7.
Referências Bibliograficas

Adams, A. J., and Angus MacEachran. "Impact on Casing Design of Thermal Expansion of Fluids in Confined Annuli." *Society of Petroleum Engineers*, Março 7, 1994: 210-216.

Alcofra, Elisa Lage Modesto. *Aumento de Pressão de Fluido Confinado no Anular de poços de Petróleo.* Dissertação de Mestrado, Departamento de Engenharia Mecânica, PUC-Rio, Rio de Janeiro, RJ: PUC-Rio, 2014, 33-64.

API 13D. "Rheology and Hydraulics of Oil-Well Fluids." Maio 2010. 22-25.

Botelho, Fabricio Vieira Cunha. *Análise Numérica do Comportamento Mecânico do Sal em Poços de Petróleo.* Dissertaão de Mestrado, Engenharia Civil, PUC-Rio, Rio de Janeiro: PUC-Rio, 2008, 211.

Costa, A. M., E. Jr. Poiate, and J. L. Falcão. "Geomechanics Applied to the Well Design Through Salt Layers in Brazil: A History of Success." *American Rock Mechanics Association*, Junho 27-30, 2010.

Costa, Alvaro Maia. "Uma Aplicação de Métodos Computacionais e Princípios de Mecânica das Rochas no Projeto e Análise de escavações Destinadas à Mineração Subterrânea." Tese de Doutorado, Departamento de Engenharia Civíl, UFRJ, Rio de Janeiro, 1984, 174-184.

Grainger, Phillip Afonso de Melo. *Numerical Analysis Of The Mechanical Behavior Of Cement Sheaths In Wells Through Salt Formations.* Dissertação de Mestrado, Engenharia Civil, PUC-Rio, Rio de Janeiro: PUC-Rio, 2012, 134.

Halal, A. S., and R. F. Mitchell. "Casing design for Trapped Annular Presure Buildup." *SPE Drilling & Completion*, June 1994: 107-114.

Hansen, Frank D., Matthew J. Leroch, Leo Van Sambeek, and Mao S. Lin. "Gas Barrier Design For the WIPP." Junho 28-30, 1993: 1-4.

Hunsche, U., and A. Hampel. "Rock Salt: The Mechanical Properties of the Host Rock Material for a Radioactive Waste repository." *Engineering Geology*, 1999: 271-291.

Jandhyala, S., Y. R. Barhate, J. Anjos, and C. E. Fonseca. "Cement Sheath Integrity in Fast Creeping Salts: Effect of Well Operations." *Society Petroleum Engineers*, Setembro 3 a 6, 2013: 1-10.

Jandhyala, Siva Krishna, and Abhinandan Chiney. "Finite Element Approach to Predict the Effect of Annular Pressure Buildup on Wellbore Materials." *Offshore Technology Conference*, Março 25-28, 2014: 1-8.

Lobão, Diomar Cesar. "Material de Métodos Numéricos." *Site dos professores da UFF*. março 21, 2015. http://www.professores.uff.br/diomar_cesar_lobao/material/Metodos_Numericos/UFF_Metodos_Numericos.pdf (accessed 2015).

Menezes, Ivan F.M., Luiz Eloy Vaz, and Anderson Pereira. "Material de Otimização de Algorítmos." Vers. 2015.1. *Site da TECGRAF*. março 22, 2015. http://webserver2.tecgraf.puc-rio.br/~ivan/MEC2011/ProgMatematica_VazPereiraMenezes-Ago2012.pdf (accessed 2015).

Munson, D. E., and W. R. Wawersik. "Constitutive Model of Salt Behavior - State of Tecnology." Tecnichal Report, Departmento of Energy (DOE), Sandia National Laboratories, Aachen, Germany, 1991, 1797-1810.

Munson, Darrell E. "M-D Constitutive Model Parameters Defined for Gulf Coast Domes and Structures." *American Rock Mechanics Association*, Junho 5-9, 2004: 1-6.

N-2752. "Segurança de poço para Projetos de Perfuração de Poços Marítimos." Normas Técnica da Petrobras, Petrobras, Rio de janeiro, 2014, 40.

Okama, Charlton. "Dimensionamento de Colunas: Fundamentos "Teóricos e Procedimentos de Cálculo à luz da Norma N354 ISO / DTR10400"." Pós-Graduação Latu Sensu, Engenharia de Petróleo e Gás, Universidade Petrobras, Salvador, 2009.

Oudeman, P, and L. J. Baccareza. "Field Trial Results of Annular Pressure Bahavior in a High-Pressure / High-Temperature Wells." *Society of Petroleum Engineers*, 1995: 84-88.

Oudeman, P., and M. Kerem. "Transient Bahavior of Anullar Pressure Build-up in HP/HT Wells." *11° Abu Dhabi International Petroleum Exhibition and Conference*, Outubro 10-13, 2004.

Patillo, Phlillip D., Brett W. Cocales, and Stephen C. Morey. "Analysis of an Annular Pressure Buildup Failure During Drill Ahead." *Society of Petroleum Engineers*, Dezembro 04, 2006: 242-247.

Peters, Ekwere J., Martin E. Chenevert, and Chunchal Zhang. "A Model for Predicting the Density of Oil-Based Muds at High Pressures and Temperatures." June 1990: 141-148.

Poiate Jr, Edgard. *Mecânica das Rochas e Mecânica Computacional para Projeto de Poços de Petróleo em Zonas de Sal.* Tese de Doutorado, Departamento de Engenharia Civil, Ponifícia Universidade Católica, Rio de Janeiro: PUC-Rio, 2012.

Poiate, E. Jr, A. M. Costa, and J. L. Falcão. "Well Design for Drilling Through Thick Evaporite Layers in Santos Basin -Brazil." *IADC/SPE Drilling Conference*, Fevereiro 21-23, 2006: 1-16.

Poiate, Edgard Junior, and Claudio dos Santos Amaral. *Sistema Especialista para Análise de Estabilidade de Poços Através de Camadas de Sal.* Pesquisa e Desenvolvimento, Rio de Janeiro: CENPES, 2014, 48.

Romanel, C, E. Poiate, A. M. Costa, and D. M. Roehl. "Creep Constitutive Modeling Applied to the Stability of Pre-Salt Wellbores Through Salt Layers." *ARMA*, June 1-4, 2014: 1-10.

Santos, João Paulo Lima. *Análise de Modelos Reológicos Viscoelásticos através de Formulações Mistas em Elementos Finitos.* Dissertação de Mestrado, Rio de Janeiro: UFRJ, 2008, 123.

Sathuvalli, U. B., L. M. Payne, P. D. Pattillo, S. Rahman, and P. V. Suryanarayana. "Development of a Screening System to Identify Deepwater Wells at Risk for Annular Pressure Build-Up." *SPE/IADC*, Fevereiro 23-25, 2005: 1-14.

Sobolik, Steven R., and Brian L. Ehgartner. *Analysis of Cavern Shapes for the Strategic Petroleum Reserve.* Geoscience and Environment Center, Sandia National Laboratories, Alburquerque: Sandia, 2006.

Timoshenko, S. P., and J. N. Goodier. *Teoria da Elasticidade.* Vol. 3° Edição. Rio de Janeiro, RJ: Guanabara Koogan, 1980.

Vargo, Richard F, Myke Payne, Ronnie Faul, John LeBlanc, and James Griffith. "Practical and Successful Prevention of Annular Pressure Buildup on the Marlin Project." *Society of Petroleum Engineers*, Setembro 29, 2002: 1-10.

Wang, Han Yi, and Samuel Robello. "Geomechanical Modeling of Wellbore Satability in Salt Formations." *Society of Petroleum Engineers*, Setembro 2, 2013: 1-17.

Zambrano, Franklin E.M. "Propriedades Termodinâmicas." *Faculdade de Engenharia Mecânica - UNICAMP*. Agosto 13, 2013. http://www.fem.unicamp.br/~franklin/EM524/em524.html (accessed Marco 18, 2015).

Zamora, Mario, Sanjit Roy, Kenneth Slater, and John Troncoso. "Study on the Volumetric Behavior of Base Oils, Brines, and Drilling Fluids Under Extreme Temperatures an Pressures." *SPE Drilling & Completion*, Setembro 8, 2013: 278-288.

Apêndice A - Evaporitos

Evaporitos são rochas sedimentares que apresentam camadas de minerais de sal, sendo o principal a halita, depositados diretamente de salmouras em condições de forte evaporação e precipitação de bacias de sedimentação restritas, quentes e subsidentes. Tais depósitos de sais podem ser de origem continental ou marinha em que haja aporte periódico de água salgada. A precipitação do sal acontece quando o soluto atinge o ponto de saturação salina daquele componente. Desta maneira a deposição de camadas salinas ocorre em uma sequência ou sucessão de salinização progressiva da bacia de deposição, dos sais menos solúveis para os mais solúveis; por exemplo, gipsita (CaSO4.H2O) e anidrita (CaSO4) nas camadas inferiores, halita ("sal de cozinha" – NaCl), silvita (KCl), carnalita (KCl.MgCl2.6H2O) nas camadas superiores (Botelho 2008).

Uma das principais características do sal é a fluência ou “creep” que é o termo utilizado para descrever a deformação plástica de um material ao longo do tempo em função da aplicação de uma tensão contínua. Esta fluência depende de diversos fatores como composição mineralógica, teor de água, presença de impurezas, tensão diferencial, tempo e temperatura. Sais clorídricos e sulfatados contendo água são mais móveis como carnalita, silvita, taquidrita e bichofita. No caso da halita, que é normalmente a maior composição, a mobilidade é relativamente lenta e da anidrita imóvel.

A moderna investigação do comportamento termodinâmico do sal começou em meados de 1930 com a disciplina mecânica do sal. O estudo evoluiu e foi produtivo no desenvolvimento de modelos constitutivos e investigações em laboratório do comportamento da fluência dependente do tempo. Estes estudos têm sido amplamente utilizados para prever o comportamento de domos salinos ou regiões com acomodações de sal para deposito de lixo radioativo e cavernas para estoque de hidrocarbonetos. O sal é um material interessante, pois tem comportamento de metal e o modelo constitutivo pode ser desenhado como a deformação de um grande corpo metálico para chegar ao modelo com comportamento apropriado. Equipamentos de teste e metodologias têm se

concentrado em ensaios de compressão uniaxial ou triaxial para obtenção de respostas de fluência em estado estacionário ou transiente (Munson and Wawersik 1991).

A descoberta de reservatórios de petróleo em carbonatos selados por camadas de sal, nas bacias do pré-sal brasileiro, gerou a necessidade de pesquisas de metodologias para avaliar a estabilidade do sal e integridade dos poços. Os sais de maior importância nestes poços são a halita, carnalita e taquidrita. A mecânica das rochas aplicadas para o sal iniciou nos anos 1970 no Brasil com a descoberta de reservas de potássio (NaCl.KCl) no estado de Sergipe durante a exploração de petróleo da Petrobras em 1960 (mina Taquari-Vassouras). Uma grande quantidade de informações sobre o tema foi adquirida com pesquisas de instrumentação de campo, aproximações numéricas e pesquisas laboratoriais. Os reservatórios do pré-sal de Santos possuem uma lamina de água que varia de 150 m a 2200 m e atravessam camadas de sal da ordem de 2000 m composta por halita com intercalações de taquidrita e carnalita (Costa, Poiate and Falcão 2010).

A.1. Comportamento de Fluência do Sal

A fluência é a evolução das deformações plásticas com o tempo em condições variáveis de tensão e temperatura e sua velocidade dependerá das características do corpo do material e do maior nível de tensão e temperatura aplicadas. A depender da fase de fluência deve-se ressaltar o fato da possibilidade de mudança da estrutura cristalina com a evolução das deformações, conduzindo a ruptura da macroestrutura do corpo sólido (A. M. Costa 1984).

A fluência do sal, no campo do macro comportamento, é caracterizada por três estágios, com diferentes taxas de deformação em função do tempo para um nível constante de temperatura e tensão. Ao aplicar a tensão há uma deformação elástica pequena que evolui para o primeiro estágio chamado de transiente ou fluência primária. Neste estágio há uma taxa de deformação mais elevada que decresce até uma taxa de deformação uniforme que é o início do segundo estágio ou fluência secundária. Ainda há um terceiro estágio caracterizado pelo fenômeno de dilatação, com incremento do volume através do desenvolvimento de micro fraturas, levando a falha do material (Wang e Robello 2013).

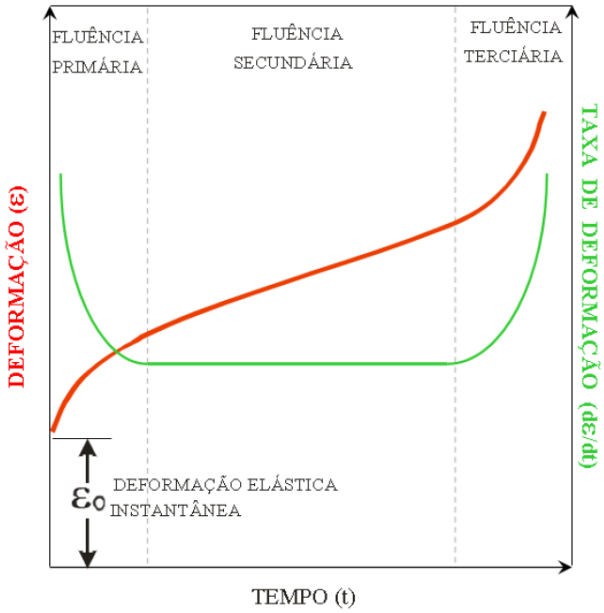

Figura A.1: Típico ensaio de fluência de um evaporito (Poiate Jr 2012).

Além dos três estágios de comportamento citados anteriormente, a recuperação das deformações é outro fenômeno característico de materiais em regime de fluência, porém não será abordado.

Muitos modelos foram elaborados para descrever o comportamento de fluência do sal. Estes podem ser agrupados em três grandes grupos: modelos empíricos, modelos reológicos e modelos físicos (A. M. Costa 1984). A maioria deles é proveniente da ciência dos materiais aplicada a metais.

No começo, os modelos constitutivos tendiam a utilizar o modelo de potência transiente, pois este modelo concordava com observações de campo em minas subterrâneas, que apresentavam decréscimo da taxa de fechamento em função do tempo. Posteriormente, os modelos empíricos e físicos provaram a necessidade de incorporar a fluência em estado estacionário (Munson and Wawersik 1991).

A.2. Modelos Empíricos de Fluência

Foram propostos diversos modelos empíricos para modelar o comportamento físico do sal através de resultados experimentais. Os modelos empíricos têm boa aplicação para fluência transiente (primária), cuja taxa de fluência é decrescente com o tempo e não é completamente entendida. Podemos agrupar estes modelos basicamente em três grupos: potencial, logaritmo e

exponencial. Nestes a taxa de deformação é dependente da tensão, constante elástica, tempo e temperatura, porém com particularidades que as diferenciam. No modelo de potência a taxa de deformação é calibrada por expoentes nos termos citados (eq. (A.1)), no modelo logaritmo a taxa de deformação é proporcional ao logaritmo do tempo (eq. (A.2)) e no modelo exponencial a taxa de deformação é proporcional ao exponencial da temperatura (eq. (A.3)). A seguir é apresentada a forma matemática destes modelos empíricos, que podem ter diferença nas constantes de calibração dependendo do material (A. M. Costa 1984):

Modelo de Empírico de Potência:

$$\varepsilon_t' = A.\sigma^b.t^c.T^d \quad (A.1)$$

Modelo Empírico Logaritmo:

$$\varepsilon_t' = A.\sigma^b.\ln(t).T^d \quad (A.2)$$

Modelo Empírico Exponencial:

$$\varepsilon_t' = A.\sigma^b.t^c.e^{T/d} \quad (A.3)$$

Onde,

$\varepsilon_t' = Taxa\ de\ deformação\ transiente$

$\sigma = tensão\ desviadora$

$t = tempo$

$T = temperatura$

$A, b, c, d, n = constantes\ empíricas\ de\ ajuste\ de\ modelo$

Muito do entendimento do processo de fluência é derivado destas leis empíricas que são ajustadas para descrever os resultados experimentais. Dentre as equações apresentadas a mais utilizada é o modelo de potência por sua simplicidade e bom ajuste.

A.3. Modelos Reológicos de Fluência

Na natureza encontram-se corpos com comportamento elástico ou viscoso. A maioria das rochas se comporta de forma reológica e pode ser matematicamente modelada pela combinação e associação de elementos elásticos com viscosos. Reologia pode ser definida como o estudo do comportamento de deformação e fluência plástica de materiais pelo uso de equipamentos.

Os materiais são classificados como elásticos quando após a aplicação de um carregamento interino e consequente deformação observa-se a recuperação de sua forma original. Neste material a deformação é atingida imediatamente após a aplicação da carga. Os materiais elásticos lineares obedecem à lei de Hooke e a tensão aplicada é diretamente proporcional à deformação do material, sendo a constante de proporcionalidade uma característica intrínseca do material. O comportamento do material linear elástico é representado por uma mola e a relação é então definida por:

$$\sigma_E = E.\varepsilon \tag{A.4}$$

Onde E é a constante de proporcionalidade e ε é a deformação.

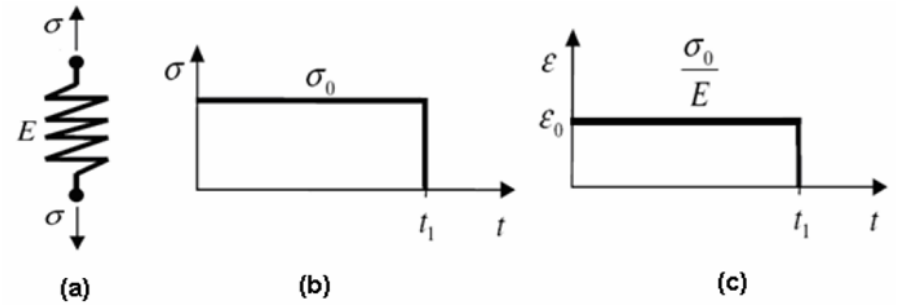

Figura A.2: (a) Elemento de mola; (b) Carga aplicada no elemento mola e (c) Deformação resultante (Santos 2008).

