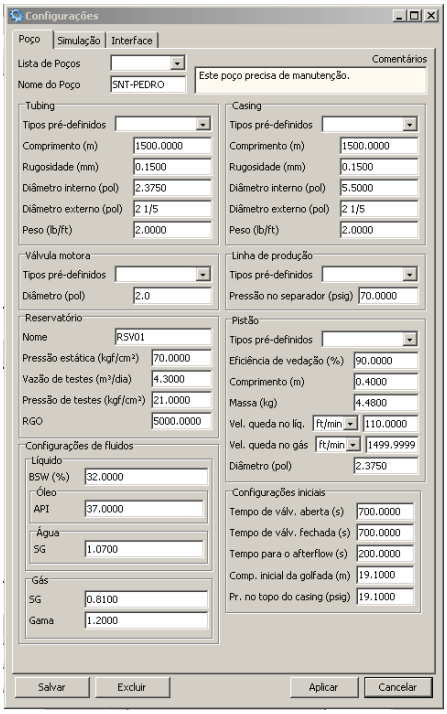


Figura 4.2: Tela de Configuração de uma Simulação

A Barra de Ferramentas, mostrada na Figura 4.3, permite a configuração de uma nova simulação e sua inicialização. As funcionalidades são apresentadas a seguir, seguindo a ordem da esquerda para a direita:

***Play*,*Pause*,*Stop*:** controle de fluxo da simulação. A simulação pode ser pausada ou interrompida em qualquer momento que se desejar. Após uma pausa, é possível resumir a mesma simulação sem que haja perda de dados. Após uma parada, a menos que o usuário salve os resultados obtidos, todos os dados são perdidos e outra simulação deve ser iniciada.

Voltar,adiantar gráficos: permite a visualização de um ciclo no gráfico.
Configurações: abre a tela de configurações.
Ocultar/Exibir animações 3D: útil para tornar a velocidade da simulação um pouco mais rápida sem as animações.
Voltar no tempo: permite a escolha de um ponto anterior para retornar o processo de simulação.

Figura 4.3: Barra de ferramentas da simulação

Os gráficos, um superior e um inferior, ver Figura 4.4 dispõem de vários recursos para melhorar a experiência do usuário ao visualizá-los. Durante a execução da simulação ele pode dar um zoom em qualquer região do gráfico para melhor visualizar o que está acontecendo durante certo período de tempo, ou mover o gráfico no tempo livremente ou ciclo a ciclo para observar efeitos que ocorreram anteriormente na simulação. Os usuários também estão livres para escolher quais variáveis serão exibidas e em qual dos gráficos (há a possibilidade de exibir a mesma variável em ambos) através da aba de variáveis (ver Figura 4.5). Outro aspecto personalizável são as cores representativas das variáveis. Os gráficos são a forma fundamental para se analisar o comportamento do poço simulado.

