Capítulo 1

Implementação

Neste capítulo do projeto de engenharia apresentamos os códigos fonte que foram desenvolvidos.

Nota: os códigos devem ser documentados usando padrão **javadoc**. Posteriormente usar o programa **doxygen** para gerar a documentação no formato html.

- Veja informações gerais aqui http://www.doxygen.org/.
- Veja exemplo aqui http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html.

Nota: ao longo deste capítulo usamos inclusão direta de arquivos externos usando o pacote *listings* do LaTeX. Maiores detalhes de como a saída pode ser gerada estão disponíveis nos links abaixo.

- http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings.
- http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf.

1.1 Código fonte

Apresenta-se a seguir um conjunto de classes (arquivos .h e .cpp) além do programa `main`.

Apresenta-se na listagem 1.1 o arquivo com código da classe CFluidorBlackOil.

Listing 1.1: Arquivo de cabeçalho da classe CFluidoBlackOil.

```
1#ifndef CFluidoBlackOil_h
2#define CFluidoBlackOil_h
3
4#include <iostream>
5#include <fstream>
6#include <vector>
7#include <string>
8
```

```
9 /**
10 @brief Classe que representa as caracteristicas do fluido blackoil
11 @class CFluidoBlackOil
12 @file CFluidoBlackOil.h
13  */
14 class CFluidoBlackOil {
15
16 ///Atributos
17 public:
18     /// Grau API do oleo
19     double api;
20     /// Temperatura do fluido
21     double temp;
22     /// Razao de solubilidade do gas no oleo no ponto de bolha
23     double rsb;
24     /// Pressao no ponto desejado
25     double p;
26     /// Pressao maxima do reservatorio
27     double pmax;
28     /// Variacao de pressao desejada
29     double deltap;
30     /// Densidade do gas no separador
31     double dgs;
32     /// Densidade do gas
33     double dg;
34     /// Densidade do oleo
35     double d_o;
36     /// Vetor com as pressoes
37     std::vector<double> vpressao;
38
39 ///Metodos
40 public:
41     ///Construtor default
42     CFluidoBlackOil(){};
43     ///Destrutor
44     ~CFluidoBlackOil (){};
45
46     /// Metodo para criar o vetor de pressoes
47    std::vector<double> Gera_vp (double p, double pmax, double deltap);
48
49     /// Metodos set e get
50     double P(){    return p;    }
51     void P(double _p){        p=_p;    }
52
53     double PMax(){    return pmax;    }
54     void PMax(double _pm){        pmax=_pm;    }
55
56     double API(){        return api;    }
```

```
57     void API(double _API){         api=_API;     }
58
59     double Temp(){          return temp;     }
60     void Temp(double _Temp){          temp=_Temp;     }
61
62     double Rsb(){          return rsb;     }
63     void Rsb(double _Rsb){          rsb=_Rsb;     }
64
65     double Dg(){          return dg;     }
66     void Dg(double _Dg){          dg=_Dg;     }
67
68     double Dgs(){          return dgs;     }
69     void Dgs(double _Dgs){          dgs=_Dgs;     }
70
71     double D_o(){          return d_o;     }
72     void D_o(double _d_o){          d_o=_d_o;     }
73
74     /// Sobrecarga de operadores >>, entrada de dados pelo disco e
          teclado
75     friend std::istream& operator >> (std::istream& in, CFluidoBlackOil&
           obj);
76     friend std::ifstream& operator >> (std::ifstream& fin,
          CFluidoBlackOil& obj);
77
78 };
79 #endif // CFluidoBlackOil_h
```

Apresenta-se na listagem 1.2 o arquivo de implementação da classe CFluidoBlackOil.

Listing 1.2: Arquivo de implementação da classe CFluidoBlackOil.

```
80 #include <iostream>
81 #include <fstream>
82
83 #include "CFluidoBlackOil.h"
84
85 /**
86 @brief Classe que representa as caracteristicas do fluido blackoil
87 @class CFluidoBlackOil
88 @file CFluidoBlackOil.cpp
89  */
90
91 using namespace std;
92
93 ///Sobrecarga do operador >>, entrada de dados pelo teclado
94 istream& operator >> (istream& in, CFluidoBlackOil& obj){
95
96     cout << "Entre␣com␣os␣valores␣das␣variaveis␣conforme␣sequencia:\n";
97     cout << "␣1␣-␣Grau␣API␣\n";
98     in >> obj.api; in.get();
```

```
99      cout << "␣\n2␣-␣Temperatura␣no␣Reservatorio␣(em␣Fahrenheit)␣\n␣";
100     in >> obj.temp; in.get();
101     cout << "␣\n3␣-␣Razao␣de␣Solubilidade␣no␣ponto␣de␣bolha␣=␣Rsb␣(em␣
            scf/STB)␣\n";
102     in >> obj.rsb; in.get();
103     cout << "␣\n4␣-␣Pressao␣do␣ponto␣desejado␣(em␣Psia)␣(caso␣queira␣
            calcular␣de␣varios␣pontos␣digitar␣0␣para␣essa␣variavel)␣\n";
104     in >> obj.p; in.get();
105     cout << "␣\n5␣-␣Pressao␣maxima␣do␣reservatorio␣(caso␣queira␣calcular
            ␣em␣um␣unico␣ponto␣digitar␣0␣para␣essa␣variavel)␣\n";
106     in >> obj.pmax; in.get();
107     cout << "␣\n6␣-␣Variacao␣de␣pressao␣desejada␣para␣acrescimo␣de␣
            pressao␣\n";
108     in >> obj.deltap; in.get();
109     cout << "␣\n7␣-␣Densidade␣do␣gas␣no␣separador␣\n";
110     in >> obj.dgs; in.get();
111     cout << "␣\n8␣-␣Densidade␣do␣gas␣\n";
112     in >> obj.dg; in.get();
113     cout << "␣\n9␣-␣Densidade␣do␣oleo\n";
114     in >> obj.d_o; in.get();
115
116     return in;
117 }
118 ///Sobrecarga do operador >>, entrada de dados pelo disco
119 ifstream& operator >> (ifstream& fin, CFluidoBlackOil& obj){
120
121     fin.ignore(500,'\n');
122     fin>>obj.p;
123     fin.ignore(500,'\n');
124     fin>>obj.pmax;
125     fin.ignore(500,'\n');
126     fin>>obj.deltap;
127     fin.ignore(500,'\n');
128     fin>>obj.api;
129     fin.ignore(500,'\n');
130     fin>>obj.dg;
131     fin.ignore(500,'\n');
132     fin>>obj.temp;
133     fin.ignore(500,'\n');
134     fin>> obj.dgs;
135     fin.ignore (500,'\n');
136     fin>>obj.rsb;
137     fin.ignore(500,'\n');
138     fin>>obj.d_o;
139     fin.close();
140     return fin;
141 }
142 /// Metodo para criar o vetor de pressoes
```

```
143 vector<double> CFluidoBlackOil :: Gera_vp(double p, double pmax, double
      deltap) {
144
145 /// Caso o calculo seja para um vetor de pressoes
146 if (p==0.0) {
147     for(double i =14.7; i<= pmax; i += deltap) {
148        vpressao.push_back(i);
149         }
150     }
151 /// Caso o calculo seja para um unico de valor de pressao
152 if(pmax==0.0 && deltap ==0.0){
153         vpressao.push_back(p);
154         }
155 /// Teste e instrucoes para entrada de dados
156 if (((p!=0.0) && (pmax!=0.0) && (deltap!=0.0)) || ((p==0.0) && (pmax
      ==0.0) && (deltap==0.0)))
157         {
158          cout << "\nDados␣inseridos␣incorretamente!␣\nFavor␣digitar␣0␣e␣
                verificar␣regras␣para␣calculo␣em␣um␣ponto␣ou␣em␣vetor.\n";
159          cout << "-␣Caso␣queira␣calcular␣os␣parametros␣em␣um␣unico␣valor
                ␣de␣pressao,\ninserir␣0␣em␣pressao␣maxima␣e␣0␣em␣variacao␣de
                ␣pressao.\n";
160          cout << "-␣Caso␣queira␣calcular␣os␣parametros␣em␣diversos␣
                valores␣de␣pressao␣inserir␣0␣em␣Pressao␣do␣ponto␣desejado.\n
                " << endl;
161          }
162     return vpressao;
163 }
```