Já os materiais viscosos são definidos como substâncias que se deformam continuamente sob ação de qualquer força tangencial. A resistência que o material oferece ao escoamento é definida como viscosidade e é mais evidente nos fluidos. É dito que o elemento se comporta como um fluido newtoniano se a taxa de deformação é diretamente proporcional à tensão cisalhante aplicada. O comportamento do fluido viscoso é representado por um amortecedor e a relação constitutiva é dada por:

$$\sigma_\eta = \eta.\varepsilon' \tag{A.5}$$

Onde η é a viscosidade e ε’ é a taxa de deformação.

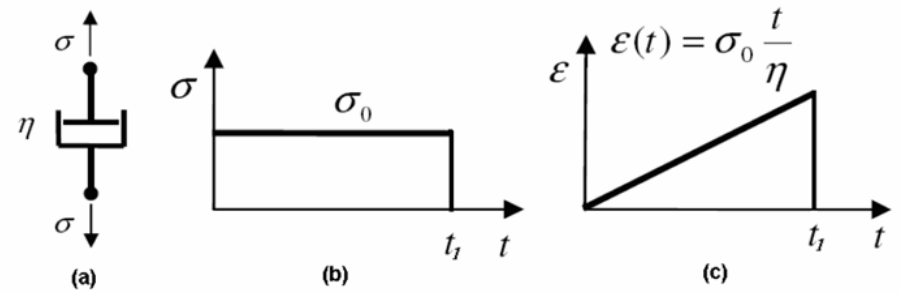

Figura A.3: (a) Elemento viscoso; (b) Carga aplicada no elemento viscoso e (c) Deformação resultante (Santos 2008).

Diversos modelos foram desenvolvidos, com a finalidade de prever o comportamento destes materiais, através da combinação e associação destes elementos e deram origem aos modelos de Maxwell, de Kelvin, de Burgers e outros. Os modelos reológicos, apresentados a seguir, são utilizados para prever o comportamento reológico de materiais visco-elásticos em função do tempo para carregamento uniaxial.

A.3.1. Modelo de Maxwell

O modelo de Maxwell resulta na combinação de um elemento elástico e de um elemento viscoso disposto em série. Este modelo é utilizado para estudar o comportamento secundário da fluência, ou seja, o estado estacionário. A resolução da equação diferencial resultante da associação em série do elemento mola e amortecedor do modelo de Maxwell, para uma tensão inicial σ_0 em um tempo inicial t_0, é dado pela equação a seguir:

$$\varepsilon(t) = {}^{\sigma_0}/_{E} + {}^{\sigma_0}/_{\eta} \cdot t \qquad \text{(A.6)}$$

Onde,

$\varepsilon = deformação$

$\sigma_0 = tensão\ desviadora\ no\ tempo\ t = 0$

$E = representa\ o\ módulo\ de\ elasticidade$

$\eta = viscosidade$

$t = tempo$

Quando um corpo desta natureza é submetido a uma carga o corpo se deforma instantaneamente o valor correspondente à deformação elástica e, em

seguida, inicia uma deformação proporcional ao tempo devido ao elemento viscoso.

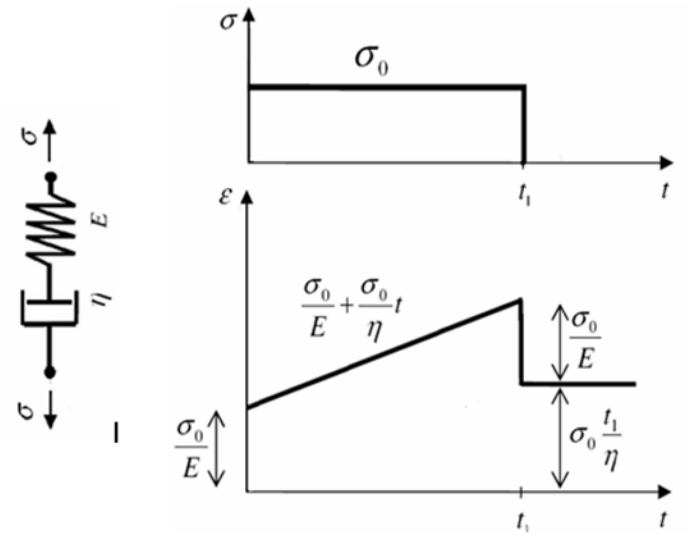

Figura A.4: Elemento de Maxwell, carga aplicada e deformação resultante (Santos 2008).

No instante t_1, em que a carga é removida, nota-se o fenômeno de reversibilidade da fluência. Dessa forma o material recupera o estado de deformação sofrido pela parcela elástica e atinge uma deformação menor equivalente a deformação viscosa. Durante a aplicação da tensão a deformação total é dada pela soma das deformações dos elementos de Maxwell.

A.3.2. Modelo de Kelvin

O modelo reológico de Kelvin é composto por um elemento elástico em paralelo com um elemento viscoso. Quando um corpo desta natureza é submetido a uma carga o elemento viscoso retarda a deformação elástica e quando a carga é retirada ocorre uma recuperação da forma inicial também dependente do tempo. Cada elemento suporta uma parcela da tensão sendo a deformação total a mesma para os dois elementos em cada instante. A parcela de tensão que cada elemento suporta é definida pela igualdade das deformações dos elementos em paralelo. A resolução da equação diferencial resultante da associação em paralelo do elemento mola e amortecedor do modelo de Kelvin, para uma deformação inicial $\varepsilon = 0$ em $t = 0$, tem a seguinte forma:

$$\varepsilon(t) = {}^{\sigma_0}/_{E} \cdot \left(1 - e^{-\frac{E.t}{\eta}}\right) \quad \text{(A.7)}$$

Onde,

$\varepsilon = deformação$

$\sigma_0 = tensão\ desviadora\ no\ tempo\ t = 0$

$E = representa\ o\ módulo\ de\ elasticidade$

$\eta = viscosidade$

$t = tempo$

Analisando a equação do modelo de Kelvin observa-se uma deformação instantânea nula. Mantido o carregamento a deformação tende a um valor assintótico e com a remoção do carregamento a deformação decai exponencialmente até a deformação nula como se pode observar na figura a seguir.

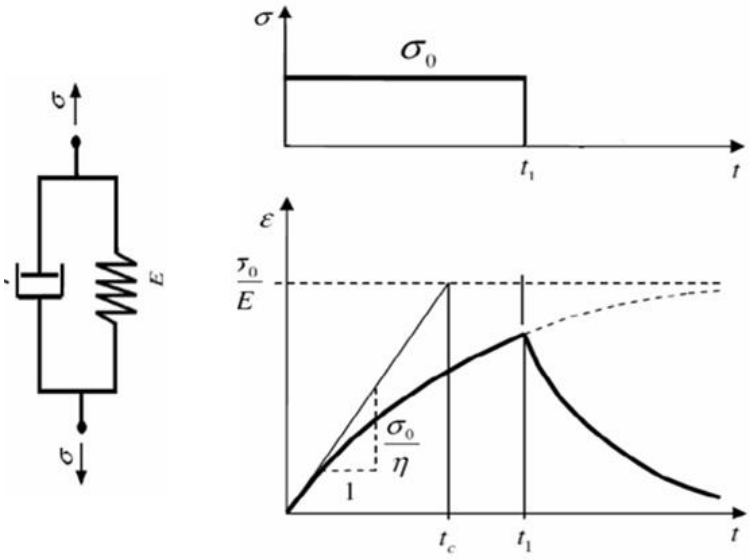

Figura A.5: Elemento de Kelvin, carga aplicada e deformação resultante (Santos 2008).

A.3.3. Modelo de Burgers

O modelo proposto por Burgers utiliza-se da associação em série do elemento de Kelvin e do elemento de Maxwell. Neste sistema a deformação total do sistema é a soma das deformações do modelo de Maxwell (ε_1) e de kelvin (ε_2),

pois estão acoplados em série. A associação em série neste modelo resulta em uma equação diferencial de segunda ordem. Considerando-se a tensão constante ($\sigma = \sigma_0$) a equação diferencial torna-se mais simples e a solução da mesma é expressa por:

$$\varepsilon(t) = {}^{\sigma_0}/_{E_2} + {}^{\sigma_0}/_{E_1} \cdot \left(1 - e^{-\frac{E_1 \cdot t}{\eta_1}}\right) + {}^{\sigma_0}/_{\eta_2} \cdot t \qquad \text{(A.8)}$$

Onde,

$\varepsilon = deformação$

$\sigma_0 = tensão\ desviadora\ no\ tempo\ t = 0$

$E_1 = módulo\ de\ elasticidade\ do\ elemento\ de\ Maxwell$

$E_2 = módulo\ de\ elasticidade\ do\ elemento\ de\ Kelvin$

$\eta = viscosidade$

$t = tempo$

Este modelo consegue prever a deformação elástica inicial (σ_0 / E_2), simular a deformação na fase transiente de fluência e, ainda, a deformação da fase de fluência secundária com velocidade de deformação constante σ_0 / η_2.

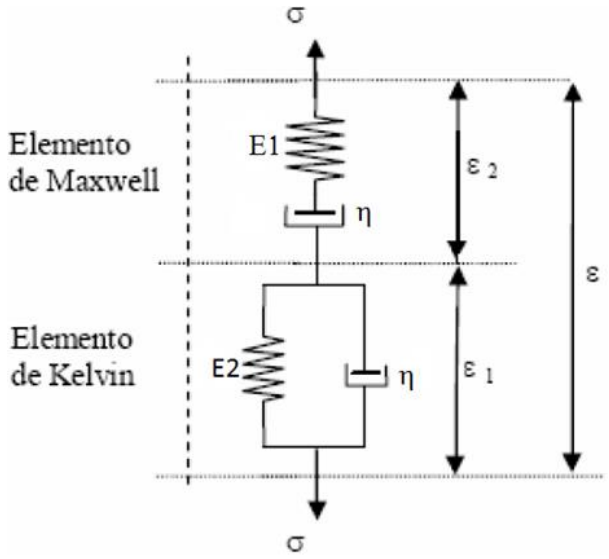

Figura A.6: Modelo de Burgers com elemento de Maxwell e elemento de Kelvin em série (Botelho 2008).

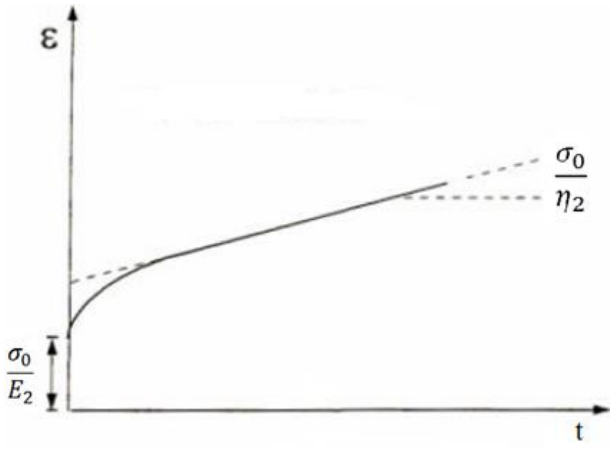

Figura A.7: Deformação resultante da aplicação da carga σ_0 no elemento de Burgers (Botelho 2008).

A.4. Modelos Físicos de Fluência

O melhor conhecimento do comportamento de fluência gerou uma sofisticação dos modelos dos evaporitos que passaram a considerar intervalo de tensão, estado de deformação, taxa de deformação, temperatura e microestrutura.

Segundo Munson (Munson and Wawersik 1991) foram elaborados diversos mapas independentes com mecanismos de deformação a fim de verificar o que a equação constitutiva deveria considerar. O mapa elaborado por Munson enumera 5 mecanismos de deformação secundária dependentes da tensão, temperatura e modulo de cisalhamento, onde o estado estacionário de cada domínio é dominado por um único mecanismo. Estes mecanismos físicos, que podem ser visualizados na figura A.8, são: (1) fluência sem defeito ("defect-less flow"), (2) discordâncias por deslizamento ("dislocation glide"), (3) discordâncias por escalonamento ("dislocation climb creep"), (4) difusão de massa ("diffusion") e (5) mecanismo indefinido ("undefined mechanism). A maior ou menor contribuição de cada mecanismo depende da temperatura e tensão desviatória que o evaporito está sujeito. Os dois regimes de alta tensão (discordância por deslizamento e sem defeito) são governados pelos processos de fluência e os três regimes restantes (difusão, discordância por escalonamento e mecanismo indefinido) são governados por processos de equilíbrio termicamente ativados.

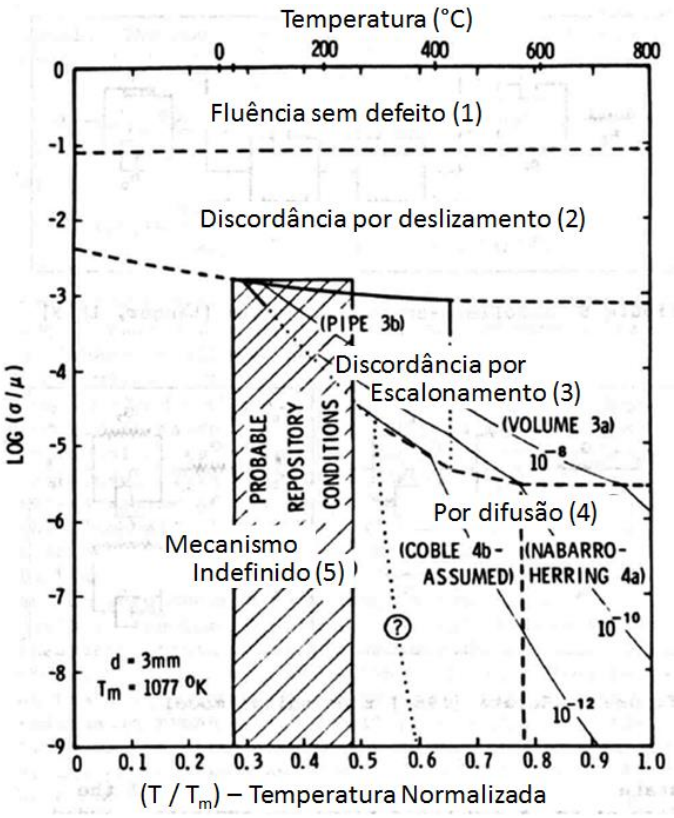

Figura A.8: Mapa de mecanismo de deformação adaptado (Munson and Wawersik 1991).

No mapa de Munson, na figura A.8, o eixo horizontal expressa a temperatura como uma fração do ponto de fusão do sal na escala absoluta Kelvin T / T_m. O eixo vertical é o logaritmo da tensão pelo módulo de cisalhamento μ.

A.4.1. Discordância por Escalonamento

O modelo "dislocation climb" recebe este nome devido ao comportamento de deformação. Ocorre em elevadas temperaturas e baixas tensões. Este modelo é governado pelo fenômeno de ativação térmica, que ocorre quando um incremento de temperatura causa um movimento atômico com redistribuição molecular na estrutura do material.

$$\varepsilon'_{s_1} = A_1 \left(\frac{\sigma}{\mu}\right)^{n1} e^{-\frac{Q_1}{RT}} \tag{A.9}$$

Onde,

ε'_{s_1} = taxa de $fluência - dislocation\ climb$

$A_1 = constante\ do\ material$

$\sigma = tensão\ desviatória$

μ = $módulo\ de\ cisalhamento\ do\ material$

$Q_1 = energia\ de\ ativação$

$R = constante\ universal\ dos\ gases$

$T = temperatura\ absoluta$, [K]

$n_1 = expoente\ de\ tensão$

A.4.2. Discordância por Deslizamento

O modelo "dislocation glide" é conhecido pelo deslizamento de planos adjacentes em um material que está submetido a elevados níveis de tensão. O modelo é representado por uma função do seno hiperbólico da tensão diferencial.

$$\varepsilon'_{s_2} = |H| \left[B_1 e^{-\frac{Q_1}{RT}} + B_2 e^{-\frac{Q_2}{RT}} \right] sinh \left[q \frac{(\sigma - \sigma_0)}{\mu} \right] \quad \text{(A.10)}$$

Onde,

$\varepsilon'_{s_2} = taxa\ de\ fluência - dislocation\ glide$

$H = função\ Heaviside\ com\ o\ argumento\ (\sigma - \sigma_0)$

$B_1, B_2 = constante\ de\ ajuste\ do\ material$

$Q_1, Q_2 = energia\ de\ ativação$

$R = constante\ universal\ dos\ gases$

$T = temperatura\ absoluta$, [K]

$\sigma_0 = tensão$ de referência

$\sigma = tensão\ desviatória$

μ = $módulo\ de\ cisalhamento\ do\ material$

$q = constante\ de\ tensão$

A.4.3. Mecanismo Indefinido

O "mecanismo indefinido" recebe este nome por não ter nenhum mecanismo micromecânico associado, contudo foi modelado empiricamente. Os experimentos realizados apresentaram o mesmo comportamento da função "dislocation climb" para baixa temperatura e baixo nível de tensão.

$$\varepsilon'_{s_3} = A_2 \left(\frac{\sigma}{\mu}\right)^{n_2} e^{-\frac{Q_2}{RT}} \quad \text{(A.11)}$$

Onde,

ε'_{s_3} = taxa de $fluência - mecanismo\ indefinido$

$A_2 = constante\ do\ material$

$\sigma = tensão\ desviatória$

μ = módulo de cisalhamento do material

$Q_2 = energia\ de\ ativação$

$R = constante\ universal\ dos\ gases$

$T = temperatura\ absoluta$, [K]

$n_2 = expoente\ de\ tensão$

A.4.4. Modelo de Multimecanismo

O modelo de multimecanismo (MD) é um modelo sofisticado que simula a fluência do sal no estado transiente e incorpora o comportamento em estado estacionário (D. E. Munson 2004). Este modelo resulta de um programa de pesquisa e testes de larga escala do departamento de energia dos Estados Unidos (DOE) para planta piloto de isolamento de resíduo (WIPP). O modelo de multimecanismo é baseado na superposição de três mecanismos micromecânicos de fluência em estado estacionário: "dislocation climb" (eq. (A.9)), "dislocation glide" (eq. (A.10)) e mecanismo indefinido (eq. (A.11)). O modelo transiente é estimado através de um ajuste no modelo estacionário utilizando uma função transiente com evolução de uma variável interna e parâmetros de ajuste (Romanel, et al. 2014).