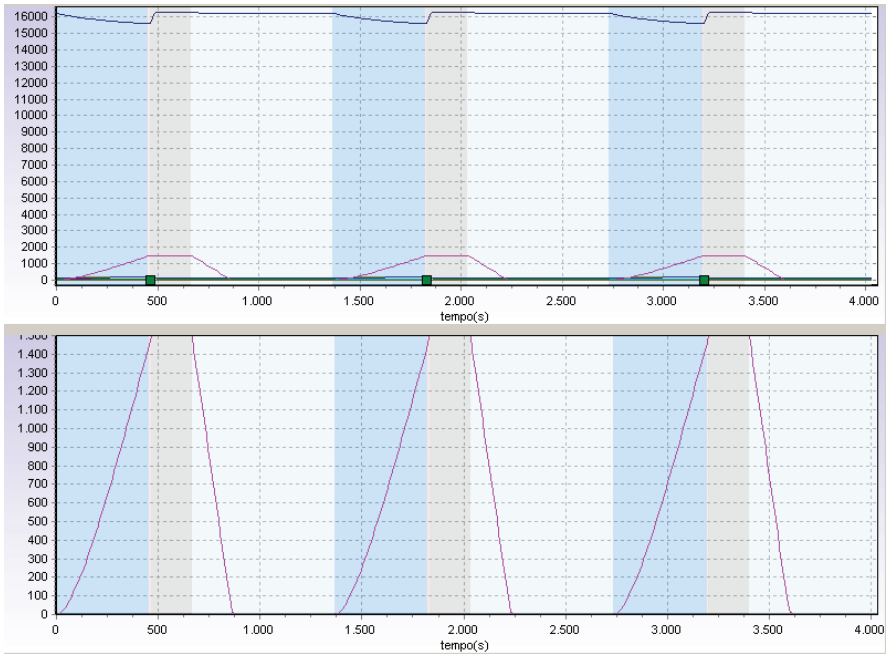


Figura 4.4: Gráficos

O modelo 3D animado (ver Figura 4.6) acompanha a execução da simulação no tempo. Há três telas de visualização do comportamento do poço: uma visão total do

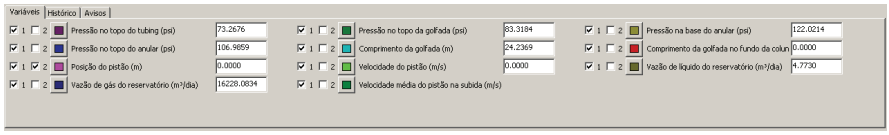


Figura 4.5: Aba para configuração de variáveis

Figura 4.6: Animação 3D

poço, uma visão na parte inferior do poço, uma visão da árvore de natal. Esses modelos desenvolvidos em OpenGL.

O Histórico (ver Figura 4.7) contém valores de variáveis relevantes referentes a cada ciclo do *Plunger Lift*, como velocidade média do pistão na subida, tempo de chegada da golfada, tempo de chegada do pistão, entre outras. Os valores dessas variáveis podem ser exportados para planilhas eletrônicas, caso o usuário deseje, e, assim, permite um maior poder de análise desses dados.

Variáveis | Histórico | Avisos

Número do ciclo	1	2	3	4	5	6	7
Velocidade média do pistão na subida (m/s)	2.38	3.22	3.21	3.21	3.21	3.21	3.21
Tempo de chegada da golfada (s)	625.49	461.03	462.83	462.93	462.93	462.93	462.93
Tempo de chegada do pistão (s)	629.04	465.14	466.99	467.11	467.11	467.11	467.11
Velocidade de impacto (m/s)	19.82	20.89	19.73	21.29	21.28	21.28	21.28
Produção (L)	65.27	73.75	74.63	74.73	74.73	74.73	74.73
Duração do ciclo (s)	1528.92	1365.99	1368.40	1367.44	1367.44	1366.59	

Figura 4.7: Histórico da simulação

4.2 Funcionamento

Ao se inicializar o *software*, é preciso primeiramente inicializar um poço para simulação, através da tela de configurações (rever Figura 4.2). Para inicializar, basta clicar no

botão *Play* da barra de ferramentas. Nesse instante, é enviada uma mensagem de início da simulação e a Interface espera os pontos retornados para a plotagem nos gráficos. Logo quando se inicia uma simulação, há um certo intervalo no qual o simulador tenta "estabilizar", ou seja, simula ciclos iniciais para colocar o poço em estado estacionário, caso os parâmetros de inicialização do usuário sejam válidos.

Após isso, começa-se a serem apresentados os pontos para o usuário da simulação. Os gráficos apresentados até agora com o início da simulação revelam dados interessantes, como a rápida aceleração do pistão quando da abertura da válvula da *Motor Valve* e o pico de velocidade do mesmo na etapa de produção da golfada. Porém a implementação do modelo passará por uma revisão minuciosa, pois dados indesejados principalmente no que se refere à pressões do poço estão ocorrendo. Um erro comum observado até agora, por exemplo, é a queda da pressão no *tubing* durante a etapa de *Build-Up*, que acredita-se ser devido às simplificações presentes no modelo.

CAPÍTULO 5

CONCLUSÕES

5.1 Conclusões

Com esse trabalho pretendemos dar uma contribuição para a otimização da produção de poços com elevação por PL, prevenindo falhas relacionadas a operadores humanos e melhorando aspectos relacionados à economia do processo, devido a uma maior produtividade conseguida com um bom controlador.

Em um primeiro momento do projeto, foi importante para a familiarização com conhecer o processo que irá ser simulado, isto é, a elevação por PL. Embora, esse estudo seja contínuo, a partir do início do desenvolvimento, as atenções foram divididas também com a implementação do modelo.