Apresenta-se na listagem 1.3 o arquivo com código da classe C3Parametros.

Listing 1.3: Arquivo de cabeçalho da classe C3Parametros.

```
164 #ifndef C3Parametros_h
165 #define C3Parametros_h
166
167 /**
168 @brief Classe responsavel pelos parametros da correlacao Vasquez e Beggs
169 @class C3Parametros
170 @file C3Parametros.h
171  */
172 class C3Parametros{
173 ///Atributos
174 protected:
175     ///Parametros
176     double c1, c2, c3;
177
178 /// Metodos set e get
179 public:
180     void C1(double _c1){
```

```
181         c1=_c1;
182     };
183     double C1(){
184         return c1;
185     };
186     void C2(double _c2){
187         c2=_c2;
188     };
189     double C2(){
190         return c2;
191     };
192     void C3(double _c3){
193         c3=_c3;
194     };
195     double C3(){
196         return c3;
197     };
198 
199 };
200 #endif
```

Apresenta-se na listagem 1.4 o arquivo de implementação da classe C3Parametros.

Listing 1.4: Arquivo de implementação da classe C3Parametros.

```
202 #include "C3Parametros.h"
```

Apresenta-se na listagem 1.5 o arquivo com código da classe CPressaoBolha.

Listing 1.5: Arquivo de cabeçalho da classe CPressaoBolha.

```
204 #ifndef CPressaoBolha_h
205 #define CPressaoBolha_h
206 
207 #include "CFluidoBlackOil.h"
208 
209 /**
210 @brief Classe responsavel pelo calculo da Pressao de bolha
211 @class CPressaoBolha
212 @file CPressaoBolha.h
213  */
214 class CPressaoBolha  {
215 
216 ///Atributos
217 public:
218     /// Pressao de bolha
219     double pb;
220 
221  ///Metodos
222 public:
223     ///Construtor default
```

```
224     CPressaoBolha  () {};
225     ///Destrutor
226     ~CPressaoBolha () {};
227 
228     /// Metodos get e set
229     void Pb(double _pb) {pb = _pb;}
230     double Pb()         {return pb;}
231 
232     /// Metodo para calculo da pressão de bolha
233     double Pb(CFluidoBlackOil&) { return pb; }
234 
235 };
236 
237 #endif // CPressaoBolha_h
```

Apresenta-se na listagem 1.6 o arquivo de implementação da classe CPressaoBolha.

Listing 1.6: Arquivo de implementação da classe CPressaoBolha.

```
238 #include "CPressaoBolha.h"
```

Apresenta-se na listagem 1.7 o arquivo com código da classe CPb_Standing.

Listing 1.7: Arquivo de cabeçalho da classe CPb$_{}standing$.

```
239 #ifndef CPb_Standing_h
240 #define CPb_Standing_h
241 
242 #include "CPressaoBolha.h"
243 #include "CFluidoBlackOil.h"
244 
245 /**
246 @brief Classe responsavel pelo calculo da Pressao de bolha pela
      correlacao Standing
247 @class CPb_Standing
248 @file CPb_Standing.h
249  */
250 class CPb_Standing : public CPressaoBolha ///Herdeira da classe
      CPressaoBolha
251 {
252 
253 ///Metodos
254 public:
255     ///Construtor default
256     CPb_Standing () {};
257     ///Destrutor
258     ~CPb_Standing () {};
259 
260     /// Metodo para calculo da pressao de bolha usando a correlacao
          Standing
261     double Pb(CFluidoBlackOil& fluido);
```

```
262
263 };
264
265 #endif // CPb_Standing_h
```

Apresenta-se na listagem 1.8 o arquivo de implementação da classe CPb_Standing.

Listing 1.8: Arquivo de implementação da classe CPb$_S$*tanding*.

```
266 #include <cmath>
267
268 #include "CPb_Standing.h"
269
270 /**
271 @class CPb_Standing
272 @file CPb_Standing.cpp
273  */
274 using namespace std;
275
276 /// Metodo para calculo da pressao de bolha usando a correlação Standing
277 double CPb_Standing::Pb(CFluidoBlackOil& fluido){
278
279 pb =18.2*((1.0/(pow(10.0,(0.0125*fluido.api-0.00091*fluido.temp)))*(pow(
      fluido.rsb/fluido.dg,0.83)))-1.4);
280
281 return pb;
282
283 }
```

Apresenta-se na listagem 1.9 o arquivo com código da classe CPb_VasquezBeggs.

Listing 1.9: Arquivo de cabeçalho da classe CPb$_V$*asquezBeggs*.

```
285 #ifndef CPb_VasquezBeggs_h
286 #define CPb_VasquezBeggs_h
287
288 #include "CPressaoBolha.h"
289 #include "C3Parametros.h"
290
291 /**
292 @brief Classe responsavel pelo calculo da Pressao de bolha pela
      correlacao Vasquez e Beggs
293 @class CPb_VasquezBeggs
294 @file CPb_VasquezBeggs.h
295  */
296 class CPb_VasquezBeggs: public CPressaoBolha , C3Parametros ///Herdeira
      das classes CPressaBolha e C3Parametros
297 {
298 ///Metodos
299 public:
300     ///Construtor default
```

```
301     CPb_VasquezBeggs() {} ;
302     ///Destrutor
303     ~CPb_VasquezBeggs () {};
304 
305     ///Metodo para calculo da pressao de bolha usando a correlacao
            Vasquez e Beggs
306    double Pb(CFluidoBlackOil& fluido);
307 
308 };
309 
310 #endif // CPb_VasquezBeggs_h
```

Apresenta-se na listagem 1.10 o arquivo de implementação da classe CPb_VasquezBeggs.