A taxa de fluência é dada por:

$$\dot{\varepsilon} = \left(A_1 \left(\frac{\sigma}{\mu}\right)^{n1} e^{-\frac{Q_1}{RT}} + |H| \left[B_1 e^{-\frac{Q_1}{RT}} + B_2 e^{-\frac{Q_2}{RT}}\right] \sinh\left[q \frac{(\sigma - \sigma_0)}{\mu}\right] + A_2 \left(\frac{\sigma}{\mu}\right)^{n_2} e^{-\frac{Q_2}{RT}}\right) \cdot F(\sigma) \quad \text{(A.12)}$$

Onde o primeiro, segundo e terceiro termos são relacionados aos mecanismos "dislocation climb", "dislocation glide" e mecanismo indefinido, respectivamente. As constantes A_1, Q_1 e n_1 são, respectivamente, fator estrutural,

energia de ativação e expoente de tensão do mecanismo "climb". Da mesma maneira as constantes A_2, Q_2 e n_2 são fator estrutural, energia de ativação e expoente de tensão do mecanismo indefinido. As constantes B_1 e B_2 são fatores estruturais do mecanismo "glide". A função degrau Heaviside $H[\sigma_{eq} - \sigma_0]$ com o argumento $(\sigma_{eq} - \sigma_0)$ limita a contribuição do mecanismo glide a uma tensão mínima σ_0. O parâmetro q é uma constante de tensão do mecanismo "glide" e μ o módulo de cisalhamento do material. A função F descreve essencialmente a parcela transiente, ou seja, a curvatura da resposta de fluência. Este multiplicador consiste em uma função cinética de ordem elevada que se divide em duas funções:

$$F = \begin{Bmatrix} \exp\left[\delta\left(1 - {\xi}/{\varepsilon_t^*}\right)^2, \xi \geq \varepsilon_t^*\right] \\ \exp\left[-\Delta\left(1 - {\xi}/{\varepsilon_t^*}\right)^2, \xi \leq \varepsilon_t^*\right] \end{Bmatrix} \quad \text{(A.13)}$$

As duas opções de F tem a curvatura definida por: δ como parâmetro "workhardening" e Δ como parâmetro de recuperação. A variável ξ é um parâmetro de estado. A evolução da variável interna (equação cinética) é dada por:

$$\dot{\xi} = (F - 1) \cdot \dot{\varepsilon_s} \quad \text{(A.14)}$$

O tipo de fluência é determinado pela variável ε_t^* que é definida como a intersecção da fluência em estado estacionário no eixo das ordenadas, para uma curva deformação x tempo, e é dado pela equação:

$$\varepsilon_t^* = K_0 e^{cT} \left(\frac{\sigma_d}{\mu}\right)^m \quad \text{(A.15)}$$

Onde K_0 e c são constantes e m é uma constante teórica.

A.5. Critérios de Falha

Os maciços rochosos são submetidos a um conjunto de solicitações durante a história geológica que estabelecem uma condição de equilíbrio. Desse conjunto de solicitações pode-se citar o peso da coluna litostática, os esforços tectônicos e outros que estabelecem as condições iniciais de um estado de tensões e campo de deslocamentos (A. M. Costa 1984).

No estudo de plasticidade e critérios de falha é importante o entendimento de invariantes de tensão, grandezas escalares que independem da orientação de

um determinado estado de tensões. A definição de invariante de tensões pode ser realizada a partir do equilíbrio de forças em planos principais, isto é, planos onde a tensão cisalhante é nula e as tensões normais adquirem seus valores máximos e mínimos. Por sua vez o estado hidrostático deve provocar apenas mudança de volume de material (Okama 2009). Quando a condição de equilíbrio é retirada pode ocorrer o escoamento da rocha ou falha da mesma a partir do novo estado de tensões.

Munson (Munson and Wawersik 1991) afirma que o sal segue o máximo potencial de cisalhamento ao invés do potencial de cisalhamento octaédrico. Logo, Tresca seria preferível ao invés de von Mises para mensurar a tensão desviatória. Para materiais com comportamento dúctil são empregados os critérios de escoamento de von Mises e Tresca. Para materiais frágeis o critério de Mohr-Coulomb é mais utilizado para caracterizar o tipo de ruptura associado ao material. Este tema, quando aplicado ao sal, gera bastante discussão.

Com o objetivo de contextualizar os critérios de falha de von Mises, Dilatância, Mohr-Coulomb e Tresca são apresentados na sequência um passo a passo até a obtenção dos invariantes de tensão. Estas servem como base para o entendimento dos critérios de falha. Define-se primeiramente o versor do plano de interesse:

$$n^t = \{n_x \quad n_y \quad n_z\} \tag{A.16}$$

Logo, considerando equilíbrio rotacional (tensões Tangenciais simétricas iguais – Teorema de Cauchy) pode-se expressar o tensor das tensões totais para o plano de interesse na seguinte forma matricial:

$$\begin{bmatrix} \sigma_x - \sigma & \tau_{yx} & \tau_{zx} \\ \tau_{xy} & \sigma_y - \sigma & \tau_{zy} \\ \tau_{xz} & \tau_{yz} & \sigma_z - \sigma \end{bmatrix} \begin{Bmatrix} n_x \\ n_y \\ n_z \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{A.17}$$

O sistema linear da eq. (A.17) tem solução não trivial se o determinante da matriz dos coeficientes for nulo. Com isso o cálculo do determinante com alguma manipulação algébrica leva a uma equação de terceiro grau (equação característica) dado por:

$$\sigma^3 - I_1 \cdot \sigma^2 + I_2 \cdot \sigma - I_3 = 0 \tag{A.18}$$

Onde:

$$I_1 = \sigma_x + \sigma_y + \sigma_z \tag{A.19}$$

$$I_2 = \sigma_x\sigma_y + \sigma_y\sigma_z + \sigma_z\sigma_x - \tau_{xy}^2 - \tau_{yz}^2 - \tau_{xz}^2 \tag{A.20}$$

$$I_3 = \sigma_x\sigma_y\sigma_z + \sigma_x\,\tau_{yz}^2 + \sigma_y\tau_{xz}^2 + \sigma_z\tau_{xy}^2 + 2\tau_{xy}\tau_{yz}\tau_{zx} \tag{A.21}$$

As raízes do polinômio cúbico da eq. (A.18) são reais e representam as tensões principais σ_1, σ_2 e σ_3, tal que $\sigma_1 \geq \sigma_2 \geq \sigma_3$. Uma vez que as raízes do polinômio determinam as tensões principais, as grandezas I_1, I_2 e I_3 definidas nas eq. (A.19), (A.20) e (A.21) devem ser invariantes, isto é, uma rotação dos eixos coordenados não altera seus valores. Por isso são definidos como invariantes do estado de tensão.

O estado de tensão total pode ser decomposto em um tensor de tensão hidrostático e um tensor de tensão desviadora, segundo eq. (A.22).

$$\begin{bmatrix} \sigma_x & \tau_{yx} & \tau_{zx} \\ \tau_{xy} & \sigma_y & \tau_{zy} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} \equiv \begin{bmatrix} \sigma_m & 0 & 0 \\ 0 & \sigma_m & 0 \\ 0 & 0 & \sigma_m \end{bmatrix} + \begin{bmatrix} \sigma_x - \sigma_m & \tau'_{yx} & \tau'_{zx} \\ \tau'_{xy} & \sigma_y - \sigma_m & \tau'_{zy} \\ \tau'_{xz} & \tau'_{yz} & \sigma_z - \sigma_m \end{bmatrix} \tag{A.22}$$

A figura A.9 equivale a eq. (A.22) e ilustra mais claramente o estado de tensão total decomposto em estado hidrostático e estado de tensão desviatória.

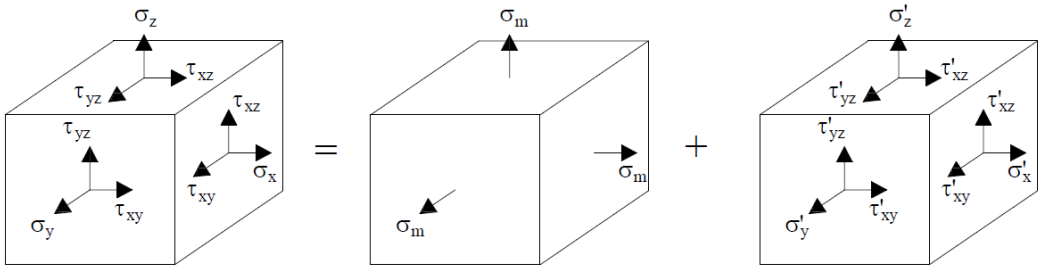

Figura A.9: Estado de tensão total decomposto em estado de tensão hidrostático e estado de tensão desviatória.

A relação entre as tensões hidrostáticas e desviatórias pode ser obtida considerando-se que as tensões desviatórias σ_x', σ_y' e σ_z' podem ser obtidas a partir do estado original de tensões σ_x, σ_y e σ_z subtraindo-se o estado de tensão hidrostático, isto é:

$$\begin{aligned} \sigma'_x &= \sigma_x - \sigma_m \\ \sigma'_y &= \sigma_y - \sigma_m \\ \sigma'_z &= \sigma_z - \sigma_m \end{aligned} \tag{A.23}$$

O estado desviatório tem por característica não provocar nenhuma variação de volume do material, apenas alteração da forma, ou seja:

$$\sigma'_x + \sigma'_y + \sigma'_z = 0 \tag{A.24}$$

Por sua vez, a parcela hidrostática provoca apenas mudança de volume no material, o que permite calcular um estado de tensão equivalente σ_m segundo três eixos perpendiculares com tensões cisalhantes nulas. Levando-se a eq. (A.23) na eq. (A.24) obtém-se a tensão média octaédrica, que pode ser relacionado com o primeiro invariante de tensões:

$$\sigma_m = {}^1\!/_3\,(\sigma_x + \sigma_y + \sigma_z) = {}^{I_1}\!/_3 \tag{A.25}$$

E portando, substituindo a eq. (A.24) na eq. (A.23) temos a seguinte relação para a parcela desviatória:

$$\begin{aligned} \sigma'_x &= (2\sigma_x - \sigma_y - \sigma_z)/3 \\ \sigma'_y &= (2\sigma_y - \sigma_x - \sigma_z)/3 \\ \sigma'_z &= (2\sigma_z - \sigma_x - \sigma_y)/3 \end{aligned} \tag{A.26}$$

Uma vez que no estado hidrostático não há tensões de cisalhamento, as tensões cisalhantes do tensor desviatório devem ser as mesmas do estado de tensões inicial, segundo a eq. (A.27):

$$\begin{aligned} \tau'_{xy} &= \tau_{xy} \\ \tau'_{yz} &= \tau_{yz} \\ \tau'_{xz} &= \tau_{xz} \end{aligned} \tag{A.27}$$

Substituindo as tensões desviatórias, eq. (A.26) e (A.27), na eq. (A.19), (A.20) e (A.21), obtém-se os chamados os invariantes de tensão do tensor desviatório:

$$J_1 = \sigma_x + \sigma_y + \sigma_z - 3.\sigma_m = 0 \tag{A.28}$$

$$\begin{aligned} J_2 = \frac{1}{2}\left[\frac{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2}{3}\right] \\ + (\tau_{xy}{}^2 + \tau_{yz}{}^2 + \tau_{zx}{}^2) = I_1^2/3 - I_2 \end{aligned} \tag{A.29}$$

$$J_3 = I_3 - I_2\sigma_m + 2\sigma_m^3 \tag{A.30}$$

A.5.1.
Critério de Von Mises

Partindo de observações empíricas von Mises (1913) concluiu que a falha de um material ocorria preferencialmente em virtude da distorção, ou seja, o material começa a se deformar plasticamente quando o segundo invariante de tensões desviadoras J_2 (eq. A.29) alcança o valor crítico.

Consequentemente o valor crítico da energia de distorção é dito igual à energia de distorção do material quando este é submetido a um estado uniaxial de tensões. Supondo a aplicação da força no eixo x, temos $\sigma_x = \sigma_{eq}$ e as demais tensões nulas. Aplicando esta condição na eq. (A.29) temos:

$$J_2 = \frac{1}{6} \cdot \left[\left(\sigma_{eq} - 0\right)^2 + (0 - 0)^2 + \left(0 - \sigma_{eq}\right)^2 \right] + (0^2 + 0^2 + 0^2) \qquad (A.31)$$

$$J_2 = \frac{{\sigma_{eq}}^2}{3} \qquad (A.32)$$

Onde,

$J_2 =$ Segundo invariante de tensões desviatórias

$\sigma_{eq} =$ tensão de escoamento do estado uniaxial de tensões

Igualando-se a expressão do invariante de tensões desviadoras J_2 resultante da aplicação de uma tensão uniaxial (eq. A.32) com a eq. (A.29) que expressa o segundo invariante de tensão desviatório chega-se a expressão para tensão de von Mises. Este critério é muito utilizado para estudos e testes de comportamento mecânico porque é composto de uma única expressão.

$$\sigma_{VM} = \frac{1}{\sqrt{2}} \left[\left(\sigma_x - \sigma_y\right)^2 + \left(\sigma_y - \sigma_z\right)^2 + (\sigma_z - \sigma_x)^2 + 6.\left({\tau_{xy}}^2 + {\tau_{yz}}^2 + {\tau_{zx}}^2\right) \right]^{1/2} \qquad (A.33)$$

Sais policristalinos exibem comportamento de deformação similar à deformação de rochas em baixa temperatura e moderada tensão e similar a metais em elevada temperatura e elevada tensão.

A.5.2.
Critério de Dilatância

Dilatância é um critério que considera o aparecimento de danos na rocha resultando em um significante incremento de permeabilidade. No sal a dilatância ocorre tipicamente quando a rocha atinge seu volume mínimo, ou limite de dilatância, no qual o microfraturamento da rocha incrementa seu volume.

Van Sambeek (Hansen, et al. 1993) define a dilatância como função linear do invariante de tensão I_1, apresentado na eq. (A.19), e a raiz quadrada do segundo invariante de tensão desviatória J_2, apresentado na eq. (A.29), baseado em um conjunto de testes de laboratório na WIPP (Waste Isolation Pilot Plant), SPR e outras amostras de sal. Deste modo ele propôs que:

$$\sqrt{J_2} = 0{,}27.\, I_1 \tag{A.34}$$

Em consequência estabelece-se um fator de segurança (razão da compressão triaxial) para definir quando ocorre o dano de dilatância:

$$SF_{VS} = \frac{0{,}27.\, I_1}{\sqrt{J_2}} \tag{A.35}$$

Van Sambeek estabelece que quando $SF_{VS} < 1$ há indicação de dano por dilatância e que quando $SF_{VS} < 0{,}6$ há indicação de falha (Sobolik and Ehgartner 2006).

A.5.3.
Critério de Mohr-Coulomb

O critério de ruptura Mohr Coulomb (1835-1918) é amplamente utilizado como critério de fratura em materiais rochosos que apresentam comportamento frágil. Este critério relaciona a falha por fricção devido à tensão normal e tensão cisalhante.

O círculo de Mohr representa as tensões em um círculo através da tensão principal σ_1 e a menor σ_3. O círculo é utilizado para encontrar as tensões σ (normal) e τ (cisalhante) em qualquer ponto do círculo. O ponto A representa 2θ para a tensão principal.

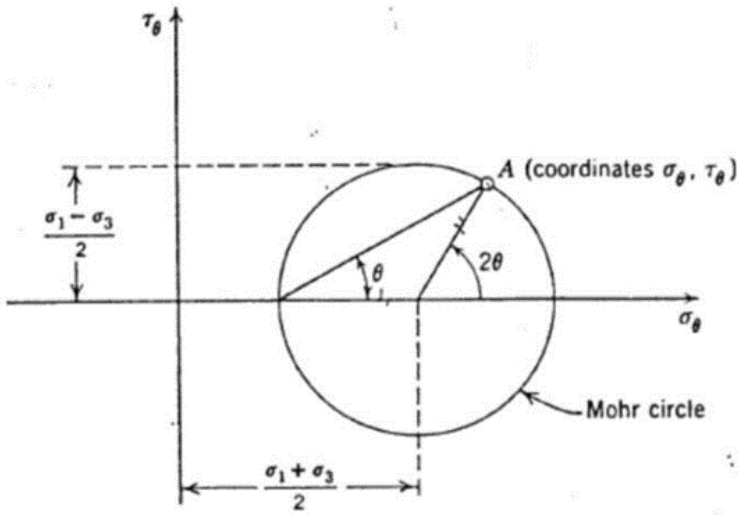

Figura A.10: Círculo de Mohr (Grainger 2012).

Pelo conhecimento da tensão principal e suas direções o círculo de Mohr (figura A.10) facilita a determinação do estado plano em um material contínuo.

Centro do círculo de Mohr:

$$\sigma_m = \frac{\sigma_1 + \sigma_3}{2} \tag{A.36}$$

Raio do círculo de Mohr:

$$R = \frac{\sigma_1 - \sigma_3}{2} = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + {\tau_{xy}}^2} \tag{A.37}$$

E as tensões principais são dadas por:

$$\sigma_{1,3} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + {\tau_{xy}}^2} \tag{A.38}$$

Em experimentos laboratoriais realizam-se diversos testes de falha triaxial para diferentes tensões de confinamento. O ângulo entre o envelope linear e o eixo horizontal representa o ângulo interno de fricção em que cada falha ocorre.