Um protótipo já foi entregue, porém o modelo implementado ainda deverá sofrer alterações, devido à ainda inconsistências relacionadas ao comportamento do poço simulado em comparação à poços reais.

Os projetos relacionados ao modelo de PL (AUTOPOC e PETRELEV) deverão caminhar em conjunto para promover o sucesso de ambos. Essa parceria diz respeito à melhora do modelo utilizado para descrever o sistema, útil para o simulador, e também na utilização do simulador para obter um maior entendimento do método. Caso este sucesso seja obtido a Universidade Federal do Rio Grande do Norte poderá ser reconhecida como um importante polo de pesquisa neste tipo de Elevação Artificial.

Os primeiros resultados foram importantes, porém há a necessidade de maiores estudos e ainda há melhoras a serem implementadas.

5.2 Próximas Etapas

Os próximos passos a serem tomados mais urgentemente é a análise dos dados obtidos e a revisão do modelo corrente implementado, para identificação de falhas e prover

estratégias para solução de erros. O aprofundamento de estudos na área de transporte de fluido serão também efetuados, para permitir um maior poder de análise.

Em outra vertente, também a curto prazo, a integração do simulador com o Controlador Lógico Programável (CLP) também deverá ser efetuada e testada. Esse ponto é de extrema importância para o teste de algoritmos de controle.

Finalmente, é também importante um movimento em torno da documentação do *software* com a construção de diagramas de caso de uso, de classes (utilizando-se de UML), documentação da arquitetura do *software*, além da documentação do Código Fonte no estilo *Doxygen*. Para o usuário, é possível a criação de um menu de ajuda para o simulador e um sítio para uma interação contínua com a equipe de projeto, com a possibilidade de visualização das atividades em andamento (utilização da ferramenta *Trac*).

REFERÊNCIAS BIBLIOGRÁFICAS

Baruzzi, J. O. A. (1994), Modelagem do plunger lift convencional., Dissertação de mestrado, Universidade Estadual de Campinas.

Baruzzi J.O.A., Alhanati F.J.S., PETROBRAS (1995), Optimum plunger lift operation, *em* 'SPE Production Operations Symposium, 2-4 April 1995, Oklahoma City, Oklahoma'.
URL: *http://www.onepetro.org/mslib/servlet/onepetropreview?id=00029455&soc=SPE*

Bolonhini, E. H. (2005), Plunger lift conceituação fundamental.

do Nascimento, J. M. A. (2005), Simulador computacional para poços de petróleo com método de elevação artificial por bombeio mecânico, Dissertação (mestrado), Universidade Federal do Rio Grande do Norte.

J. Chacin, Intevep S.A. [et al.] (1994), 'Modeling and optimization of plunger lift assisted intermittent gas lift installations', *SPE Advanced Technology Series* **2**(1), 25–33.
URL: *http://www.onepetro.org/mslib/servlet/onepetropreview?id=00023683&soc=SPE*

L. Marcano, J. Chacin, Intevep S.A. (1994), 'Mechanistic design of conventional plunger-lift installation', *SPE Advanced Technology Series* **2**(1), 15–24.

Law, Averill M. (2008), 'How to build valid and credible simulation models', pp. 39–47.

Shannon, Robert E. (1998), Introduction to the art and science of simulation, *em* 'WSC '98: Proceedings of the 30th conference on Winter simulation', IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 7–14.

Software, Edgewall (2009), 'Trac open source project'.
URL: *http://trac.edgewall.org/*

Stroustrup, B. (2000), *The C++ Programming Language*, Addison-Wesley.

Tanenbaum, A. S. (2003), *Sistemas Operacionais Modernos*, Pearson Prentice Hall.

Thomas, J. E. [et al.] (2001), *Fundamentos de Engenharia de Petróleo*, Interciência.

van Heesch, Dimitri (2009), 'Doxygen'. 24.
URL: *http://www.stack.nl/ dimitri/doxygen/*

Vidal, Francisco José Targino (2005), Desenvolvimento de um simulador de bombeio por cavidades progressivas, Dissertação de mestrado, Universidade Federal do Rio Grande do Norte.

Vidal, José Romualdo Dantas (2008), 'Notas de aula da disciplina deq0376 - introdução a engenharia de petróleo'.