Listing 1.10: Arquivo de implementação da classe $\mathrm{CPb}_{V}asquezBeggs$.

```
311 #include <cmath>
312 
313 #include "CPb_VasquezBeggs.h"
314 #include "CSimuladorBlackOil.h"
315 
316 /**
317 @class CPb_VasquezBeggs
318 @file CPb_VasquezBeggs.cpp
319  */
320 using namespace std;
321 
322  /// Metodo para calculo da pressão de bolha usando a correlação Vasquez
        e Beggs
323 double CPb_VasquezBeggs::Pb(CFluidoBlackOil& fluido){
324    /// Seta os parametros da correlacao C3Parametros
325     if (fluido.api <= 30.0){
326         c1 = 0.0362;
327         c2 = 1.0937;
328         c3 = 25.724;
329     }
330     else {
331         c1 = 0.0178;
332         c2 = 1.187;
333         c3 = 23.931;
334     }
335 
336     pb=pow((fluido.rsb/(c1*fluido.dgs* exp((c3*fluido.api)/(fluido.temp
            +459.67)))),(1.0/c2));
337 
338     return pb;
339 }
```

Apresenta-se na listagem 1.11 o arquivo com código da classe CRazaoSolubilidade.

Listing 1.11: Arquivo de cabeçalho da classe CRazaoSolubilidade.

```
340 #ifndef CRazaoSolubilidade_h
341 #define CRazaoSolubilidade_h
342
343 #include <vector>
344
345 #include "CFluidoBlackOil.h"
346 #include "CPressaoBolha.h"
347
348 /**
349 @brief Classe responsavel pelo calculo da Razao de Solubilidade
350 @class CRazaoSolubilidade
351 @file CRazaoSolubilidade.h
352  */
353 class CRazaoSolubilidade {
354
355 ///Atributo
356 public:
357     /// Vetor de Razao de Solubilidade
358     std::vector <double> vrs;
359
360 ///Metodos
361 public:
362     ///Construtor default
363     CRazaoSolubilidade  () {};
364     ///Destrutor
365     ~CRazaoSolubilidade () {};
366
367     /// Metodos get e set
368     void Vrs(std::vector<double> _vrs) {vrs = _vrs;}
369     std::vector<double> Vrs()          {return vrs;}
370
371     /// Metodo para calculo da razao de solubilidade
372     std::vector<double> Rs(CFluidoBlackOil&, CPressaoBolha&) {return vrs
        ; }
373
374     /// Metodo para gerar grafico
375     void Plot(CFluidoBlackOil& fluido);
376
377 };
378 #endif // CRazaoSolubilidade_h
```

Apresenta-se na listagem 1.12 o arquivo de implementação da classe CRazaoSolubilidade.

Listing 1.12: Arquivo de implementação da classe CRazaoSolubilidade.

```
379 #include "CRazaoSolubilidade.h"
380 #include "CGnuplot.h"
```

```
381
382 /**
383 @class CRazaoSolubilidade
384 @file CRazaoSolubilidade.cpp
385  */
386
387 /// Metodo para gerar grafico
388 void CRazaoSolubilidade:: Plot(CFluidoBlackOil& fluido) {
389
390 static CGnuplot g2d;
391 g2d.Title("Pressao␣x␣Razao␣de␣Solubilidade");
392 g2d.XLabel("Pressao(psia)");
393 g2d.YLabel("Rs(scf/STB)");
394 g2d.Style("linespoints");
395 g2d.PlotVector(fluido.vpressao,vrs);
396 }
```

Apresenta-se na listagem 1.13 o arquivo com código da classe CRs_Standing.

Listing 1.13: Arquivo de cabeçalho da classe CRs$_S$*tanding*.

```
397 #ifndef CRs_Standing_h
398 #define CRs_Standing_h
399
400 #include "CRazaoSolubilidade.h"
401
402 /**
403 @brief Classe responsavel pelo calculo da Razao de Solubilidade pela
      correlacao Standing
404 @class CRs_Standing
405 @file CRs_Standing.h
406  */
407 class CRs_Standing : public CRazaoSolubilidade  ///Herdeira da classe
      CRazaoSolubilidade
408 {
409
410 ///Metodos
411 public:
412     ///Construtor default
413     CRs_Standing  () {};
414     ///Destrutor
415     ~CRs_Standing () {};
416
417     /// Metodo para calculo da razao de solubilidade pela correlacao de
          Standing
418     std::vector<double> Rs(CFluidoBlackOil& fluido, CPressaoBolha& cpb);
419 };
420
421 #endif // CRs_Standing_h
```

Apresenta-se na listagem 1.14 o arquivo de implementação da classe CRs_Standing.

Listing 1.14: Arquivo de implementação da classe CRs$_Standing$.

```
422#include <vector>
423#include <cmath>
424
425#include "CRs_Standing.h"
426
427/**
428@class CRs_Standing
429@file CRs_Standing.cpp
430 */
431using namespace std;
432
433 /// Metodo para calculo da razao de solubilidade pela correlacao de
      Standing
434vector<double> CRs_Standing:: Rs(CFluidoBlackOil& fluido, CPressaoBolha&
      cpb){
435     vrs.resize(fluido.vpressao.size()); //mesmo tamanho
436
437/// Para pressoes abaixo do Pb a razao de solubilidade e calculada, para
      pressoes acima do Pb o Rs se mantem constante e igual ao Rsb
438for (int i=0; i<fluido.vpressao.size(); i++)
439     {
440         if (fluido.vpressao[i] <= cpb.pb){
441             vrs[i] = fluido.dg*(pow(((fluido.vpressao[i]/18.2+1.4)*(pow
                  (10.0, (0.0125*fluido.api-0.00091*fluido.temp)))),
                  1.2048));
442             }
443         else
444           vrs[i] = fluido.rsb;
445     }
446     return vrs;
447}
```

Apresenta-se na listagem 1.15 o arquivo com código da classe CRs_VasquezBeggs.

Listing 1.15: Arquivo de cabeçalho da classe CRs$_VasquezBeggs$.

```
448#ifndef CRs_VasquezBeggs_h
449#define CRs_VasquezBeggs_h
450
451#include "CRazaoSolubilidade.h"
452#include "C3Parametros.h"
453
454/**
455@brief Classe responsavel pelo calculo da Razao de solubilidade pela
     correlacao Vasquez e Beggs
456@class CRs_VazquezBeggs
457@file CRs_VasquezBeggs.h
```

```
458 */
459 class CRs_VasquezBeggs: public CRazaoSolubilidade, C3Parametros ///
      Herdeira das classes CRazaoSolubilidade e C3Parametros
460 {
461
462 ///Metodos
463 public:
464     ///Construtor default
465     CRs_VasquezBeggs  () {};
466     ///Destrutor
467     ~CRs_VasquezBeggs () {};
468
469     /// Metodo para calculo da razao de solubilidade pela correlacao de
          Vasquez e beggs
470     std::vector<double> Rs(CFluidoBlackOil& fluido, CPressaoBolha& cpb);
471
472 };
473
474 #endif // CRs_VasquezBeggs_h
```

Apresenta-se na listagem 1.16 o arquivo de implementação da classe CRs_VasquezBeggs.