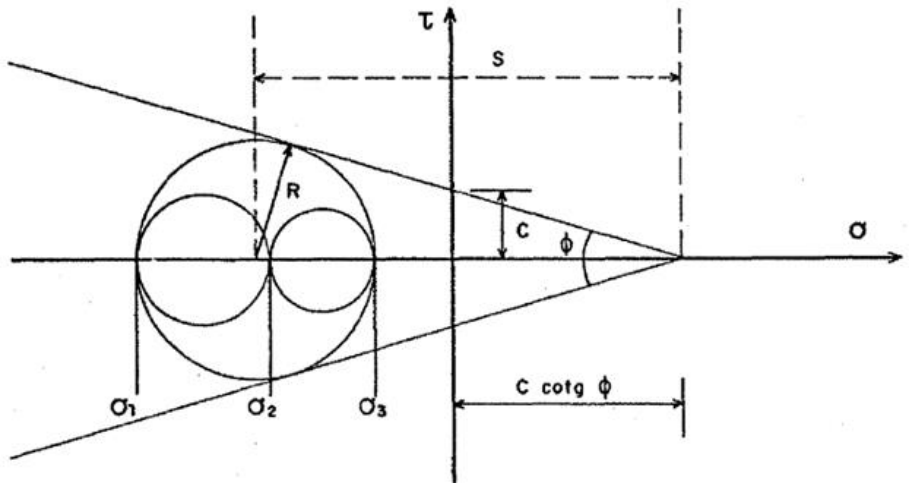

Figura A.11: Critério de falha de Mohr-Coulomb com tensão de corte (A. M. Costa 1984).

Da análise do círculo de Mohr verifica-se que a máxima tensão de cisalhamento ocorre em planos que formam 45° com os planos principais e é igual à metade da diferença entre a tensão máxima e tensão mínima. A extensão da reta até o eixo das ordenadas fornece a força coesiva (c). O ponto em que a reta do envelope é tangente com o círculo é a tensão normal (σ) e a tensão no momento da falha correspondem à tensão de cisalhamento (τ) na falha.

$$\tau = c + \sigma.\tan(\phi) \qquad \text{(A.39)}$$

Onde,

$\tau =$ tensão de cisalhamento para falha

$c =$ Resistência coesiva do material

$\sigma =$ Tensão normal no plano de falha

$\phi =$ angulo de atrito interno

A.5.4. Critério de Tresca

O critério de falha de Tresca é baseado em metais e estabelece que o escoamento do material é provocado pela máxima tensão de cisalhamento que age em um plano de 45° em relação à tensão normal principal. Também é conhecido como critério da máxima tensão de cisalhamento.

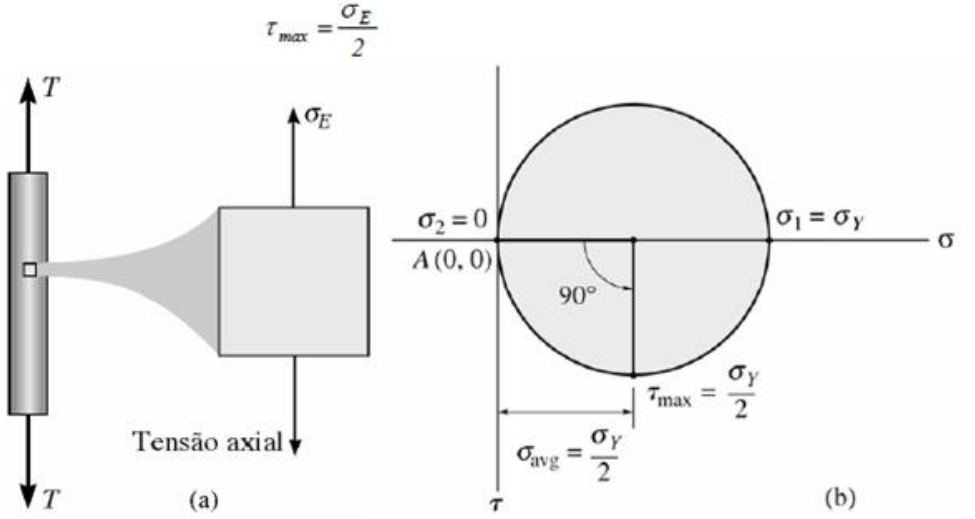

Figura A.12: (a) Elemento de corpo de prova e (b) Círculo de Mohr para condição de Tresca.

$$\tau_{max} = \text{Max}\left(\left[\frac{\sigma_1 - \sigma_2}{2}\right], \left[\frac{\sigma_2 - \sigma_3}{2}\right], \left[\frac{\sigma_1 - \sigma_3}{2}\right]\right) \quad \text{(A.40)}$$

Apêndice B – Implementação do Código APBsal

A seguir são apresentadas as rotinas implementadas no código APBsal para calcular o APB do sal através da rotina APBsal.

B.1. Geometria

```
function [a, b, topo, base, n, cc, af] = Geometria()
% cc = 1 significa poço com anular cimentado;
% cc = 2 significa poço com fluido livre no anular;
% s = indida o numero da seção avaliada
% n = indica o numero de anulares considerados
% i = indica o revestimento / anular avaliado
% a = raio interno [m]
% b = raio externo [m]
i = [1, 2, 3, 4, 5, 6];
di = [5.791, 9.156, 12.375, 18.00, 26].*0.0254; %[m]
do = [6.625, 10.75, 13.625, 20, 1100].*0.0254; %[m]
a = di/2;
b = do/2;
%% Seção 1:
   n(1) = 3;
   topo(1) = 1800;
   base(1) = 2800;
   cc(1) = 1; % cimento atras do último revestimento;
   af(1) = 26./2.*0.0254; %[m]
%% Seção 2
   n(2) = 2;
   topo(2) = 2800;
   base(2) = 4150;
   cc(2) = 1; % cimento atras do último revestimento;
   af(2) = 17.5./2.*0.0254; %[m]
```

```
%% Seção 3 -> Está considerando o anular com cimento
    n(3) = 2; %Para CC(2) deve-se incluir um anular do poço aberto com fluido para análise de APB
    topo(3) = 4150;
    base(3) = 5000;
    cc(3) = 2; % poço aberto na seção (2);
    af(3) = 14.75./2.*0.0254; %[m]
%% Seção 4
    n(4) = 1;
    topo(4) = 5000;
    base(4) = 5400;
    cc(4) = 1; % cimento atras do último revestimento;
    af(4) = 14.75./2.*0.0254; %[m]
```

B.2. Temperatura

```
function [T1, T2, T2R, T2_col, Tg] = Temperatura(topo, base)
dk = 1; % Perfil ajustado para o tubing
   %% Perfis para condição inicial
   for i = 1:1:4 % Perfil na superfície,
      for k = 1
         Tg(k) = 20;
         T1(i,k) = 20;
         T2(i,k) = 20;
         T2R(i,k) = 20;
         T2_col(k) = 20;
      end
      % Perfil de temperatura no mar até 400 m.
      for k = 2:1:400
         Tg(k)= -2.67*log(k) + 20;
         T1(i,k)= -2.67*log(k) + 20;
         T2(i,k)= -2.67*log(k) + 20;
         T2R(i,k) = -2.67*log(k) + 20;
         T2_col(k) = -2.67*log(k) + 20;
      end
      % Perfil de temperatura no mar até a cabeça do poço.
      for k = 401:1:1799
         Tg(k) = 4;
         T1(i,k) = 4;
         T2(i,k) = 4;
         T2R(i,k) = 4;
         T2_col(k) = 4;
      end
      % Perfil geotérmico do poço.
      for k = 1800:1:5400
         Tg(k) = 103.22.*log(k) - 769.09; % Perfil geotérmico;
      end
```

```
    end
    %% Perfil de produção para 5.000 bbl;
    for s = 1:1:4
       for k = topo(s):dk:base(s)
          T1(1,k) = 103.22.*log(k) - 769.09; % Perfil geotérmico;
          T1(2,k) = 103.22.*log(k) - 769.09; % Perfil geotérmico;
          T1(3,k) = 103.22.*log(k) - 769.09; % Perfil geotérmico;
          T1(4,k) = 103.22.*log(k) - 769.09; % Perfil geotérmico;
    %% Perfis para produção de 5000 bb/dia
          % Perfil de temperatura para os fluidos dos anulares na condição 2
          T2(1,k) = 27.825*log(k) - 115.12;
          T2(2,k) = 39.016*log(k) - 212.18;
          T2(3,k) = 53.008*log(k) - 329.65;
          T2(4,k) = 53.008*log(k) - 329.65;
          % Perfil de temperatura para COP condição 2.
          T2_col(k) = 16.149*log(k) - 12.9;
          % Perfil de temperatura para os revestimentos na condição 2.
          T2R(1,k) = 18.008*log(k) - 30.263;
          T2R(2,k) = 33.828*log(k) - 167.48;
          T2R(3,k) = 44.408*log(k) - 257.94;
          T2R(4,k) = 57.143*log(k) - 365.45;
          % Indicador de profundidade.
          z(k) = k;
       end
    end
end
```

B.3.
PVT1

```
function [rho_o1, rho_w1, rho_f1, P1, P1_HC1, m_0, mT_0, Va_fl_0, VaT_fl_0] = PVT1(topo, base, a, af, cc, b, n, T1, Tg)

%%  Pressão deve ser em [psi] e temperatura em [ºF] para o modelo de PVT do Zamora
dk = 1; % comprimento do elemento de cálculo
PF = 5400; % profundidade final para cálculo da pressão do reservatório
T1 = (T1)*9/5 + 32; % Converte [ºC] --> [ºF] para utilizar nas equações de massa específica
Tg = (Tg)*9/5 + 32;
d_f = 1e-6; % Tolerância para convergência da massa específica;
%% Composição do fluido sintético utilizado nos anulares
for i = 1:4 %Mesma composição para todos os anulares
   fw(i) = 0.20; % Fração de brine
   fc(i) = 0.01; % Fração de quimicos
   fs(i) = 0.165; % Fração de sólidos
   fo(i) = 1 - fw(i) - fc(i) - fs(i); % Fração de parafina
   % Massa específica da baritina e produtos químicos:
   rho_s = 8.33*4.3; % Massa específica dos sólidos em [lb/gal]
   rho_c = 9.5; % Massa específica dos produtos químicos em [lb/gal]
   % Parametros de pressão do fluido MO2:
   a1_MO2 = 6.8701; %[lbm/gal]
   b1_MO2 = 3.13e-5; %[lbm/gal/psi]
   c1_MO2 = -2.22e-10; %[lbm/gal/psi^2]
   %Coeficientes de temperatura HC:
   a2_MO2 = -2.82e-3; %[lbm/gal/ºF]
   b2_MO2 = 6.11e-8; %[lbm/gal/psi/ºF]
   c2_MO2 = -9.47e-13; %[lbm/gal/psi^2/ºF]
   % Parametros de pressão do fluido S2: Zamora
   a1_s2 = 6.8467; %[lbm/gal]
   b1_s2 = 3.05e-5; %[lbm/gal/psi]
```

```
c1_s2 = -2.43e-10; %[lbm/gal/psi^2]
% Coeficientes de temperatura sintético S2:
a2_s2 = -2.72e-3; %[lbm/gal/ºF]
b2_s2 = 5.35e-8; %[lbm/gal/psi/ºF]
c2_s2 = -6.99e-13; %[lbm/gal/psi^2/ºF]
% Parametros do fluido B4: Zamora
a1_b4 = 9.8426; %[lbm/gal]
b1_b4 = 1.95e-5; %[lbm/gal/psi]
c1_b4 = -1.01e-10; %[lbm/gal/psi^2]
% Coeficientes de temperatura brine:
a2_b4 = -3.14e-3; %[lbm/gal/ºF]
b2_b4 = 2.31e-8; %[lbm/gal/psi/ºF]
c2_b4 = -8.74e-14; %[lbm/gal/psi^2/ºF]
end
%%  Cálculo da massa específica do fluido dos anulares na condição inicial:
% Cálculo para hidrostática no riser
for i = 1:1:3 %Varredura dos anulare
  for k = 1:dk:1800 % Varredura da profundidade da seção;
    % Contador de iterações para verificar a convergência de Newton-Raphson;
    l = 0;
    % Peso de fluido estimado para iteração 1 do PVT1;
    rho_f1_e(i,k) = 0; %[lb/gal]
    % Hidrostática do fluido na condição 1 para chute inicial
    P1(i,k) = 0.172.*sum(rho_f1_e(i,:).*dk); %[psi]
    % Cálculo da massa específica do fluido sintético para o chute inicial:
    rho_o1(i,k) = (a1_s2 + b1_s2.*P1(i,k) + c1_s2.*P1(i,k).^2) + (a2_s2 + b2_s2.*P1(i,k) + c2_s2.*P1(i,k).^2).*T1(i,k);
    rho_w1(i,k) = (a1_b4 + b1_b4.*P1(i,k) + c1_b4.*P1(i,k).^2) + (a2_b4 + b2_b4.*P1(i,k) + c2_b4.*P1(i,k).^2).*T1(i,k);
    rho_f1(i,k) = (rho_o1(i,k).*fo(i) + rho_w1(i,k).*fw(i) + rho_s.*fs(i) + rho_c.*fc(i));
```

```
% Newton Raphson para convergência da massa específica:
while norm(rho_f1_e(i,k) - rho_f1(i,k)) > norm (d_f);
  l = l+1; % Contador
  rho_f1_e(i,k) = rho_f1(i,k); % Atualizador de valores de massa específica
  P1(i,k) = 0.172.*sum(rho_f1_e(i,:).*dk); % Atualizador da pressão para a massa específica da iteração [psi]
  % Função da massa específica no formato para Newton_Raphson
  rho_o1(i,k) = (a1_s2 + b1_s2.*P1(i,k) + c1_s2.*P1(i,k).^2) + (a2_s2 + b2_s2.*P1(i,k) + c2_s2.*P1(i,k).^2).*T1(i,k);
  rho_w1(i,k) = (a1_b4 + b1_b4.*P1(i,k) + c1_b4.*P1(i,k).^2) + (a2_b4 + b2_b4.*P1(i,k) + c2_b4.*P1(i,k).^2).*T1(i,k);
  f_rho_f1(i,k) = (rho_o1(i,k).*fo(i) + rho_w1(i,k).*fw(i) + rho_s.*fs(i) + rho_c.*fc(i)) - rho_f1_e(i,k);
  % Derivada da função da massa específica;
  dP1(i,k) = 0.172.*k; %Expressao para derivada de P1
  dP1_2(i,k) = 2*P1(i,k).*dP1(i,k); %Expressao para derivada de P1^2 em função de rho_f1_e
  drho_o1(i,k) = (b1_s2.*dP1(i,k) + c1_s2.*dP1_2(i,k)) + (b2_s2.*dP1(i,k) + c2_s2.*dP1_2(i,k)).*T1(i,k);
  drho_w1(i,k) = (b1_b4.*dP1(i,k) + c1_b4.*dP1_2(i,k)) + (b2_b4.*dP1(i,k) + c2_b4.*dP1_2(i,k)).*T1(i,k);
  df_rho_f1(i,k) = (drho_o1(i,k).*fo(i) + drho_w1(i,k).*fw(i)) - 1;
  % Função de Newton-Raphson para cálculo da massa específica
  rho_f1(i,k) = rho_f1_e(i,k) - f_rho_f1(i,k)./df_rho_f1(i,k);
 end
 end
end
% Cálculo para hidrostática no poço
for s = 1:1:4 % Seção analisada
 for i = 1:1:n(s); %Varredura dos anulares
  for k = topo(s):dk:base(s) % Varredura da profundidade da seção;
  % Contador de iterações para verificar a convergência de Newton-Raphson;
```

```
l = 0;
% Peso de fluido estimado para iteração 1 do PVT1;
rho_f1_e(i,k) = 0; %[lb/gal]
% Hidrostática do fluido na condição 1 para chute inicial
P1(i,k) = 0.172.*sum(rho_f1_e(i,:).*dk); %[psi]
% Cálculo da massa específica do fluido sintético para o chute inicial:
rho_o1(i,k) = (a1_s2 + b1_s2.*P1(i,k) + c1_s2.*P1(i,k).^2) + (a2_s2 + b2_s2.*P1(i,k) + c2_s2.*P1(i,k).^2).*T1(i,k);
rho_w1(i,k) = (a1_b4 + b1_b4.*P1(i,k) + c1_b4.*P1(i,k).^2) + (a2_b4 + b2_b4.*P1(i,k) + c2_b4.*P1(i,k).^2).*T1(i,k);
rho_f1(i,k) = (rho_o1(i,k).*fo(i) + rho_w1(i,k).*fw(i) + rho_s.*fs(i) + rho_c.*fc(i));
% Newton Raphson para convergência da massa específica:
   while norm(rho_f1_e(i,k) - rho_f1(i,k)) > norm (d_f);
      l = l+1; % Contador
      rho_f1_e(i,k) = rho_f1(i,k); % Atualizador de valores de massa específica
      P1(i,k) = 0.172.*sum(rho_f1_e(i,:).*dk); % Atualizador da pressão para a massa específica da iteração [psi]
      % Função da massa específica no formato para Newton_Raphson
      rho_o1(i,k) = (a1_s2 + b1_s2.*P1(i,k) + c1_s2.*P1(i,k).^2) + (a2_s2 + b2_s2.*P1(i,k) + c2_s2.*P1(i,k).^2).*T1(i,k);
      rho_w1(i,k) = (a1_b4 + b1_b4.*P1(i,k) + c1_b4.*P1(i,k).^2) + (a2_b4 + b2_b4.*P1(i,k) + c2_b4.*P1(i,k).^2).*T1(i,k);
      f_rho_f1(i,k) = (rho_o1(i,k).*fo(i) + rho_w1(i,k).*fw(i) + rho_s.*fs(i) + rho_c.*fc(i)) - rho_f1_e(i,k);
      % Derivada da função da massa específica;
      dP1(i,k) = 0.172.*k; %Expressao para derivada de P1
      dP1_2(i,k) = 2*P1(i,k).*dP1(i,k); %Expressao para derivada de P1^2 em função de rho_f1_e
      drho_o1(i,k) = (b1_s2.*dP1(i,k) + c1_s2.*dP1_2(i,k)) + (b2_s2.*dP1(i,k) + c2_s2.*dP1_2(i,k)).*T1(i,k);
```

```
                drho_w1(i,k) = (b1_b4.*dP1(i,k) + c1_b4.*dP1_2(i,k)) +
(b2_b4.*dP1(i,k) + c2_b4.*dP1_2(i,k)).*T1(i,k);
                df_rho_f1(i,k) = (drho_o1(i,k).*fo(i) + drho_w1(i,k).*fw(i)) -
1;
                % Função de Newton-Raphson para cálculo da massa
específica
                rho_f1(i,k) = rho_f1_e(i,k) - f_rho_f1(i,k)./df_rho_f1(i,k);
            end
        end
    end
end
%% Rotina específica para o cálculo de pressão na coluna de produção
dentro do riser para condição 1
for k = 1:1:1800 % Varredura da profundidade da seção;
    % Contador de iterações para verifica a convergência de Newton-
Raphson;
    l = 0;
    % Peso de fluido estimado para iteração 1 do PVT1;
    rho_HC1_e(k) = 0; %[lb/gal]
    % Pressão do fluido da COP na condição 1 para chute inicial
    P1_HC1(k) = 0.172.*sum(rho_HC1_e(:).*dk); % Pressão na coluna
para massa específica da iteração [psi]
    % Cálculo da massa específica na coluna de produção para o chute
inicial na condição 1:
    rho_HC1(k) = (a1_MO2 + b1_MO2.*P1_HC1(k) +
c1_MO2.*P1_HC1(k).^2) + (a2_MO2 + b2_MO2.*P1_HC1(k) +
c2_MO2.*P1_HC1(k).^2).*Tg(k);
    % Newton Raphson para convergência da massa específica:
    while norm(rho_HC1_e(k) - rho_HC1(k)) > norm (d_f); %
Comparador de peso específico com a tolerancia
        l = l+1; % Contador
        rho_HC1_e(k) = rho_HC1(k); % Atualizador de valores de massa
específica;
```

```
            P1_HC1(k) = 0.172.*sum(rho_HC1_e(:).*dk); % Atualizador da pressão [psi]
            % Função da massa específica no formato para Newton_Raphson
            f_rho_HC1(k) = (a1_MO2 + b1_MO2.*P1_HC1(k) + c1_MO2.*P1_HC1(k).^2) + (a2_MO2 + b2_MO2.*P1_HC1(k) + c2_MO2.*P1_HC1(k).^2).*Tg(k) - rho_HC1_e(k);
            % Derivada da função da massa específica;
            dP1_col(k) = 0.172.*k; %Expressao para derivada de P1_col
            dP1_col2(k) = 2*P1_HC1(k).*dP1_col(k); %Expressao para derivada de (P1_col)^2
            df_rho_HC1(k) = (b1_MO2.*dP1_col(k) + c1_MO2.*dP1_col2(k)) + (b2_MO2.*dP1_col(k) + c2_MO2.*dP1_col2(k)).*Tg(k) - 1; % Derivada de f_rho_HC1
            % Função de Newton-Raphson para cálculo da massa específica
            rho_HC1(k) = rho_HC1_e(k) - f_rho_HC1(k)./df_rho_HC1(k);
        end
    end
    %% Cálculo da massa específica do HC na COP para condição 1
    %Rotina para cálculo da massa específica no poço
    for s = 1:1:4 % Seção analisada
        for k = topo(s):dk:base(s) % Varredura da profundidade da seção;
            % Contador de iterações para verifica a convergência de Newton-Raphson;
            l = 0;
            % Peso de fluido estimado para iteração 1 do PVT1;
            rho_HC1_e(k) = 0; %[lb/gal]
            % Pressão do fluido da COP na condição 1 para chute inicial
            P1_HC1(k) = 0.172.*sum(rho_HC1_e(:).*dk); % Pressão na coluna para massa específica da iteração [psi]
            % Cálculo da massa específica na coluna de produção para o chute inicial na condição 1:
            rho_HC1(k) = (a1_MO2 + b1_MO2.*P1_HC1(k) + c1_MO2.*P1_HC1(k).^2) + (a2_MO2 + b2_MO2.*P1_HC1(k) + c2_MO2.*P1_HC1(k).^2).*Tg(k);
```

```
% Newton Raphson para convergência da massa específica:
while norm(rho_HC1_e(k) - rho_HC1(k)) > norm (d_f); % Comparador de peso específico com a tolerancia
l = l+1; % Contador
rho_HC1_e(k) = rho_HC1(k); % Atualizador de valores de massa específica;
P1_HC1(k) = 0.172.*sum(rho_HC1_e(:).*dk); % Atualizador da pressão [psi]
% Função da massa específica no formato para Newton_Raphson
f_rho_HC1(k) = (a1_MO2 + b1_MO2.*P1_HC1(k) + c1_MO2.*P1_HC1(k).^2) + (a2_MO2 + b2_MO2.*P1_HC1(k) + c2_MO2.*P1_HC1(k).^2).*Tg(k) - rho_HC1_e(k);
% Derivada da função da massa específica;
dP1_col(k) = 0.172.*k; %Expressao para derivada de P1_col
dP1_col2(k) = 2*P1_HC1(k).*dP1_col(k); %Expressao para derivada de (P1_col^2)
df_rho_HC1(k) = (b1_MO2.*dP1_col(k) + c1_MO2.*dP1_col2(k)) + (b2_MO2.*dP1_col(k) + c2_MO2.*dP1_col2(k)).*Tg(k) - 1; % Derivada de f_rho_HC2
% Função de Newton-Raphson para cálculo da massa específica
rho_HC1(k) = rho_HC1_e(k) - f_rho_HC1(k)./df_rho_HC1(k);
    end
  end
end
%% Cálculo dos volumes de fluido dos anulares do poço na condição 1:
fc = 0.453592/3.78541e-3;
for s = 1:1:4
  for i = 1:1:n(s);
    for k = topo(s):1:base(s)
      Va_fl_0(i,k) = pi.*(a(i+1).^2 - b(i).^2)*dk; % Cálculo do volume inicial dos anulares em [m3] por segmento
      m_0(i,k) = rho_f1(i,k)*fc*Va_fl_0(i,k); % Cálculo da massa inicial de cada seção dos anulares [kg] por segmento
```

```
                % Para condição de contorno 2 --> anular adjacente à formação com fluido no poço aberto
                if cc(s) == 2
                    a(n(s)+1) = af(s); % Último raio torna-se o poço aberto
                    Va_fl_0(n(s),k) = pi.*(a(n(s)+1).^2 - b(n(s)).^2)*dk; % Cálculo do volume inicial dos anulares em [m3] por segmento
                    m_0(n(s),k) = rho_f1(n(s),k)*fc*Va_fl_0(n(s),k); % Cálculo da massa total inicial dos anulares [kg] por segmento
                end
            end
            VaT_fl_0(i) = sum(Va_fl_0(i,:)); % Cálculo dos volumes totais dos anulares em [m3]
            mT_0(i) = sum(m_0(i,:)); % Cálculo da massa inicial dos anulares em [kg/m3]
        end
    end
end
```

B.4.
PVT2

function [rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB, topo, base, n, T2, T2_col)

tic;

%% Pressão deve ser em [psi] e temperatura em [ºF] para o modelo de PVT do Zamora

dk = 1; % comprimento do elemento de cálculo

PF = 5400; % profundidade final para cálculo da perda de carga

La = 1800; % Lamina de água para cálculo da pressão acima do packoff

T2 = (T2)*9/5 + 32; % Converte [K] --> [ºF] para utilizar nas equações de massa específica;

T2_col = (T2_col)*9/5 + 32;

d_f = 1e-4; % Tolerância para convergência da massa específica;

%% Alocação de memoria das variáveis para looping

rho_HC2_e = zeros(1, 5400); P2_col = zeros(1, 5400); f_rho_HC2 = zeros(1, 5400); dP2_col = zeros(1, 5400);

dP2_col2 = zeros(1, 5400); df_rho_HC2 = zeros(1, 5400); rho_HC2 = zeros(1, 5400); APB_col = zeros(1, 5400); Ph_cabp = zeros(1,4);

rho_f2_e = zeros(4, 5400); P2 = zeros(4, 5400); rho_o2 = zeros(4, 5400); rho_w2 = zeros(4, 5400); f_rho_f2 = zeros(4, 5400);

dP2 = zeros(4, 5400); dP2_2 = zeros(4, 5400); drho_w2 = zeros(4, 5400); df_rho_f2 = zeros(4, 5400); rho_f2 = zeros(4, 5400);

drho_o2 = zeros(4, 5400);

%% Composição do fluido sintético utilizado nos anulares

for i = 1:4

fw(i) = 0.20; % Fração de brine

fc(i) = 0.01; % Fração de quimicos

fs(i) = 0.165; % Fração de sólidos

fo(i) = 1 - fw(i) - fc(i) - fs(i); % Fração de parafina

% Massa específica da baritina e produtos químicos:

rho_s = 8.33*4.3; % Massa específica dos sólidos em [lb/gal]

rho_c = 9.5; % Massa específica dos produtos químicos em [lb/gal]

```
% Parametros de pressão do fluido MO2:
a1_MO2 = 6.8701; %[lbm/gal]
b1_MO2 = 3.13e-5; %[lbm/gal/psi]
c1_MO2 = -2.22e-10; %[lbm/gal/psi^2]
%Coeficientes de temperatura HC:
a2_MO2 = -2.82e-3; %[lbm/gal/ºF]
b2_MO2 = 6.11e-8; %[lbm/gal/psi/ºF]
c2_MO2 = -9.47e-13; %[lbm/gal/psi^2/ºF]
% Parametros de pressão do fluido S2: Zamora
a1_s2 = 6.8467; %[lbm/gal]
b1_s2 = 3.05e-5; %[lbm/gal/psi]
c1_s2 = -2.43e-10; %[lbm/gal/psi^2]
% Coeficientes de temperatura sintético S2:
a2_s2 = -2.72e-3; %[lbm/gal/ºF]
b2_s2 = 5.35e-8; %[lbm/gal/psi/ºF]
c2_s2 = -6.99e-13; %[lbm/gal/psi^2/ºF]
% Parametros do fluido B4: Zamora
a1_b4 = 9.8426; %[lbm/gal]
b1_b4 = 1.95e-5; %[lbm/gal/psi]
c1_b4 = -1.01e-10; %[lbm/gal/psi^2]
% Coeficientes de temperatura brine:
a2_b4 = -3.14e-3; %[lbm/gal/ºF]
b2_b4 = 2.31e-8; %[lbm/gal/psi/ºF]
c2_b4 = -8.74e-14; %[lbm/gal/psi^2/ºF]
end
%% Rotina específica para o cálculo de pressão na coluna de produção dentro do riser para condição 2
for k = 1:1:1800 % Varredura da profundidade da seção;
% Contador de iterações para verifica a convergência de Newton-Raphson;
l = 0;
% Peso de fluido estimado para iteração 1 do PVT2;
rho_HC2_e(k) = 0; %[lb/gal]
% Pressão do fluido da COP na condição 2 para chute inicial
```

P2_col(k) = 0.172.*sum(rho_HC2_e(:).*dk); % Pressão na coluna para massa específica da iteração [psi]

% Cálculo da massa específica na coluna de produção para o chute inicial na condição 2:

rho_HC2(k) = (a1_MO2 + b1_MO2.*P2_col(k) + c1_MO2.*P2_col(k).^2) + (a2_MO2 + b2_MO2.*P2_col(k) + c2_MO2.*P2_col(k).^2).*T2_col(k);

% Newton Raphson para convergência da massa específica:

while norm(rho_HC2_e(k) - rho_HC2(k)) > norm (d_f); % Comparador de peso específico com a tolerancia

l = l+1; % Contador

rho_HC2_e(k) = rho_HC2(k); % Atualizador de valores de massa específica;

P2_col(k) = 0.172.*sum(rho_HC2_e(:).*dk); % Atualizador da pressão [psi]

% Função da massa específica no formato para Newton_Raphson

f_rho_HC2(k) = (a1_MO2 + b1_MO2.*P2_col(k) + c1_MO2.*P2_col(k).^2) + (a2_MO2 + b2_MO2.*P2_col(k) + c2_MO2.*P2_col(k).^2).*T2_col(k) - rho_HC2_e(k);

% Derivada da função da massa específica;

dP2_col(k) = 0.172.*k; %Expressao para derivada de P2_col

dP2_col2(k) = 2*P2_col(k).*dP2_col(k); %Expressao para derivada de (P2_col)^2

df_rho_HC2(k) = (b1_MO2.*dP2_col(k) + c1_MO2.*dP2_col2(k)) + (b2_MO2.*dP2_col(k) + c2_MO2.*dP2_col2(k)).*T2_col(k) - 1; % Derivada de f_rho_HC2

% Função de Newton-Raphson para cálculo da massa específica

rho_HC2(k) = rho_HC2_e(k) - f_rho_HC2(k)./df_rho_HC2(k);

end

APB_col(k) = (P2_col(k) - P1_col(k)); % Diferencial de pressão na coluna de produção para condição 2 [psi]

end

%% Cálculo da massa específica do HC na COP para condição 2

%Rotina para cálculo da massa específica no poço

```
for s = 1:1:4 % Seção analisada
  for k = topo(s):dk:base(s) % Varredura da profundidade da seção;
    % Contador de iterações para verifica a convergência de Newton-Raphson;
    l = 0;
    % Peso de fluido estimado para iteração 1 do PVT2;
    rho_HC2_e(k) = 0; %[lb/gal]
    % Pressão do fluido da COP na condição 2 para chute inicial
    P2_col(k) = 0.172.*sum(rho_HC2_e(:).*dk); % Pressão na coluna para massa específica da iteração [psi]
    % Cálculo da massa específica na coluna de produção para o chute inicial na condição 2:
    rho_HC2(k) = (a1_MO2 + b1_MO2.*P2_col(k) + c1_MO2.*P2_col(k).^2) + (a2_MO2 + b2_MO2.*P2_col(k) + c2_MO2.*P2_col(k).^2).*T2_col(k);
    % Newton Raphson para convergência da massa específica:
    while norm(rho_HC2_e(k) - rho_HC2(k)) > norm (d_f); % Comparador de peso específico com a tolerancia
      l = l+1; % Contador
      rho_HC2_e(k) = rho_HC2(k); % Atualizador de valores de massa específica;
      P2_col(k) = 0.172.*sum(rho_HC2_e(:).*dk); % Atualizador da pressão [psi]
      % Função da massa específica no formato para Newton_Raphson
      f_rho_HC2(k) = (a1_MO2 + b1_MO2.*P2_col(k) + c1_MO2.*P2_col(k).^2) + (a2_MO2 + b2_MO2.*P2_col(k) + c2_MO2.*P2_col(k).^2).*T2_col(k) - rho_HC2_e(k);
      % Derivada da função da massa específica;
      dP2_col(k) = 0.172.*k; %Expressao para derivada de P2_col
      dP2_col2(k) = 2*P2_col(k).*dP2_col(k); %Expressao para derivada de (P2_col^2)
      df_rho_HC2(k) = (b1_MO2.*dP2_col(k) + c1_MO2.*dP2_col2(k)) + (b2_MO2.*dP2_col(k) + c2_MO2.*dP2_col2(k)).*T2_col(k) - 1; % Derivada de f_rho_HC2
```

```
                % Função de Newton-Raphson para cálculo da massa específica
                rho_HC2(k) = rho_HC2_e(k) - f_rho_HC2(k)./df_rho_HC2(k);
            end
            APB_col(k) = (P2_col(k) - P1_col(k)); % Diferencial de pressão na coluna de produção para condição 2 [psi]
        end
    end
    %%  Cálculo da massa específica do fluido dos anulares na condição 2 com APB:
    for s = 1:1:4 % Seção analisada
        for i = 1:1:n(s); %Varredura dos anulares
            for k = topo(s):dk:base(s) % Varredura da profundidade da seção;
                % Contador de iterações para verificar a convergência de Newton-Raphson;
                l = 0;
                % Peso de fluido estimado para iteração 1 do PVT2;
                rho_f2_e(i,k) = 12.0; %[lb/gal]
                % Pressão hidrostática + APB do fluido na condição 2 para chute inicial
                Ph_cabp(i) = P1(i,La); % Constante [psi]
                P2(i,k) = APB(i) + Ph_cabp(i) + 0.172.*sum(rho_f2_e(i,:).*dk); %[psi]
                % Cálculo da massa específica do fluido sintético na condição 2 para o chute inicial:
                rho_o2(i,k) = (a1_s2 + b1_s2.*P2(i,k) + c1_s2.*P2(i,k).^2) + (a2_s2 + b2_s2.*P2(i,k) + c2_s2.*P2(i,k).^2).*T2(i,k);
                rho_w2(i,k) = (a1_b4 + b1_b4.*P2(i,k) + c1_b4.*P2(i,k).^2) + (a2_b4 + b2_b4.*P2(i,k) + c2_b4.*P2(i,k).^2).*T2(i,k);
                rho_f2(i,k) = (rho_f1(i,k))./(1 + fo(i).*(rho_o1(i,k)./rho_o2(i,k) - 1) + fw(i).*(rho_w1(i,k)./rho_w2(i,k) - 1));
                % Newton Raphson para convergência da massa específica:
                while norm(rho_f2_e(i,k) - rho_f2(i,k)) > norm (d_f);
                    l = l+1; % Contador
```

```
                rho_f2_e(i,k) = rho_f2(i,k); % Atualizador de valores de massa
específica
                P2(i,k) = APB(i) + Ph_cabp(i) + 0.172.*sum(rho_f2_e(i,:).*dk);
% Atualizador da pressão para a massa específica da iteração [psi]
                % Função da massa específica no formato para
Newton_Raphson
                rho_o2(i,k) = (a1_s2 + b1_s2.*P2(i,k) + c1_s2.*P2(i,k).^2) +
(a2_s2 + b2_s2.*P2(i,k) + c2_s2.*P2(i,k).^2).*T2(i,k);
                rho_w2(i,k) = (a1_b4 + b1_b4.*P2(i,k) + c1_b4.*P2(i,k).^2) +
(a2_b4 + b2_b4.*P2(i,k) + c2_b4.*P2(i,k).^2).*T2(i,k);
                f_rho_f2(i,k) = (rho_f1(i,k))./(1 +
fo(i).*(rho_o1(i,k)./rho_o2(i,k) - 1) + fw(i).*(rho_w1(i,k)./rho_w2(i,k) - 1)) -
rho_f2_e(i,k);
                % Derivada da função da massa específica;
                dP2(i,k) = 0.172.*(k - La); %Expressao para derivada de P2
                dP2_2(i,k) = 2*P2(i,k).*dP2(i,k); %Expressao para derivada de
P2^2 em função de rho_f2_e
                drho_o2(i,k) = (b1_s2.*dP2(i,k) + c1_s2.*dP2_2(i,k)) +
(b2_s2.*dP2(i,k) + c2_s2.*dP2_2(i,k)).*T2(i,k);
                drho_w2(i,k) = (b1_b4.*dP2(i,k) + c1_b4.*dP2_2(i,k)) +
(b2_b4.*dP2(i,k) + c2_b4.*dP2_2(i,k)).*T2(i,k);
                df_rho_f2(i,k) = (-1).*rho_f1(i,k)./(1 +
fo(i).*(rho_o1(i,k)./rho_o2(i,k) - 1) + fw(i).*(rho_w1(i,k)./rho_w2(i,k) -
1)).^2.*((-1).*fo(i).*rho_o1(i,k)./rho_o2(i,k).^2.*drho_o2(i,k) + (-
1).*fw(i).*rho_w1(i,k)./rho_w2(i,k).^2.*drho_w2(i,k)) - 1;
                % Função de Newton-Raphson para cálculo da massa específica
                rho_f2(i,k) = rho_f2_e(i,k) - f_rho_f2(i,k)./df_rho_f2(i,k);
            end
        end
    end
    tempo_ciclo_PVT_2 = toc;
end
```