Listing 1.16: Arquivo de implementação da classe $CRs_VasquezBeggs$.

```
475 #include <vector>
476 #include <cmath>
477
478 #include "CRs_VasquezBeggs.h"
479
480 /**
481 @class CRs_VasquezBeggs
482 @file CRs_VasquezBeggs.cpp
483 */
484 using namespace std;
485
486 /// Metodo para calculo da pressão de bolha usando a correlação Vasquez
      e Beggs
487 vector<double> CRs_VasquezBeggs::Rs(CFluidoBlackOil& fluido,
      CPressaoBolha& cpb){
488     /// Seta os parametros da correlacao C3Parametros
489     if (fluido.api <= 30.0){
490         c1 = 0.0362;
491         c2 = 1.0937;
492         c3 = 25.724;
493     }
494     else {
495         c1 = 0.0178;
496         c2 = 1.187;
497         c3 = 23.931;
498     }
```

```
499
500     vrs.resize(fluido.vpressao.size()); //mesmo tamanho
501
502 for (int i=0; i<fluido.vpressao.size(); i++){
503         /// Para pressoes abaixo do Pb a razao de solubilidade e
                calculada, para pressoes acima do Pb o Rs se mantem constante
                 e igual ao Rsb
504         if (fluido.vpressao[i] <= cpb.pb){
505         vrs[i] = c1*fluido.dgs*pow(fluido.vpressao[i], c2)*exp((c3*
                fluido.api)/(fluido.temp+459.67));
506         }
507        else
508           vrs[i] = fluido.rsb;
509
510     }
511     return vrs;
512 }
```

Apresenta-se na listagem 1.17 o arquivo com código da classe CBoFatorVolFormacao-Oleo.

Listing 1.17: Arquivo de cabeçalho da classe CBoFatorVolFormacaoOleo.

```
513 #ifndef CBoFatorVolFormacaoOleo_h
514 #define CBoFatorVolFormacaoOleo_h
515
516 #include <vector>
517
518 #include "CFluidoBlackOil.h"
519 #include "CRazaoSolubilidade.h"
520 #include "CCompresOleo.h"
521 #include "CPressaoBolha.h"
522
523 /**
524 @brief Classe responsavel pelo calculo do Fator volume de formacao do
      oleo
525 @class CBoFatorVolFormacaoOleo
526 @file CBoFatorVolFormacaoOleo.h
527  */
528 class CBoFatorVolFormacaoOleo{
529
530 ///Atributos
531 public:
532     /// Fator volume de formacao no ponto de bolha
533     double bob;
534     /// Vetor de Fator volume de formacao em outros pontos
535     std::vector<double> vbo;
536
537 ///Metodos
```

```
538 public:
539     ///Construtor default
540     CBoFatorVolFormacaoOleo  (){};
541     ///Destrutor
542     ~CBoFatorVolFormacaoOleo (){};
543 
544     /// Metodos get e set
545     void Bob(double _bob) {bob = _bob;}
546     double Bob()          {return bob;}
547 
548     void Vbo(std::vector<double> _vbo) {vbo = _vbo;}
549     std::vector<double> Vbo()          {return vbo;}
550 
551 
552     /// Metodo para calculo do Bo no ponto de bolha
553     double Bob(CFluidoBlackOil&) { return bob; }
554 
555     /// Metodo para calculo do Bo em varios pontos
556     std::vector<double> Bo(CFluidoBlackOil&, CPressaoBolha&,
        CRazaoSolubilidade&, CCompresOleo&) { return vbo; }
557 
558     /// Metodo para gerar grafico
559     void Plot(CFluidoBlackOil& fluido);
560 };
561 
562 #endif // CBoFatorVolFormacaoOleo_h
```

Apresenta-se na listagem 1.18 o arquivo de implementação da classe CBoFatorVolFormacaoOleo.

Listing 1.18: Arquivo de implementação da classe CBoFatorVolFormacaoOleo.

```
563 #include "CBoFatorVolFormacaoOleo.h"
564 #include "CGnuplot.h"
565 
566 /**
567 @class CBoFatorVolFormacaoOleo
568 @file CBoFatorVolFormacaoOleo.cpp
569  */
570 
571 /// Metodo para gerar grafico
572 void CBoFatorVolFormacaoOleo :: Plot (CFluidoBlackOil& fluido) {
573 
574 static CGnuplot g2d;
575 g2d.Title("Pressao␣x␣Fator␣Volume␣de␣Formacao␣do␣Oleo");
576 g2d.XLabel("Pressao(psia)");
577 g2d.YLabel("Bo(bbl/STB)");
578 g2d.Style("linespoints");
579 g2d.PlotVector(fluido.vpressao,vbo);
```

```
580
581}
```

Apresenta-se na listagem 1.19 o arquivo com código da classe CBo_Standing.

Listing 1.19: Arquivo de cabeçalho da classe CBo$_{S}tanding$.

```
582#ifndef CBo_Standing_h
583#define CBo_Standing_h
584
585#include "CBoFatorVolFormacaoOleo.h"
586
587/**
588@brief Classe responsavel pelo calculo do Fator volume de formacao do
      oleo pela correlacao Standing
589@class CBo_Standing
590@file CBo_Standing.h
591 */
592class CBo_Standing : public CBoFatorVolFormacaoOleo    ///Herdeira da
      classe CFatorVolumeFormacaoOleo
593{
594///Metodos
595public:
596    ///Construtor default
597    CBo_Standing (){};
598    ///Destrutor
599    ~CBo_Standing(){};
600
601    /// Metodo para calculo do Bo no ponto de bolha pela correlacao
          Standing
602    double Bob(CFluidoBlackOil& fluido);
603};
604
605#endif // CBo_Standing_h
```

Apresenta-se na listagem 1.20 o arquivo de implementação da classe CBo_Standing.

Listing 1.20: Arquivo de implementação da classe CBo$_{S}tanding$.

```
606#include <cmath>
607
608#include "CBo_Standing.h"
609
610/**
611@class CBo_Standing
612@file CBo_Standing.cpp
613 */
614using namespace std;
615
616/// Metodo para calculo do Bo no ponto de bolha pela correlacao Standing
617double CBo_Standing::Bob(CFluidoBlackOil& fluido){
```

```
618
619     bob= 0.972 + 1.47*pow(10.0, -4.0)*pow((fluido.rsb*(pow((fluido.dg/
            fluido.d_o), 0.5))+1.25*fluido.temp), 1.175);
620
621     return bob;
622 }
```

Apresenta-se na listagem 1.21 o arquivo com código da classe CBo_VasquezBeggs.

Listing 1.21: Arquivo de cabeçalho da classe $\mathrm{CBo}_{V}asquezBeggs$.

```
624 #ifndef CBo_VasquezBeggs_h
625 #define CBo_VasquezBeggs_h
626
627 #include <vector>
628
629 #include "CBoFatorVolFormacaoOleo.h"
630 #include "C3Parametros.h"
631
632 /**
633 @brief Classe responsavel pelo calculo do Fator volume de formacao do
        oleo pela correlacao Vasquez e Beggs
634 @class CRs_VazquezBeggs
635 @file CRs_VasquezBeggs.h
636  */
637 class CBo_VasquezBeggs: public CBoFatorVolFormacaoOleo, C3Parametros ///
        Herdeira das classes CFatorVolumeFormacaoOleo e C3Parametros
638 {
639 ///Metodos
640 public:
641     ///Construtor default
642     CBo_VasquezBeggs () {} ;
643     ///Destrutor
644     ~CBo_VasquezBeggs() {} ;
645
646     /// Metodo para calculo do Bo pela correlacao Vasquez e Beggs
647     std::vector<double> Bo(CFluidoBlackOil& fluido, CPressaoBolha& cpb,
            CRazaoSolubilidade& crs, CCompresOleo& cco);
648
649 };
650 #endif // CBo_VasquezBeggs_h
```

Apresenta-se na listagem 1.22 o arquivo de implementação da classe CBo_VasquezBeggs.

Listing 1.22: Arquivo de implementação da classe $\mathrm{CBo}_{V}asquezBeggs$.

```
651 #include <cmath>
652
653 #include "CBo_VasquezBeggs.h"
654
655 /**
```

```
656 @class CBo_VasquezBeggs
657 @file CBo_VasquezBeggs.cpp
658  */
659 using namespace std;
660
661 /// Metodo para calculo do Bo pela correlacao Vasquez e Beggs
662 vector<double> CBo_VasquezBeggs::Bo(CFluidoBlackOil& fluido,
      CPressaoBolha& cpb, CRazaoSolubilidade& crs, CCompresOleo& cco){
663     /// Seta os parametros da correlacao C3Parametros
664     if (fluido.api <= 30.0){
665         c1 = 0.0004677;
666         c2 = 0.00001751;
667         c3 = -0.00000001811;
668     }
669     else {
670         c1 = 0.0004677;
671         c2 = 0.000011;
672         c3 = -0.000000001377;
673     }
674
675         /// Calculo do Bo no ponto de bolha
676         bob = 1+c1*fluido.rsb+c2*(fluido.temp-60.0)*(fluido.api/fluido.
               dgs)+c3*fluido.rsb*(fluido.temp-60.0)*(fluido.api/fluido.dgs)
               ;
677
678     vbo.resize(fluido.vpressao.size()); //mesmo tamanho
679
680     /// Calculo do Bo acima do ponto de bolha e diferente do calculo
           para abaixo do ponto de bolha
681     for (int i=0; i<fluido.vpressao.size(); i++){
682
683         ///Verifica se a pressao esta acima ou abaixo do ponto de bolha
684         if (fluido.vpressao[i] <= cpb.pb)
685             vbo[i] = 1.0+c1*crs.vrs[i]+c2*(fluido.temp-60.0)*(fluido.api
                   /fluido.dgs)+c3*crs.vrs[i]*(fluido.temp-60.0)*(fluido.api
                   /fluido.dgs);
686
687         else
688             vbo[i] = bob*exp(cco.vco[i]*(cpb.pb - fluido.vpressao[i]));
689     }
690
691     return vbo;
692 }
```

Apresenta-se na listagem 1.23 o arquivo com código da classe CCompresOleo.

Listing 1.23: Arquivo de cabeçalho da classe CCompresOleo.

```
693 #ifndef CCompresOleo_h
694 #define CCompresOleo_h
```

```
695
696 #include <iostream>
697 #include <vector>
698
699 #include "CFluidoBlackOil.h"
700 #include "CRazaoSolubilidade.h"
701
702 /**
703 @brief Classe responsavel pelo calculo da Compressibilidade do Oleo
704 @class CCompresOleo
705 @file CCompresoleo.h
706  */
707 class CCompresOleo {
708
709 ///Atributos
710 public:
711   /// Vetor de compressibilidade do oleo
712   std::vector<double> vco;
713
714 ///Metodos
715 public:
716     ///Construtor default
717    CCompresOleo    (){};
718    ///Destrutor
719     ~CCompresOleo (){};
720
721     /// Metodos get e set
722     void Vco(std::vector<double> _vco) {vco = _vco;}
723     std::vector<double> Vco()          {return vco;}
724
725     /// Metodo para calculo da compressibilidade do oleo
726     std::vector<double> Co(CFluidoBlackOil&, CRazaoSolubilidade&) {
          return vco; }
727
728     /// Metodo para gerar grafico
729     void Plot(CFluidoBlackOil& fluido);
730
731 };
732 #endif // CCompres_Oleo_h
```

Apresenta-se na listagem 1.24 o arquivo de implementação da classe CCompresOleo.

Listing 1.24: Arquivo de implementação da classe CCompresOleo.

```
733 #include "CCompresOleo.h"
734 #include "CGnuplot.h"
735
736 /**
737 @class CCompresOleo
738 @file CCompresOleo.cpp
```

```
739 */
740
741 /// Metodo para gerar grafico
742 void CCompresOleo:: Plot(CFluidoBlackOil& fluido) {
743
744 static CGnuplot g2d;
745 g2d.Title("Pressao␣x␣Compressibilidade␣do␣Oleo");
746 g2d.XLabel("Pressao(psia)");
747 g2d.YLabel("Co(1/psi)");
748 g2d.Style("linespoints");
749 g2d.PlotVector(fluido.vpressao,vco);
750 }
```

Apresenta-se na listagem 1.25 o arquivo com código da classe CCo_VasquezBeggs.

Listing 1.25: Arquivo de cabeçalho da classe $CCo_VasquezBeggs$.

```
751 #ifndef CCo_VasquezBeggs_h
752 #define CCo_VasquezBeggs_h
753
754 #include <iostream>
755 #include <vector>
756
757 #include "CCompresOleo.h"
758
759 /**
760 @brief Classe responsavel pelo calculo da compressibilidade do oleo pela
       correlacao Vasquez e Beggs
761 @class CCo_VazquezBeggs
762 @file CCo_VasquezBeggs.h
763 */
764 class CCo_VasquezBeggs: public CCompresOleo {       ///Herdeira da
      classe CCompresOleo
765
766 ///Metodos
767 public:
768     ///Construtor default
769     CCo_VasquezBeggs  () {};
770     ///Destrutor
771     ~CCo_VasquezBeggs () {};
772
773  /// Metodo para calculo do Co pela correlacao Vasquez e Beggs
774  std::vector<double> Co (CFluidoBlackOil& fluido,CRazaoSolubilidade& crs
       );
775
776 };
777 #endif // CCo_VasquezBeggs_h
```

Apresenta-se na listagem 1.26 o arquivo de implementação da classe CCo_VasquezBeggs.