B.5.
DVSal

```
function [raio_sal, desl_sal, DVaT_sal] = DVsal(t, raio_sal, desl_sal, Va_fl_0, APB, af, b, topo, base, cc, n, m)
dk = 1;
APB_sal = APB(2);
%% Geração do vetor tempo:
t0 = 72; % [h] para descontar o relaxamento inicial
if m == 1
   ti = 0 + t0; %[h]
else
   ti = t(m-1) + t0; %[h]
end
tf = t(m) + t0; %[h]
%% Cálculo do fechamento diametral do poço devido ao sal na seção 3 anular 2
   for s = 3
      for i = n(s) % anular 2 com sal na seção 3
         if cc(s) == 2
            %% Cálculo da taxa de deslocamento do sal em função de P e t no intervalo #1
            for k = 4150:1:4250 % Tempo inicial com APB_sal = 0 nos intervalos determinados
               epslon(k) = 2.9441e-4*tf^(-4.6736e-1); % 4200 m / 12 ppg
               if APB_sal > 0.172*4200*(12.25 - 12.0) %Interpolação [psi]
                  epslon(k) = (2.02883e-4*tf^(-4.4941e-1) + 2.9441e-4*tf^(-4.6736e-1))/2; % [in/h]
               end
               if APB_sal > 0.172*4200*(12.5 - 12.0)
                  epslon(k) = 2.02883e-4*tf^(-4.4941e-1);
               end
               if APB_sal > 0.172*4200*(12.75 - 12.0) % Interpolação
```

```
    epslon(k) = (2.02883e-4*tf^(-4.4941e-1) + 1.1325e-4*tf^(-
4.1242e-1))/2;
  end
  if APB_sal > 0.172*4200*(13.0 - 12.0)
    epslon(k) = 1.1325e-4*tf^(-4.1242e-1);
  end
  if APB_sal > 0.172*4200*(13.25 - 12.0) % Interpolação
    epslon(k) = (1.1325e-4*tf^(-4.1242e-1) + 6.1853e-5*tf^(-
4.2392e-1))/2;
  end
  if APB_sal > 0.172*4200*(13.5 - 12.0)
    epslon(k) = 6.1853e-5*tf^(-4.2392e-1);
  end
  if APB_sal > 0.172*4200*(13.75 - 12.0) % Interpolação
    epslon(k) = (6.1853e-5*tf^(-4.2392e-1) + 2.2423e-5*tf^(-
7.9378e-1))/2;
  end
  if APB_sal > 0.172*4200*(14.0 - 12.0)
    epslon(k) = 2.2423e-5*tf^(-7.9378e-1);
  end
  if APB_sal > 0.172*4200*(14.25 - 12.0) % Interpolação
    epslon(k) = 0;
  end
  if APB_sal > 0.172*4200*(14.5 - 12.0)
    epslon(k) = -1.1424e-5;
  end
  if APB_sal > 0.172*4200*(14.75 - 12.0) % Interpolação
    epslon(k) = ((-1.1424e-5) + (-2.3055e-5))/2;
  end
  if APB_sal > 0.172*4200*(15.0 - 12.0)
    epslon(k) = -2.3055e-5;
  end
  if APB_sal > 0.172*4200*(15.25 - 12.0) % Interpolação
```

```
                epslon(k) = ((-2.3055e-5) + (-4.9203e-5))/2; % 4200 m / 15
ppg
            end
            if APB_sal > 0.172*4200*(15.5 - 12.0)
              epslon(k) = -4.9203e-5;
            end
            if APB_sal > 0.172*4200*(15.75 - 12.0) % Interpolação
              epslon(k) = ((-4.9203e-5) + (-7.5351e-5))/2; % 4200 m / 15
ppg
            end
            if APB_sal > 0.172*4200*(16.0 - 12.0)
              epslon(k) = -7.5351e-5;
            end
            desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m]
deslocamento no intervalo de tempo "m" do passo da simulação
          end
          %% Cálculo da taxa de deslocamento do sal em função de P e t no
intervalo #2
          for k = 4250:1:4350 % Tempo inicial com APB_sal = 0 nos
intervalos determinados
            epslon(k) = 3.6724e-4*tf^(-4.5393e-1); % 4200 m / 12 ppg
            if APB_sal > 0.172*4300*(12.25 - 12.0) %Interpolação [psi]
              epslon(k) = (3.6724e-4*tf^(-4.5393e-1) + 2.5397e-4*tf^(-
4.3787e-1))/2; % [in/h]
            end
            if APB_sal > 0.172*4300*(12.5 - 12.0)
              epslon(k) = 2.5397e-4*tf^(-4.3787e-1);
            end
            if APB_sal > 0.172*4300*(12.75 - 12.0) % Interpolação
              epslon(k) = (2.5397e-4*tf^(-4.3787e-1) + 1.4197e-4*tf^(-
4.0358e-1))/2;
            end
            if APB_sal > 0.172*4300*(13.0 - 12.0)
              epslon(k) = 1.4197e-4*tf^(-4.0358e-1);
```

```
end
if APB_sal > 0.172*4300*(13.25 - 12.0) % Interpolação
  epslon(k) = (1.4197e-4*tf^(-4.0358e-1) + 7.6071e-5*tf^(-
4.0718e-1))/2;
end
if APB_sal > 0.172*4300*(13.5 - 12.0)
  epslon(k) = 7.6071e-5*tf^(-4.0718e-1);
end
if APB_sal > 0.172*4300*(13.75 - 12.0) % Interpolação
  epslon(k) = (7.6071e-5*tf^(-4.0718e-1) + 1.0539e-5*tf^(-
4.7362e-1))/2;
end
if APB_sal > 0.172*4300*(14.0 - 12.0)
  epslon(k) = 1.0539e-5*tf^(-4.7362e-1);
end
if APB_sal > 0.172*4300*(14.25 - 12.0) % Interpolação
  epslon(k) = 0;
end
if APB_sal > 0.172*4300*(14.5 - 12.0)
  epslon(k) = -1.0694e-5;
end
if APB_sal > 0.172*4300*(14.75 - 12.0) % Interpolação
  epslon(k) = ((-1.0694e-5) + (-2.8913e-5))/2;
end
if APB_sal > 0.172*4300*(15.0 - 12.0)
  epslon(k) = -2.8913e-5;
end
if APB_sal > 0.172*4300*(15.25 - 12.0) % Interpolação
  epslon(k) = ((-2.8913e-5) + (-6.2100e-5))/2;
end
if APB_sal > 0.172*4300*(15.5 - 12.0)
  epslon(k) = -6.2100e-5;
end
if APB_sal > 0.172*4300*(15.75 - 12.0) % Interpolação
```

```
    epslon(k) = ((-6.2100e-5) + (-9.5286e-5))/2;
  end
  if APB_sal > 0.172*4300*(16.0 - 12.0)
    epslon(k) = -9.5286e-5;
  end
  desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d] deslocamento no intervalo de tempo do passo da simulação
end
%% Cálculo da taxa de deformação do sal em função de P e t no intervalo #3
for k = 4350:1:4450 % Tempo inicial com APB_sal = 0 nos intervalos determinados
  epslon(k) = 4.9869e-4*tf^(-4.6667e-1); % 4400 m / 12 ppg
  if APB_sal > 0.172*4400*(12.25 - 12.0) % Interpolação
    epslon(k) = (4.9869e-4*tf^(-4.6667e-1) + 3.2533e-4*tf^(-4.3829e-1))/2;
  end
  if APB_sal > 0.172*4400*(12.5 - 12.0)
    epslon(k) = 3.2533e-4*tf^(-4.3829e-1);
  end
  if APB_sal > 0.172*4400*(12.75 - 12.0) % Interpolação
    epslon(k) = (3.2533e-4*tf^(-4.3829e-1) + 1.6222e-4*tf^(-3.7986e-1))/2;
  end
  if APB_sal > 0.172*4400*(13.0 - 12.0)
    epslon(k) = 1.6222e-4*tf^(-3.7986e-1);
  end
  if APB_sal > 0.172*4400*(13.25 - 12.0) % Interpolação
    epslon(k) = (1.6222e-4*tf^(-3.7986e-1) + 8.6310e-5*tf^(-3.7791e-1))/2;
  end
  if APB_sal > 0.172*4400*(13.5 - 12.0)
    epslon(k) = 8.6310e-5*tf^(-3.7791e-1);
  end
```

```
if APB_sal > 0.172*4400*(13.75 - 12.0) % Interpolação
  epslon(k) = (8.6310e-5*tf^(-3.7791e-1) + 1.0538e-5*tf^(-
3.5242e-1))/2;
end
if APB_sal > 0.172*4400*(14.0 - 12.0)
  epslon(k) = 1.0538e-5*tf^(-3.5242e-1);
end
if APB_sal > 0.172*4400*(14.25 - 12.0) % Interpolação
  epslon(k) = 0;
end
if APB_sal > 0.172*4400*(14.5 - 12.0)
  epslon(k) = -1.7698e-5;
end
if APB_sal > 0.172*4400*(14.75 - 12.0) % Interpolação
  epslon(k) = ((-1.7698e-5) + (-3.6597e-5))/2;
end
if APB_sal > 0.172*4400*(15.0 - 12.0)
  epslon(k) = -3.6597e-5;
end
if APB_sal > 0.172*4400*(15.25 - 12.0) % Interpolação
  epslon(k) = ((-3.6597e-5) + (-7.9109e-5))/2;
end
if APB_sal > 0.172*4400*(15.5 - 12.0)
  epslon(k) = -7.9109e-5;
end
if APB_sal > 0.172*4400*(15.75 - 12.0) % Interpolação
  epslon(k) = ((-7.9109e-5) + (-1.2162e-4))/2;
end
if APB_sal > 0.172*4400*(16.0 - 12.0) %Inclui APB superior a
15 ppg
  epslon(k) = -1.2162e-4;
end
desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
```

```
end
%% Cálculo da taxa de deformação do sal em função de P e t no intervalo #4
for k = 4450:1:4550 % Tempo inicial com APB_sal = 0 nos intervalos determinados
  epslon(k) = 5.7913e-4*tf^(-4.4856e-1);
  if APB_sal > 0.172*4500*(12.25 - 12.0) % Interpolação
    epslon(k) = (5.7913e-4*tf^(-4.4856e-1) + 3.7025e-4*tf^(-4.1611e-1))/2;
  end
  if APB_sal > 0.172*4500*(12.5 - 12.0)
    epslon(k) = 3.7025e-4*tf^(-4.1611e-1);
  end
  if APB_sal > 0.172*4500*(12.75 - 12.0) % Interpolação
    epslon(k) = (3.7025e-4*tf^(-4.1611e-1) + 1.777e-4*tf^(-3.5001e-1))/2;
  end
  if APB_sal > 0.172*4500*(13.0 - 12.0)
    epslon(k) = 1.777e-4*tf^(-3.5001e-1);
  end
  if APB_sal > 0.172*4500*(13.25 - 12.0) % Interpolação
    epslon(k) = (1.777e-4*tf^(-3.5001e-1) + 9.5221e-5*tf^(-3.4538e-1))/2;
  end
  if APB_sal > 0.172*4500*(13.5 - 12.0)
    epslon(k) = 9.5221e-5*tf^(-3.4538e-1);
  end
  if APB_sal > 0.172*4500*(13.75 - 12.0) % Interpolação
    epslon(k) = (9.5221e-5*tf^(-3.4538e-1) + 1.3225e-5*tf^(-2.9787e-1))/2;
  end
  if APB_sal > 0.172*4500*(14.0 - 12.0)
    epslon(k) = 1.3225e-5*tf^(-2.9787e-1);
  end
```

```
                if APB_sal > 0.172*4500*(14.25 - 12.0) % Interpolação
                  epslon(k) = 0;
                end
                if APB_sal > 0.172*4500*(14.5 - 12.0)
                  epslon(k) = -1.6567e-5;
                end
                if APB_sal > 0.172*4400*(14.75 - 12.0) % Interpolação
                  epslon(k) = ((-1.6567e-5) + (-4.6634e-5))/2;
                end
                if APB_sal > 0.172*4500*(15.0 - 12.0)
                  epslon(k) = -4.6634e-5;
                end
                if APB_sal > 0.172*4500*(15.25 - 12.0) % Interpolação
                  epslon(k) = ((-4.6634e-5) + (-1.0141e-4))/2;
                end
                if APB_sal > 0.172*4500*(15.5 - 12.0)
                  epslon(k) = -1.0141e-4;
                end
                if APB_sal > 0.172*4500*(15.75 - 12.0) % Interpolação
                  epslon(k) = ((-1.0141e-4) + (-1.5619e-4))/2;
                end
                if APB_sal > 0.172*4500*(16.0 - 12.0) %Inclui APB superior a
15 ppg
                  epslon(k) = -1.5619e-4; %
                end
                desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
              end
              %% Cálculo da taxa de deformação do sal em função de P e t no
intervalo #5
              for k = 4550:dk:4650 % Tempo inicial com APB_sal = 0 nos
intervalos determinados
                epslon(k) = 6.6208e-4*tf^(-4.2891e-1); % 4600 m / 12 ppg
                if APB_sal > 0.172*4600*(12.25 - 12.0) % Interpolação
```

```
                epslon(k) = (6.6208e-4*tf^(-4.2891e-1) + 4.1639e-4*tf^(-
3.9302e-1))/2;
            end
            if APB_sal > 0.172*4600*(12.5 - 12.0)
              epslon(k) = 4.1639e-4*tf^(-3.9302e-1);
            end
            if APB_sal > 0.172*4600*(12.75 - 12.0) % Interpolação
              epslon(k) = (4.1639e-4*tf^(-3.9302e-1) + 1.9406e-4*tf^(-
3.2072e-1))/2;
            end
            if APB_sal > 0.172*4600*(13.0 - 12.0)
              epslon(k) = 1.9406e-4*tf^(-3.2072e-1);
            end
            if APB_sal > 0.172*4600*(13.25 - 12.0) % Interpolação
              epslon(k) = (1.9406e-4*tf^(-3.2072e-1) + 1.0593e-4*tf^(-
3.1597e-1))/2;
            end
            if APB_sal > 0.172*4600*(13.5 - 12.0)
              epslon(k) = 1.0593e-4*tf^(-3.1597e-1);
            end
            if APB_sal > 0.172*4600*(13.75 - 12.0) % Interpolação
              epslon(k) = (1.0593e-4*tf^(-3.1597e-1) + 1.8241e-5*tf^(-
2.7533e-1))/2;
            end
            if APB_sal > 0.172*4600*(14.0 - 12.0)
              epslon(k) = 1.8241e-5*tf^(-2.7533e-1);
            end
            if APB_sal > 0.172*4600*(14.25 - 12.0) % Interpolação
              epslon(k) = 0;
            end
            if APB_sal > 0.172*4600*(14.5 - 12.0)
              epslon(k) = -2.8203e-5;
            end
            if APB_sal > 0.172*4600*(14.75 - 12.0) % Interpolação
```

```
            epslon(k) = ((-2.8203e-5) + (-5.9629e-5))/2;
        end
        if APB_sal > 0.172*4600*(15.0 - 12.0)
            epslon(k) = -5.9629e-5;
        end
        if APB_sal > 0.172*4600*(15.25 - 12.0) % Interpolação
            epslon(k) = ((-5.9629e-5) + (-1.3036e-4))/2;
        end
        if APB_sal > 0.172*4600*(15.5 - 12.0)
            epslon(k) = -1.3036e-4;
        end
        if APB_sal > 0.172*4600*(15.75 - 12.0) % Interpolação
            epslon(k) = ((-1.3036e-4) + (-2.0110e-4))/2;
        end
        if APB_sal > 0.172*4600*(16.0 - 12.0)
            epslon(k) = -2.0110e-4;
        end
        desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
    end
    %% Cálculo da taxa de deformação do sal em função de P e t no
intervalo #6
    for k = 4650:dk:4750 % Tempo inicial com APB_sal = 0 nos
intervalos determinados
        epslon(k) = 7.5405e-4*tf^(-4.0966e-1); % 4700 m / 12 ppg
        if APB_sal > 0.172*4700*(12.25 - 12.0) % Interpolação
            epslon(k) = (7.5405e-4*tf^(-4.0966e-1) + 4.7064e-4*tf^(-
3.7187e-1))/2;
        end
        if APB_sal > 0.172*4700*(12.5 - 12.0)
            epslon(k) = 4.7064e-4*tf^(-3.7187e-1);
        end
        if APB_sal > 0.172*4700*(12.75 - 12.0) % Interpolação
```

```
                epslon(k) = (4.7064e-4*tf^(-3.7187e-1) + 2.1707e-4*tf^(-
2.9671e-1))/2;
              end
              if APB_sal > 0.172*4700*(13.0 - 12.0)
                epslon(k) = 2.1707e-4*tf^(-2.9671e-1);
              end
              if APB_sal > 0.172*4700*(13.25 - 12.0) % Interpolação
                epslon(k) = (2.1707e-4*tf^(-2.9671e-1) + 1.2158e-4*tf^(-
2.9358e-1))/2;
              end
              if APB_sal > 0.172*4700*(13.5 - 12.0)
                epslon(k) = 1.2158e-4*tf^(-2.9358e-1);
              end
              if APB_sal > 0.172*4700*(13.75 - 12.0) % Interpolação
                epslon(k) = (1.2158e-4*tf^(-2.9358e-1) + 2.6180e-5*tf^(-
2.7013e-1))/2;
              end
              if APB_sal > 0.172*4700*(14.0 - 12.0)
                epslon(k) = 2.6180e-5*tf^(-2.7013e-1);
              end
              if APB_sal > 0.172*4700*(14.25 - 12.0) % Interpolação
                epslon(k) = 0;
              end
              if APB_sal > 0.172*4700*(14.5 - 12.0)
                epslon(k) = -2.6195e-5;
              end
              if APB_sal > 0.172*4700*(14.75 - 12.0) % Interpolação
                epslon(k) = ((-2.6195e-5) + (-1.2401e-4))/2;
              end
              if APB_sal > 0.172*4700*(15.0 - 12.0)
                epslon(k) = -1.2401e-4;
              end
              if APB_sal > 0.172*4700*(15.25 - 12.0) % Interpolação
                epslon(k) = ((-1.2401e-4) + (-1.9141e-4))/2;
```

```
                end
                if APB_sal > 0.172*4700*(15.5 - 12.0)
                  epslon(k) = -1.9141e-4;
                end
                if APB_sal > 0.172*4700*(15.75 - 12.0) % Interpolação
                  epslon(k) = ((-1.9141e-4) + (-2.5881e-4))/2;
                end
                if APB_sal > 0.172*4700*(16.0 - 12.0)
                  epslon(k) = -2.5881e-4;
                end
                desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
              end
              %% Cálculo da taxa de deformação do sal em função de P e t no
intervalo #7
              for k = 4750:dk:4850 % Tempo inicial com APB_sal = 0 nos
intervalos determinados
                epslon(k) = 8.6250e-4*tf^(-3.9209e-1);
                if APB_sal > 0.172*4800*(12.25 - 12.0) % Interpolação
                  epslon(k) = (8.6250e-4*tf^(-3.9209e-1) + 5.4033e-4*tf^(-
3.5438e-1))/2;
                end
                if APB_sal > 0.172*4800*(12.5 - 12.0)
                  epslon(k) = 5.4033e-4*tf^(-3.5438e-1);
                end
                if APB_sal > 0.172*4800*(12.75 - 12.0) % Interpolação
                  epslon(k) = (5.4033e-4*tf^(-3.5438e-1) + 2.5252e-4*tf^(-
2.8051e-1))/2;
                end
                if APB_sal > 0.172*4800*(13.0 - 12.0)
                  epslon(k) = 2.5252e-4*tf^(-2.8051e-1);
                end
                if APB_sal > 0.172*4800*(13.25 - 12.0) % Interpolação
```

```
    epslon(k) = (2.5252e-4*tf^(-2.8051e-1) + 1.4533e-4*tf^(-
2.7978e-1))/2;
  end
  if APB_sal > 0.172*4800*(13.5 - 12.0)
    epslon(k) = 1.4533e-4*tf^(-2.7978e-1);
  end
  if APB_sal > 0.172*4800*(13.75 - 12.0) % Interpolação
    epslon(k) = (1.4533e-4*tf^(-2.7978e-1) + 3.8031e-5*tf^(-
2.7458e-1))/2;
  end
  if APB_sal > 0.172*4800*(14.0 - 12.0)
    epslon(k) = 3.8031e-5*tf^(-2.7458e-1);
  end
  if APB_sal > 0.172*4800*(14.25 - 12.0) % Interpolação
    epslon(k) = 0;
  end
  if APB_sal > 0.172*4800*(14.5 - 12.0)
    epslon(k) = -4.5345e-5;
  end
  if APB_sal > 0.172*4800*(14.75 - 12.0) % Interpolação
    epslon(k) = ((-4.5345e-5) + (-9.7415e-5))/2;
  end
  if APB_sal > 0.172*4800*(15.0 - 12.0)
    epslon(k) = -9.7415e-5;
  end
  if APB_sal > 0.172*4800*(15.25 - 12.0) % Interpolação
    epslon(k) = ((-9.7415e-5) + (-2.1482e-4))/2;
  end
  if APB_sal > 0.172*4800*(15.5 - 12.0)
    epslon(k) = -2.1482e-4;
  end
  if APB_sal > 0.172*4800*(15.75 - 12.0) % Interpolação
    epslon(k) = ((-2.1482e-4) + (-3.3222e-4))/2;
  end
```

```
                if APB_sal > 0.172*4800*(16.0 - 12.0)
                  epslon(k) = -3.3222e-4;
                end
                desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
              end
              %% Cálculo da taxa de deformação do sal em função de P e t no
intervalo #8
              for k = 4850:dk:4950 % Tempo inicial com APB_sal = 0 nos
intervalos determinados
                epslon(k) = 9.9528e-4*tf^(-3.7685e-1);
                if APB_sal > 0.172*4900*(12.25 - 12.0) % Interpolação
                  epslon(k) = (9.9528e-4*tf^(-3.7685e-1) + 6.3305e-4*tf^(-
3.4126e-1))/2;
                end
                if APB_sal > 0.172*4900*(12.5 - 12.0)
                  epslon(k) = 6.3305e-4*tf^(-3.4126e-1);
                end
                if APB_sal > 0.172*4900*(12.75 - 12.0) % Interpolação
                  epslon(k) = (6.3305e-4*tf^(-3.4126e-1) + 3.6609e-4*tf^(-
2.7270e-1))/2;
                end
                if APB_sal > 0.172*4900*(13.0 - 12.0)
                  epslon(k) = 3.6609e-4*tf^(-2.7270e-1);
                end
                if APB_sal > 0.172*4900*(13.25 - 12.0) % Interpolação
                  epslon(k) = (3.6609e-4*tf^(-2.7270e-1) + 1.8086e-4*tf^(-
2.7437e-1))/2;
                end
                if APB_sal > 0.172*4900*(13.5 - 12.0)
                  epslon(k) = 1.8086e-4*tf^(-2.7437e-1);
                end
                if APB_sal > 0.172*4900*(13.75 - 12.0) % Interpolação
```

```
                epslon(k) = (1.8086e-4*tf^(-2.7437e-1) + 5.5034e-5*tf^(-
2.8407e-1))/2;
            end
            if APB_sal > 0.172*4900*(14.0 - 12.0)
              epslon(k) = 5.5034e-5*tf^(-2.8407e-1);
            end
            if APB_sal > 0.172*4900*(14.25 - 12.0) % Interpolação
              epslon(k) = 0;
            end
            if APB_sal > 0.172*4900*(14.5 - 12.0)
              epslon(k) = -4.1563e-5;
            end
            if APB_sal > 0.172*4900*(14.75 - 12.0) % Interpolação
              epslon(k) = ((-4.1563e-5) + (-1.2401e-4))/2;
            end
            if APB_sal > 0.172*4900*(15.0 - 12.0)
              epslon(k) = -1.2401e-4;
            end
            if APB_sal > 0.172*4900*(15.25 - 12.0) % Interpolação
              epslon(k) = ((-1.2401e-4) + (-2.7442e-4))/2;
            end
            if APB_sal > 0.172*4900*(15.5 - 12.0)
              epslon(k) = -2.7442e-4;
            end
            if APB_sal > 0.172*4900*(15.75 - 12.0) % Interpolação
              epslon(k) = ((-2.7442e-4) + (-4.2483e-4))/2;
            end
            if APB_sal > 0.172*4900*(16.0 - 12.0)
              epslon(k) = -4.2483e-4;
            end
            desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
        end
```

```
%% Cálculo da taxa de deformação do sal em função de P e t no intervalo #9
for k = 4950:dk:5000 % Tempo inicial com APB_sal = 0 nos intervalos determinados
  epslon(k) = 1.1018e-3*tf^(-3.5304e-1); % 5000 m / 12 ppg
  if APB_sal > 0.172*5000*(12.25 - 12.0) % Interpolação
    epslon(k) = (1.1018e-3*tf^(-3.5304e-1) + 7.3205e-4*tf^(-3.2528e-1))/2;
  end
  if APB_sal > 0.172*5000*(12.5 - 12.0)
    epslon(k) = 7.3205e-4*tf^(-3.2528e-1);
  end
  if APB_sal > 0.172*5000*(12.75 - 12.0) % Interpolação
    epslon(k) = (7.3205e-4*tf^(-3.2528e-1) + 3.8693e-4*tf^(-2.7232e-1))/2;
  end
  if APB_sal > 0.172*5000*(13.0 - 12.0)
    epslon(k) = 3.8693e-4*tf^(-2.7232e-1);
  end
  if APB_sal > 0.172*5000*(13.25 - 12.0) % Interpolação
    epslon(k) = (3.8693e-4*tf^(-2.7232e-1) + 2.3250e-4*tf^(-2.7584e-1))/2;
  end
  if APB_sal > 0.172*5000*(13.5 - 12.0)
    epslon(k) = 2.3250e-4*tf^(-2.7584e-1);
  end
  if APB_sal > 0.172*5000*(13.75 - 12.0) % Interpolação
    epslon(k) = (2.3250e-4*tf^(-2.7584e-1) + 7.8526e-5*tf^(-2.9556e-1))/2;
  end
  if APB_sal > 0.172*5000*(14.0 - 12.0)
    epslon(k) = 7.8526e-5*tf^(-2.9556e-1);
  end
  if APB_sal > 0.172*5000*(14.25 - 12.0) % Interpolação
```

```
                epslon(k) = 0;
            end
            if APB_sal > 0.172*5000*(14.5 - 12.0)
                epslon(k) = -7.2487e-5;
            end
            if APB_sal > 0.172*5000*(14.75 - 12.0) % Interpolação
                epslon(k) = ((-7.2487e-5) + (-1.5727e-4))/2;
            end
            if APB_sal > 0.172*5000*(15.0 - 12.0)
                epslon(k) = -1.5727e-4;
            end
            if APB_sal > 0.172*5000*(15.25 - 12.0) % Interpolação
                epslon(k) = ((-1.5727e-4) + (-3.4906e-4))/2;
            end
            if APB_sal > 0.172*5000*(15.5 - 12.0)
                epslon(k) = -3.4906e-4;
            end
            if APB_sal > 0.172*5000*(15.75 - 12.0) % Interpolação
                epslon(k) = ((-3.4906e-4) + (-5.4084e-4))/2;
            end
            if APB_sal > 0.172*5000*(16.0 - 12.0)
                epslon(k) = -5.4084e-4;
            end
            desl_sal_D(k,m) = epslon(k)*0.0254*(tf-ti); %[m/d]
deslocamento no intervalo de tempo do passo da simulação
          end
        end
      end
    end
    %% Cálculo do deslocamento acumulado e do raio em função do passo
de tempo.
    for k = 4150:dk:5000
```

```
desl_sal(k,m) = desl_sal_D(k,m)./2; % [m/d] deslocamento no intervalo de tempo "m" do passo da simulação. Este dado é exportado para soma na próxima iteração.
desl_salT(k) = sum(desl_sal(k,:)); % Deslocamento do sal acumulada das iterações no tempo [m].
a_sal(n(s)+1,k) = af(s) - desl_salT(k); % Raio do sal com a deformação acumulada [m];
raio_sal(k,m) = a_sal(n(s)+1,k); % Raio do sal com a deformação acumulada até m para armazenamento [m];
        end
        %% Cálculo da variação de volume pelo fechamento do sal
        for s = 1:1:4
          for i = 1:1:n(s) % anular 2 com sal na seção 3
            for k = topo(s):dk:base(s)
              if cc(s) == 2
                Va_fl_sal_0(n(s),k) = Va_fl_0(n(s),k);
                Va_fl_sal(n(s),k) = pi.*(a_sal(n(s)+1,k).^2 - b(n(s)).^2)*dk; % Cálculo do volume inicial dos anulares em [m3]
                ele(n(s),k) = (k+1)-(k);
                DVa_sal(n(s),k) = Va_fl_sal(n(s),k) - Va_fl_sal_0(n(s),k); % [m3]
               end
             end
          end
        end
        eleT_sal(2) = sum(ele(2,:))-1;
        VaT_fl_sal_0(2) = sum(Va_fl_sal_0(2,:));
        VaT_fl_sal(2) = sum(Va_fl_sal(2,:));
        DVaT_sal = sum(DVa_sal(2,:));
      end
```

B.6. LameFlex

```
function [DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc, af, DVaT_sal, T1, T2R)
dk = 1;
APB(4) = 0;
APB = APB'*101.3e3/14.7/1e9; %converde de [psi] para [GPa]
APB_col = APB_col*101.3e3/14.7/1e9; %converde de [psi] para [GPa]
%% Alocação de memória
F = zeros(3,4,5400); G = zeros(3,4,5400); F_col = zeros(3,5400); FT_col = zeros(3); FT = zeros(3,4); GT = zeros(3,4); ele = zeros(3,5400);
eleT = zeros(1,3); DVa_col = zeros(3,5400); DVaT_T = zeros(3); DVa_ele = zeros(3);
% Constantes do processo para o aço
E = 210; %[GPa]
ni = 0.3;
alfa = 14e-6; %Coeficiente de expansão térmica do aço em ºC^(-1)
% Constantes do processo para o cimento
Ef = 10.9; %[GPa] 10.9
nif = 0.36;  %0.188;
alfa_f = 0;
% Constantes do processo para o sal
Es = 10.9; %[GPa] 20.4
nis = 0.36; % 0.36
%% Cálculo para DVa_cop
   for s = 1:1:4
      for i = 1
         for k = topo(s):dk:base(s)
            F(i,i,k) = (2.*pi.*dk./E).*(a(i+1).^2./(b(i+1).^2 - a(i+1).^2)).*(a(i+1).^2.*(1 - ni) + b(i+1).^2.*(1 + ni)) + (2.*pi.*dk./E).*(b(i).^2./(b(i).^2 - a(i).^2)).*(b(i).^2.*(1 - ni) + a(i).^2.*(1 + ni)); % Eq.A10
```

```
                F(i,i+1,k) = -(4.*pi.*dk./E)*(b(i+1).^2.*a(i+1).^2)./(b(i+1).^2 -
a(i+1).^2);
                G(i,i,k) = (-2.*pi.*dk.*alfa.*b(i).^2).*(T2R(i,k) - T1(i,k));
                G(i,i+1,k) = (2.*pi.*dk.*alfa.*a(i+1).^2 + pi.*(a(i+1).^2 -
b(i).^2).*dk.*alfa).*(T2R(i+1,k) - T1(i+1,k));
                % Cálculo da variação de volume da componente relativo a
APB_col
                F_col(i,k) = -(4.*pi.*dk./E)*(b(i).^2.*a(i).^2)./(b(i).^2 - a(i).^2);
                DVa_col(i,k) = F_col(i,k).*APB_col(k);
                ele(i,k) = (k+1)-(k);
              end
            end
          end
          for i = 1;
            FT_col(i) = sum(F_col(i,:));
            FT(i,i) = sum(F(i,i,:));
            FT(i,i+1) = sum(F(i,i+1,:));
            DVaT_col(i) = sum(DVa_col(i,:));
            eleT(i) = sum(ele(i,:))-1; %Total de elementos que subdividirão a
variação de volume do anular referente a cop
          end
        %% Cálculo dos anulares
          for s = 1:1:4
            for i = 2:1:n(s); % Anulares exceto coluna de produção
              for k = topo(s):dk:base(s)
                F(i,i-1,k) = -(4.*pi.*dk./E)*(b(i).^2.*a(i).^2)./(b(i).^2 - a(i).^2);
                F(i,i,k) = (2.*pi.*dk./E).*(a(i+1).^2./(b(i+1).^2 -
a(i+1).^2)).*(a(i+1).^2.*(1 - ni) + b(i+1).^2.*(1 + ni)) +
(2.*pi.*dk./E).*(b(i).^2./(b(i).^2 - a(i).^2)).*(b(i).^2.*(1 - ni) + a(i).^2.*(1 + ni));
% Eq.A10
                F(i,i+1,k) = -(4.*pi.*dk./E)*(b(i+1).^2.*a(i+1).^2)./(b(i+1).^2 -
a(i+1).^2);
                G(i,i,k) = (-2.*pi.*dk.*alfa.*b(i).^2).*(T2R(i,k) - T1(i,k));
```

```
                G(i,i+1,k) = (2.*pi.*dk.*alfa.*a(i+1).^2 + pi.*(a(i+1).^2 -
b(i).^2).*dk.*alfa).*(T2R(i+1,k) - T1(i+1,k));
                ele(i,k) = (k+1)-(k);
                % Condições de contorno para o último anular adjacente à
formação cimentada
                Hf = 2.*a(n(s)+1).^2./((1 + nif).*E.*(b(n(s)+1).^2 -
a(n(s)+1).^2)./Ef + b(n(s)+1).^2.*(1 - ni) + a(n(s)+1).^2.*(1 + ni));
                F(n(s),n(s)+1,k) = -
(4.*pi.*dk./E)*(b(n(s)+1).^2.*a(n(s)+1).^2)./(b(n(s)+1).^2 - a(n(s)+1).^2).*Hf;
                F(n(s),n(s),k) = F(n(s),n(s),k) + F(n(s),n(s)+1,k); % Soma os
termos dependentes de Hf.
                F(n(s),n(s)+1,k) = 0; %Apenas para zerar termo inutil da
LameFlex
                if cc(s) == 2 % Condições de contorno para o último anular
adjacente à formação com fluido no poço aberto
                    a(n(s)+1) = af(s);
                    F(n(s),n(s)-1,k) = -
(4.*pi.*dk./E)*(b(n(s)).^2.*a(n(s)).^2)./(b(n(s)).^2 - a(n(s)).^2);
                    F(n(s),n(s),k) = (2.*pi.*dk./Es).*(a(n(s)+1).^2.*(1 + nis)) +
(2.*pi.*dk./E).*(b(n(s)).^2./(b(n(s)).^2 - a(n(s)).^2)).*(b(n(s)).^2.*(1 - ni) +
a(n(s)).^2.*(1 + ni)); % Eq.A10
                    F(n(s),n(s)+1,k) = 0;
                    G(n(s),n(s)+1,k) = (2.*pi.*dk.*alfa.*a(n(s)+1).^2 +
pi.*(a(n(s)+1).^2 - b(i).^2).*dk.*alfa).*(T2R(n(s)+1,k) - T1(n(s)+1,k));
                end
            end
        end
    end
    %% Montagem da matriz de rigidez e cálculo de DVaT
    % Soma dos termos de rigidez F
    for i = 2:1:3
        FT(i,i-1) = sum(F(i,i-1,:));
        FT(i,i) = sum(F(i,i,:));
        FT(i,i+1) = sum(F(i,i+1,:));
```

```
    eleT(i) = sum(ele(i,:))-1;
  end
  % Soma dos termos de dilatação térmica;
  for i = 1:1:3
    GT(i,i) = sum(G(i,i,:));
    GT(i,i+1) = sum(G(i,i+1,:));
    DVaT_T(i) = GT(i,i) + GT(i,i+1);
  end
  % Cálculo da APB
  DVaT_Flex = FT*APB; % Calcula a variação de volume no anular do revestimento
  % Cálculo de DVaT
  DVaT(1) = DVaT_Flex(1) + DVaT_T(1) + DVaT_col(1); % Soma a variação de volume devido a parcela da cop
  DVaT(2) = DVaT_Flex(2) + DVaT_T(2) + DVaT_sal; % Soma a variação de volume devido a parcela do fechamento do sal no anular 2
  DVaT(3) = DVaT_Flex(3) + DVaT_T(3);
  for i = 1:3
  DVa_ele(i) = DVaT(i)/eleT(i); % divide a variação de volume pelo numero de elementos para utilizara na convergência
  end
end
```