Listing 1.26: Arquivo de implementação da classe $CCo_VasquezBeggs$.

```
778 #include "CCo_VasquezBeggs.h"
779
780 /**
781 @class CBo_VasquezBeggs
782 @file CBo_VasquezBeggs.cpp
783  */
784 using namespace std;
785
786 /// Metodo para calculo do Co pela correlacao Vasquez e Beggs
787 vector<double> CCo_VasquezBeggs:: Co (CFluidoBlackOil& fluido,
      CRazaoSolubilidade& crs) {
788     vco.resize(fluido.vpressao.size()); //mesmo tamanho
789
790 for (int i=0; i<fluido.vpressao.size(); i++){
791         vco[i]= ( -1433.0 + 5.0*crs.vrs[i] + 17.2*fluido.temp - 1180.0*
              fluido.dgs + 12.61*fluido.api) / (fluido.vpressao[i] *
              100000.0);
792         }
793     return vco;
794 }
```

Apresenta-se na listagem 1.27 o arquivo com código da classe CFatorZComp.

Listing 1.27: Arquivo de cabeçalho da classe CFatorZComp.

```
795 #ifndef CFatorZComp_h
796 #define CFatorZComp_h
797
798
799 #include <iostream>
800 #include <vector>
801
802 #include "CFluidoBlackOil.h"
803 #include "CSutton.h"
804
805 /**
806 @brief Classe responsavel pelo calculo do Fator Z de compressibilidade
      do gas
807 @class CFatorZComp
808 @file CFatorZComp.h
809  */
810 class CFatorZComp {
811 ///Atributos
812 public:
813     ///Vetor de Fator Z
814     std::vector<double> vz;
815
816 ///Metodos
```

```
817 public:
818     ///Construtor default
819     CFatorZComp ()  {};
820     ///Destrutor
821     ~CFatorZComp () {};
822 
823     /// Metodos get e set
824     void Vz(std::vector<double> _vz) {vz = _vz;}
825     std::vector<double> Vz()         {return vz;}
826 
827     ///Metodo para o calculo do Fator de compressibilidade do gas Z
828     std::vector<double> Vz(CSutton&) {return vz;}
829 
830     /// Metodo para gerar grafico
831     void Plot(CFluidoBlackOil&);
832 
833 };
834 
835 #endif // CFatorZComp_h
```

Apresenta-se na listagem 1.28 o arquivo de implementação da classe CFatorZComp.

Listing 1.28: Arquivo de implementação da classe CFatorZComp.

```
836 #include "CFatorZComp.h"
837 #include "CGnuplot.h"
838 
839 /**
840 @class CFatorZComp
841 @file CFatorZComp.cpp
842  */
843 
844 /// Metodo para gerar grafico
845 void CFatorZComp:: Plot(CFluidoBlackOil& fluido){
846 
847 static CGnuplot g2d;
848 g2d.Title("Pressao␣x␣Fator␣de␣Compressibilidade␣do␣Gas");
849 g2d.XLabel("Pressao(psia)");
850 g2d.YLabel("Z");
851 g2d.Style("linespoints");
852 g2d.PlotVector(fluido.vpressao,vz);
853 }
```

Apresenta-se na listagem 1.29 o arquivo com código da classe CFatorZ_Papay.

Listing 1.29: Arquivo de cabeçalho da classe CFatorZ$_{P}apay$.

```
854 #ifndef CFatorZ_Papay_h
855 #define CFatorZ_Papay_h
856 
857 #include "CFatorZComp.h"
```

```
858
859 /**
860 @brief Classe responsavel pelo calculo do Fator Z de compressibilidade
      do gas pela correlacao Papay
861 @class CFatorZ_Papay
862 @file CFatorZ_Papay.h
863  */
864 class CFatorZ_Papay: public CFatorZComp     ///Herdeira da classe
      CFatorZComp
865 {
866 ///Metodos
867 public:
868     ///Construtor default
869     CFatorZ_Papay  () {};
870     ///Destrutor
871     ~CFatorZ_Papay () {};
872
873     ///Metodo para calculo do fator Z pela correlacao Papay
874     std::vector<double> Z (CSutton& prtr);
875
876 };
877 #endif // CFatorZ_Papay_h
```

Apresenta-se na listagem 1.30 o arquivo de implementação da classe CFatorZ_Papay.

Listing 1.30: Arquivo de implementação da classe CFatorZ$_{P}apay$.

```
878 #include <cmath>
879
880 #include "CFatorZ_Papay.h"
881
882 /**
883 @class CFatorZ_Papay
884 @file CFatorZ_Papay.cpp
885  */
886 using namespace std;
887
888 ///Metodo para calculo do fator Z pela correlacao Papay
889 std::vector<double> CFatorZ_Papay :: Z (CSutton& prtr){
890
891 vz.resize(prtr.vpr.size()); //mesmo tamanho
892
893 for (int i=0; i < prtr.vpr.size(); i++){
894         vz[i] =  1 - (3.52 * prtr.vpr[i])/( pow(10.0, (0.9813* prtr.tr))
               ) + (0.274 * prtr.vpr[i] * prtr.vpr[i]) / (pow(10.0,(0.8157*
               prtr.tr)));
895     }
896
897     return vz;
898 }
```

Apresenta-se na listagem 1.31 o arquivo com código da classe CFatorZ_BeggsBrill.

Listing 1.31: Arquivo de cabeçalho da classe CFatorZ$_{B}eggsBrill$.

```
899 #ifndef CFatorZ_BeggsBrill_h
900 #define CFatorZ_BeggsBrill_h
901
902 #include <iostream>
903 #include <vector>
904
905 #include "CFatorZComp.h"
906
907 /**
908 @brief Classe responsavel pelo calculo do Fator Z de compressibilidade
      do gas pela correlacao Beggs e Brill
909 @class CFatorZ_BeggsBrill
910 @file CFatorZ_BeggsBrill.h
911  */
912 class CFatorZ_BeggsBrill : public CFatorZComp       ///Herdeira da
      classe CFatorZComp
913 {
914
915 ///Atributos
916 private:
917     ///Parametros da formula de Beggs e Brill(nao variam com a pressao)
918     double a, c, d;
919     /// Vetor de Paramentro de Beggs e Brill (varia com a pressao)
920     std::vector<double> b;
921
922 ///Metodos
923 public:
924     ///Construtor default
925     CFatorZ_BeggsBrill  () {};
926     ///Destrutor
927     ~CFatorZ_BeggsBrill () {};
928
929     ///Metodo para calculo do fator Z pela correlacao Beggs e Brill
930     std::vector<double> Z (CSutton& prtr);
931 };
932
933 #endif // CFatorZ_BeggsBrill_h
```

Apresenta-se na listagem 1.32 o arquivo de implementação da classe CFatorZ_BeggsBrill.