B.7.
Bmassa

```
function [DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0, mT_0)
fc = 0.453592/3.78541e-3; %converte [lb/gal] para [Kg/m3]
dk = 1;
  %% Cálculo da variação de volume de fluido do anular para comparação com DVaT:
    for s = 1:1:4
      for i = 1:1:n(s);
        for k = topo(s):dk:base(s)
          d_rho(i,k) = (rho_f2(i,k) - rho_f1(i,k))./rho_f2(i,k); % Cálculo das variações de massas específicas adimensional
          dVa_fl(i,k) = -Va_fl_0(i,k).*d_rho(i,k); % Cálculo das variações de volume de fluidos em [m3]
        end
        DVaT_fl(i) = sum(dVa_fl(i,:)); % Cálculo dos volumes totais dos anulares em [m3]
      end
    end
    DVaT_fl; % Sem valor, apenas para verificação
    %% Cálculo da massa final por elemento e total feito por segmento k:
    for s = 1:1:4
      for i = 1:1:n(s);
        for k = topo(s):dk:base(s)
          Va_fl(i,k) = (Va_fl_0(i,k) + DVa_ele(i));
          m_f(i,k) = rho_f2(i,k)*fc*Va_fl(i,k); % [kg/m3]
        end
        VaT_fl(i) = sum(Va_fl(i,:)); % Calcula o volume de fluido na condição 2 com o sal;
        mT_f(i) = sum(m_f(i,:)); % Cálculo da massa inicial dos anulares em [kg/m3]
      end
```

```
    end
    DTm = mT_f - mT_0;
end
```

B.8.
APBsal

```
clear;clc;

%% Pré processamento: Dados de entrada para simulaçaõ de fluência:
tic; % Abertura da contagem de tempo da simulação;
% Gerador do vetor tempo
d = 360; % Tempo em dias
passo = 24; % Passo da simulação em [h/dia]
for m = 1:1:d
   t(m) = m*passo; %[h]
   dias(m) = t(m)./24;
end

% Valores para inicialização das simulações do sal
desl_sal = 0; raio_sal = 0; DVaT_sal = 0;
% Geometria do poço:
[a, b, topo, base, n, cc, af] = Geometria();
% Cálculo do perfil de temperatura nos anulares:
[T1, T2, T2R, T2_col, Tg] = Temperatura(topo, base); % [ºC]
% Cálculo dos perfis de massa específica e pressão na condição 1:
[rho_o1, rho_w1, rho_f1, P1, P1_HC1, m_0, mT_0, Va_fl_0, VaT_fl_0] =
PVT1(topo, base, a, af, cc, b, n, T1, Tg);

%% Processamento: Cálculo do volume do anular com fluência do sal em
função do tempo;
   desalt = 1; % --> Importante preenchimento
   %  1 --> Desativa perfil termico com perdas de carga de produção para
simulação de fluência do sal APB térmica
   %  0 --> Mantém ativado perfil térmico com perdas de carga de produção
ativadas por dafault e calcula APB térmica
   P1_col = P1(1,:);
   if desalt == 1
```

```
        T2 = T1;
        T2R = T1;
        T2_col = Tg;
        P1_col = P1_HC1;
    end
    tempo_pre_processamento = toc; % Contador de tempo para pré-processamento;

    %% Simulação em função do passo de tempo.
    for m = 1:1:d % Simulação de APB por fluência do sal
        m; % Controle do passo tempo
        % Método de Bisseção para o cálculo da APB
        APB_L = [0 0 0]; APB_R = [10000 10000 10000]; % Alfa de inicialização;
        APB_M = (APB_L + APB_R)./2; % Alfa médio
        APB_a = (APB_L + APB_M)./2; % Alfa 1/4 -> esquerdo
        APB_b = (APB_M + APB_R)./2; % Alfa 3/4 -> direito

        APB = APB_a; % Inicialização para função objetivo f1
        % Cálculo dos perfis de massa específica e pressão na condição 2:
        [rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB, topo, base, n, T2, T2_col);
        % Variação de volume do anular em função do fechamento do sal
        [raio_sal, desl_sal, DVaT_sal] = DVsal(t, raio_sal, desl_sal, Va_fl_0, APB, af, b, topo, base, cc, n, m);
        % Cálculo das constantes da matriz de rigidez e da variação de volume dos anulares para condição estimada:
        [DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc, af, DVaT_sal, T1, T2R);
        % Compara a diferença de volume DVaT e DVaT_fl através do balanço de conservação de massa:
        [DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0, mT_0);
        DTm_a = DTm; % Resultado da função objetivo f1
```

```
APB = APB_b; % Inicialização para função f2
% Cálculo dos perfis de massa específica e pressão na condição 2:
[rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB, topo, base, n, T2, T2_col);
% Variação de volume do anular em função do fechamento do sal
[raio_sal, desl_sal, DVaT_sal] = DVsal(t, raio_sal, desl_sal, Va_fl_0, APB, af, b, topo, base, cc, n, m);
% Cálculo das constantes da matriz de rigidez e da variação de volume dos anulares para condição estimada:
[DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc, af, DVaT_sal, T1, T2R);
% Compara a diferença de volume DVaT e DVaT_fl através do balanço de conservação de massa:
[DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0, mT_0);
DTm_b = DTm;  % Resultado da função objetivo f2

%% Ciclo iterativo para convergência do balanço de massa através do ajuste de pressão em LameFlex
Erro_massa = 1; iter_max = 100; j = 1; % Parametros para iteração para função while
Erro_pressao = 1; delta_P = 200; correc_dir = 0; % Parâmetros de correção para função bisseção

while norm(DTm_b) > Erro_massa && j < iter_max
   tic;
   % Avaliação do intervalo de busca para próxima busca por bissecção
   for i = 1:3
      if (norm(DTm_b(i)) > norm(DTm_a(i)))
         APB_R(i) = APB_M(i);
      else
         APB_L(i) = APB_M(i);
      end
```

```
        APB_M(i) = (APB_L(i) + APB_R(i))./2; % Alfa médio
        APB_a(i) = (APB_L(i) + APB_M(i))./2; % Alfa esquerdo
        APB_b(i) = (APB_M(i) + APB_R(i))./2; % Alfa direito
    end

    % Cálculo da função f1
    APB = APB_a;
    % Cálculo dos perfis de massa específica e pressão na condição 2:
    [rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB,
topo, base, n, T2, T2_col);
    % Variação de volume do anular em função do fechamento do sal
    [raio_sal, desl_sal, DVaT_sal] = DVsal(t, raio_sal, desl_sal, Va_fl_0,
APB, af, b, topo, base, cc, n, m);
    % Cálculo das constantes da matriz de rigidez e da variação de volume
dos anulares para condição estimada:
    [DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc,
af, DVaT_sal, T1, T2R);
    % Compara a diferença de volume DVaT e DVaT_fl através do
balanço de conservação de massa:
    [DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0,
mT_0);
    DTm_a = DTm;

    % Cálculo da função f2
    APB = APB_b;
    % Cálculo dos perfis de massa específica e pressão na condição 2:
    [rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB,
topo, base, n, T2, T2_col);
    % Variação de volume do anular em função do fechamento do sal
    [raio_sal, desl_sal, DVaT_sal] = DVsal(t, raio_sal, desl_sal, Va_fl_0,
APB, af, b, topo, base, cc, n, m);
    % Cálculo das constantes da matriz de rigidez e da variação de volume
dos anulares para condição estimada:
```

```
[DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc, af, DVaT_sal, T1, T2R);
% Compara a diferença de volume DVaT e DVaT_fl através do balanço de conservação de massa:
[DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0, mT_0);
DTm_b = DTm;

%% Rotina de cálculo para correção da direção de busca --> Novo intervalo de bissecção
if norm(APB_a - APB_b) < Erro_pressao
   correc_dir = 1 + correc_dir;
   delta_P = delta_P./correc_dir.^3;
   if correc_dir > 4
      delta_P = 250;
   end
   APB_R = APB_R + delta_P;
   APB_L = APB_L - delta_P;
end
tempo_j = toc; % Contagem do tempo iteração j
controle_iteracao = [m, j, correc_dir, tempo_j];
convergencia_iteracao = [APB_a; APB_b; DTm_a; DTm_b];
tempo_iteracao(j) = tempo_j; % Armazenamento da variável simulada em j;
j = j+1; %Contador do ciclo de iteração
end

tempo(m) = sum(tempo_iteracao(:)); % Contagem do tempo de simulação do passo m
APBt(m,1) = APB(1); APBt(m,2) = APB(2); APBt(m,3) = APB(3); %Evolução da APB no anular
DTmt(m,1) = DTm(1); DTmt(m,2) = DTm(2); DTmt(m,3) = DTm(3); %Evolução da APB no anular
```

```
        DVaTt(m,1) = DVaT(1); DVaTt(m,2) = DVaT(2); DVaTt(m,3) = DVaT(3); %Evolução DVaT;
        DVaTt_sal(m) = [DVaT_sal]; %Evolução DVaT_sal;
        controle(m,1) = dias(m); controle(m,2) = j; controle(m,3) = tempo(m); controle(m,4) = correc_dir; controle(m,5) = norm(DTm_b);
        controle_APBt = [controle(m,:), APBt(m,:)/1000] %Acompanhamento por etapa simulada
    end

    controle_APBt = [controle, APBt/1000]
    Tempo_Total_Simulacao = sum(tempo(:))/60 % min
    Sal_Pontos_Notaveis_a = [controle(:,1)'; raio_sal(4200,:); raio_sal(4400,:); raio_sal(4600,:); raio_sal(4800,:); raio_sal(5000,:)];
    Sal_Pontos_Notaveis_def = [controle(:,1)'; desl_sal(4200,:); desl_sal(4400,:); desl_sal(4600,:); desl_sal(4800,:); desl_sal(5000,:)];
```

B.9.
APBsal_secundário

% A rotina $_{APBsal}$_secundaria é simualada após o APBsal para obter o APB acoplado com a carga térmica através da variação de volume na seção em que há fluência do evaporito. Rodar este módulo após simular fluência do sal sem módulo térmico

%% Alocação de memória e reset parâmetros.

APBt = zeros(d,3); DVaTt = zeros(d,3); DTmt = zeros(d,3); controle = zeros(360,5);

%% Processamento da APB térmica somada a APB do sal

for m = 7:7:d % Simulação de APB após fluência do sal

m; % Controle do passo tempo

tic; % Abertura da contagem de tempo da iteração de m

P1_col = P1(1,:); %Simula APB_col para produção do poço.

% Ativação do perfil térmico e do histórico variação de volume do anular em função do sal

[T1, T2, T2R, T2_col, Tg] = Temperatura(topo, base); % [ºC]

DVaT_sal = DVaTt_sal(m); % Único parâmetro importado da planilha APB_sal_Primária

% Método de Bisseção para o cálculo da APB

APB_L = [0 0 0]; APB_R = [10000 10000 10000]; % Alfa de inicialização;

APB_M = (APB_L + APB_R)./2; % Alfa médio

APB_a = (APB_L + APB_M)./2; % Alfa 1/4 -> esquerdo

APB_b = (APB_M + APB_R)./2; % Alfa 3/4 -> direito

APB = APB_a; % Inicialização para função objetivo f1

% Cálculo dos perfis de massa específica e pressão na condição 2:

[rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB, topo, base, n, T2, T2_col);

% Cálculo das constantes da matriz de rigidez e da variação de volume dos anulares para condição estimada:

```
[DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc, af, DVaT_sal, T1, T2R);
% Compara a diferença de volume DVaT e DVaT_fl através do balanço de conservação de massa:
[DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0, mT_0);
DTm_a = DTm; % Resultado da função objetivo f1

APB = APB_b; % Inicialização para função f2
% Cálculo dos perfis de massa específica e pressão na condição 2:
[rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB, topo, base, n, T2, T2_col);
% Cálculo das constantes da matriz de rigidez e da variação de volume dos anulares para condição estimada:
[DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc, af, DVaT_sal, T1, T2R);
% Compara a diferença de volume DVaT e DVaT_fl através do balanço de conservação de massa:
[DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0, mT_0);
DTm_b = DTm;  % Resultado da função objetivo f2

%% Ciclo iterativo para convergência do balanço de massa através do ajuste de pressão em LameFlex
Erro_massa = 1; iter_max = 100; j = 1; % Parametros para iteração para função while
Erro_pressao = 1; delta_P = 200; correc_bis = 0; % Parâmetros de correção para função bisseção
while norm(DTm_b) > Erro_massa && j < iter_max
tic;
   % Avaliação do intervalo de busca para próxima busca por bissecção
   for i = 1:3
      if (norm(DTm_b(i)) > norm(DTm_a(i)))
         APB_R(i) = APB_M(i);
```

```
            else
               APB_L(i) = APB_M(i);
            end
            APB_M(i) = (APB_L(i) + APB_R(i))./2; % Alfa médio
            APB_a(i) = (APB_L(i) + APB_M(i))./2; % Alfa esquerdo
            APB_b(i) = (APB_M(i) + APB_R(i))./2; % Alfa direito
         end
         % Cálculo da função f1
         APB = APB_a;
         % Cálculo dos perfis de massa específica e pressão na condição 2:
         [rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB,
topo, base, n, T2, T2_col);
         % Cálculo das constantes da matriz de rigidez e da variação de volume
dos anulares para condição estimada:
         [DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc,
af, DVaT_sal, T1, T2R);
         % Compara a diferença de volume DVaT e DVaT_fl através do
balanço de conservação de massa:
         [DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0,
mT_0);
         DTm_a = DTm;
         % Cálculo da função f2
         APB = APB_b;
         % Cálculo dos perfis de massa específica e pressão na condição 2:
         [rho_f2, APB_col] = PVT2(rho_o1, rho_w1, rho_f1, P1, P1_col, APB,
topo, base, n, T2, T2_col);
         % Cálculo das constantes da matriz de rigidez e da variação de volume
dos anulares para condição estimada:
         [DVa_ele, DVaT] = LameFlex(APB_col, APB, a, b, topo, base, n, cc,
af, DVaT_sal, T1, T2R);
         % Compara a diferença de volume DVaT e DVaT_fl através do
balanço de conservação de massa:
         [DTm] = BMassa(rho_f1, rho_f2, n, topo, base, DVa_ele, Va_fl_0,
mT_0);
```

```
            DTm_b = DTm;
            %% Rotina de cálculo para correção da direção de busca --> Novo
intervalo de bissecção
            if norm(APB_a - APB_b) < Erro_pressao
               correc_bis = 1 + correc_bis;
               delta_P = delta_P./correc_bis.^3;
               if correc_bis == 5
                  delta_P = 250;
               end
               APB_R = APB_R + delta_P;
               APB_L = APB_L - delta_P;
            end
            tempo_j = toc; % Contagem do tempo iteração j
            controle_iteracao = [m, j, correc_dir, tempo_j];
            convergencia_iteracao = [APB; DTm];
            tempo_iteracao(j) = tempo_j; % Armazenamento da variável simulada
em j;
            j = j+1; %Contador do ciclo de iteração
         end


         tempo(m) = sum(tempo_iteracao(:)); % Contagem do tempo de simulação
do passo m
         APBt(m,1) = APB(1); APBt(m,2) = APB(2); APBt(m,3) = APB(3);
%Evolução da APB no anular
         DTmt(m,1) = DTm(1); DTmt(m,2) = DTm(2); DTmt(m,3) = DTm(3);
%Evolução da APB no anular
         DVaTt(m,1) = DVaT(1); DVaTt(m,2) = DVaT(2); DVaTt(m,3) =
DVaT(3); %Evolução DVaT;
         DVaTt_sal(m) = [DVaT_sal]; %Evolução DVaT_sal;
         controle(m,1) = dias(m); controle(m,2) = j; controle(m,3) = tempo(m);
controle(m,4) = correc_dir; controle(m,5) = norm(DTm_b);
         controle_APBt = [controle(m,:), APBt(m,:)] %Acompanhamento por
etapa simulada
      end
```