Listing 1.32: Arquivo de implementação da classe CFatorZ$_{B}eggsBrill$.

```
934 #include <iostream>
935 #include <cmath>
936
937 #include "CFatorZ_BeggsBrill.h"
938
```

```
939 /**
940 @class CFatorZ_BeggsBrill
941 @file CFatorZ_BeggsBrill.cpp
942  */
943 using namespace std;
944
945 ///Metodo para calculo do fator Z pela correlacao Beggs e Brill
946 vector<double> CFatorZ_BeggsBrill :: Z (CSutton& prtr) {
947         /// Calculo dos parametros independentes da pressao
948         a = 1.39 * (pow((prtr.tr - 0.92), 0.5)) - 0.36 * prtr.tr - 0.1;
949         c = 0.132-0.32*log10(prtr.tr);
950         d = pow(10, (0.3106 - (0.49 * prtr.tr) + (0.1824 * prtr.tr *
              prtr.tr)));
951
952             b.resize(prtr.vpr.size());  //mesmo tamanho
953             vz.resize(prtr.vpr.size()); //mesmo tamanho
954         /// Calculo do paramentro dependente da pressao e do Fator Z
955 for (int i=0; i<prtr.vpr.size(); i++){
956         b[i] = ( 0.62 - (0.23 * prtr.tr)) * prtr.vpr[i] + ((0.066/(prtr.
              tr - 0.86)) - 0.037) * prtr.vpr[i] * prtr.vpr[i] + (0.32 / (
              pow(10, (9 *(prtr.tr-1))))) * (pow(prtr.vpr[i],6));
957
958         vz[i] =  a + ((1-a)/ exp(b[i])) + (c * pow(prtr.vpr[i],d));
959
960     }
961     return vz;
962 }
```

Apresenta-se na listagem 1.33 o arquivo com código da classe CSutton.

Listing 1.33: Arquivo de cabeçalho da classe CSutton.

```
963 #ifndef CSutton_h
964 #define CSutton_h
965
966 #include <iostream>
967 #include <vector>
968
969 #include "CFluidoBlackOil.h"
970
971 /**
972 @brief Classe responsavel pelo calculo da Pressao e Temperatura pseudo-
      reduzida do gas pela correlacao Sutton
973 @class CSutton
974 @file CSutton.h
975  */
976 class CSutton {
977 ///Atributos
978 public:
979     /// Vetor de Pressao pseudo-reduzida
```

```
980     std::vector<double> vpr;
981     /// Temperatura pseudo-reduzida
982     double tr;
983
984 ///Metodos
985 public:
986     ///Construtor default
987     CSutton  () {};
988     ///Destrutor
989     ~CSutton () {};
990
991     /// Metodos get e set
992     void Vpr(std::vector<double> _vpr) {vpr = _vpr;}
993     std::vector<double> Vpr()          {return vpr;}
994
995     void Tr(double _tr)     {tr = _tr;}
996     double Tr()             {return tr;}
997
998     ///Metodo para o calculo da pressao pseudo-reduzida pela correlacao
            Sutton
999     std::vector<double> Pr (CFluidoBlackOil& fluido);
1000
1001    ///Metodo para o calculo da temperatura pseudo-reduzida pela
            correlacao Sutton
1002    double Tr (CFluidoBlackOil& fluido);
1003
1004 };
1005 #endif // CSutton_h
```

Apresenta-se na listagem **??** o arquivo de implementação da classe CSutton.

Listing 1.34: Arquivo de implementação da classe CSutton.

```
1006 #include "CSutton.h"
1007
1008 /**
1009 @class CSutton
1010 @file CSutton.cpp
1011  */
1012 using namespace std;
1013
1014 ///Metodo para o calculo da pressao pseudo-reduzida pela correlacao
       Sutton
1015 vector<double> CSutton :: Pr (CFluidoBlackOil& fluido) {
1016 vpr.resize(fluido.vpressao.size()); //mesmo tamanho
1017
1018 for (int i=0; i<fluido.vpressao.size(); i++){
1019         vpr[i] = fluido.vpressao[i]/(671.1 + (14.0 - 34.3*fluido.dg)*
                fluido.dg);
1020         }
```

```
1021     return vpr;
1022 }
1023
1024 ///Metodo para o calculo da temperatura pseudo-reduzida pela correlacao
       Sutton
1025 double CSutton :: Tr(CFluidoBlackOil& fluido) {
1026
1027     tr = (fluido.temp + 459.67) / (120.1 + (429.0 - 62.9*fluido.dg)*
           fluido.dg);
1028     return tr;
1029 }
```

Apresenta-se na listagem 1.35 o arquivo com código da classe CBgFatorVolFormacaoGas.

Listing 1.35: Arquivo de cabeçalho da classe CBgFatorVolFormacaoGas.

```
1030 #ifndef CBgFatorVolFormacaoGas_h
1031 #define CBgFatorVolFormacaoGas_h
1032
1033 #include <vector>
1034
1035 #include "CFluidoBlackOil.h"
1036 #include "CFatorZComp.h"
1037
1038 /**
1039 @brief Classe responsavel pelo calculo do Fator volume de formacao do
       gas
1040 @class CBgFatorVolFormacaoGas
1041 @file CBgFatorVolFormacaoGas.h
1042  */
1043
1044 class CBgFatorVolFormacaoGas {
1045 ///Atributos
1046 public:
1047     /// Vetor de Fator volume de formacao do gas Bg
1048     std::vector<double> vbg;
1049
1050 ///Metodos
1051 public:
1052     ///Construtor default
1053     CBgFatorVolFormacaoGas  () {};
1054     ///Destrutor
1055     ~CBgFatorVolFormacaoGas () {};
1056
1057     /// Metodos get e set
1058     void Vbg(std::vector<double> _vbg) {vbg = _vbg;}
1059     std::vector<double> Vbg()          {return vbg;}
1060
```

```
1061     /// Metodo para calcular o fator volume de formacao do gas
1062     std::vector<double> Bg (CFluidoBlackOil& fluido, CFatorZComp& fcz);
1063 
1064     /// Metodo para gerar grafico
1065     void Plot(CFluidoBlackOil& fluido);
1066 };
1067 
1068 #endif // CBgFatorVolFormacaoGas_h
```

Apresenta-se na listagem 1.36 o arquivo de implementação da classe CBgFatorVolFormacaoGas.

Listing 1.36: Arquivo de implementação da classe CBgFatorVolFormacaoGas.

```
1069 #include "CBgFatorVolFormacaoGas.h"
1070 #include "CGnuplot.h"
1071 
1072 /**
1073 @class CBgFatorVolFormacaoGas
1074 @file CBgFatorVolFormacaoGas.cpp
1075  */
1076 using namespace std;
1077 
1078 /// Metodo para calcular o fator volume de formacao do gas
1079 vector<double> CBgFatorVolFormacaoGas :: Bg(CFluidoBlackOil& fluido,
      CFatorZComp& fcz){
1080     vbg.resize(fluido.vpressao.size()); //mesmo tamanho
1081 
1082 for (int i=0; i<fluido.vpressao.size(); i++){
1083         vbg[i] = (0.00504 * fcz.vz[i] * (fluido.temp + 459.67)) / fluido
              .vpressao[i];
1084         }
1085 
1086     return vbg;
1087 }
1088 /// Metodo para gerar grafico
1089 void CBgFatorVolFormacaoGas:: Plot(CFluidoBlackOil& fluido){
1090 
1091 static CGnuplot g2d;
1092 g2d.Title("Pressao␣x␣Fator␣Volume␣de␣Formacao␣do␣Gas");
1093 g2d.XLabel("Pressao(psia)");
1094 g2d.YLabel("Bg␣(bbl/scf)");
1095 g2d.Style("linespoints");
1096 g2d.PlotVector(fluido.vpressao,vbg);
1097 
1098 }
```

Apresenta-se na listagem 1.37 o arquivo com código da classe CSimuladorBlackOil.

Listing 1.37: Arquivo de cabeçalho da classe CSimuladorBlackOil.

```
1099 #include <iostream>
1100 #include <string>
1101 #include <fstream>
1102 #include <iomanip>
1103
1104 #include "CFluidoBlackOil.h"
1105 #include "CPressaoBolha.h"
1106 #include "CPb_Standing.h"
1107 #include "CPb_VasquezBeggs.h"
1108 #include "CRazaoSolubilidade.h"
1109 #include "CRs_Standing.h"
1110 #include "CRs_VasquezBeggs.h"
1111 #include "CBoFatorVolFormacaoOleo.h"
1112 #include "CBo_Standing.h"
1113 #include "CBo_VasquezBeggs.h"
1114 #include "CCompresOleo.h"
1115 #include "CCo_VasquezBeggs.h"
1116 #include "CFatorZComp.h"
1117 #include "CSutton.h"
1118 #include "CFatorZ_Papay.h"
1119 #include "CFatorZ_BeggsBrill.h"
1120 #include "CBgFatorVolFormacaoGas.h"
1121
1122 using namespace std;
1123
1124 /**
1125 @brief Classe que simula o calculo de todos as propriedades do fluido
      blackoil
1126 @class CSimuladorBlackOil
1127 @file CSimuladorBlackOil.h
1128  */
1129 class CSimuladorBlackOil {
1130 ///Atributos
1131 public:
1132     ///Objeto da classe CFluidoBlackOil
1133     CFluidoBlackOil fblackoil;
1134     ///Objeto da classe CPb_Standing
1135     CPb_Standing standing_pb;
1136     ///Objeto da classe CPb_VasquezBeggs
1137     CPb_VasquezBeggs vb_pb;
1138     ///Objeto da classe CRs_Standing
1139     CRs_Standing standing_rs;
1140     ///Objeto da classe CRs_VasquezBeggs
1141     CRs_VasquezBeggs vb_rs;
1142     ///Objeto da classe CBo_Standing
1143     CBo_Standing standing_bo;
1144     ///Objeto da classe CBo_VasquezBeggs
```

```
1145     CBo_VasquezBeggs vb_bo;
1146     ///Objeto da classe CCo_VasquezBeggs
1147     CCo_VasquezBeggs vb_co;
1148     ///Objeto da classe CSutton
1149     CSutton sutton_prtr;
1150     ///Objeto da classe CFatorZ_Papay
1151     CFatorZ_Papay papay_z;
1152     ///Objeto da classe CFatorZ_BeggsBrill
1153     CFatorZ_BeggsBrill bb_z;
1154     ///Objeto da classe CFatorVolumeFormacaoGas
1155     CBgFatorVolFormacaoGas bg;
1156
1157 ///Metodos
1158 public:
1159     ///Construtor default
1160     CSimuladorBlackOil () {};
1161     ///Destrutor
1162     ~CSimuladorBlackOil() {};
1163
1164 ///Menu de execucao
1165 int Executar();
1166
1167 /// Metodo que solicita a entrada de dados pelo teclado ou pelo disco
1168 void EntradadeDados();
1169
1170 ///Metodo que calcula a pressao de bolha
1171 void CalculoPressaoBolha();
1172
1173 ///Metodo que calcula a razao de solubilidade
1174 void CalculoRazaodeSolubilidade();
1175
1176 ///Metodo que calcula o fator volume de formacao do oleo
1177 void CalculoFatorVolumeFormacaoOleo();
1178
1179 ///Metodo que calcula a compressibilidade do oleo
1180 void CalculoCompressibilidadeOleo();
1181
1182 ///Metodo que calcula o fator de compressibilidade do gas Z
1183 void CalculoFatordeCompressibilidadeZ();
1184
1185 ///Metodo que calcula o fator volume de formacao do gas
1186 void CalculoFatorVolumeFormacaoGas();
1187
1188 };
```

Apresenta-se na listagem 1.38 o arquivo de implementação da classe CSimuladorBlackOil.

Listing 1.38: Arquivo de implementação da classe CSimuladorBlackOil.

```
1189 #include <iostream>
1190 #include <string>
1191 #include <fstream>
1192 #include <iomanip>
1193
1194 #include "CSimuladorBlackOil.h"
1195
1196 /**
1197 @class CSimuladorBlackOil
1198 @file CSimuladorBlackOil.cpp
1199  */
1200 using namespace std;
1201
1202  string linha = "
        ------------------------------------------------------------------------------
        \n";
1203
1204  ///Menu de execucao
1205 int CSimuladorBlackOil:: Executar()
1206 {
1207     cout << "
            ------------------------------------------------------------------------------
            " << endl;
1208     cout << "|                UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE -
            DARCY RIBEIRO                |" << endl;
1209     cout << "|                          CENTRO DE CIENCIAS E TECNOLOGIA - 
            CCT                          |" << endl;
1210     cout << "|                LABORATORIO DE ENGENHARIA E EXPLORACAO DE 
            PETROLEO - LENEP              |" << endl;
1211     cout << "|                    DISCIPLINA: PROPRAMACAO PRATICA - 
            PROJETO C++                   |" << endl;
1212     cout << "|                          PROFESSOR: ANDRE DUARTE BUENO   
            |" << endl;
1213     cout << "|                          ALUNOS: JOAO BOSCO MACIEL FILHO 
            |" << endl;
1214     cout << "|                          SUSANE CHELLI CAIRES GOIS   
            |" << endl;
1215     cout << "|   PROGRAMA PARA CALCULO DE PARAMETROS DE RESERVATORIO 
            USANDO A MODELAGEM BLACK-OIL   |" << endl;
1216     cout << "
            ------------------------------------------------------------------------------
            " << endl;
1217
1218     ///Solicita a entrada de dados
1219     EntradadeDados();
1220
1221     ///Gera o vetor de pressoes, seja de um ponto especifico ou de um
```

```
        intervalo
1222    fblackoil.Gera_vp(fblackoil.p, fblackoil.pmax, fblackoil.deltap);
1223
1224    int resp;
1225    do {
1226        do{
1227    cout << linha;
1228    cout << linha;
1229    cout << "Qual parametro calcular:" << endl;
1230    cout << " 1 - Pressao de bolha - Pb " << endl;
1231    cout << " 2 - Razao de Solubilidade - Rs" << endl;
1232    cout << " 3 - Fator Volume de Formacao do Oleo - Bo" << endl;
1233    cout << " 4 - Compressibilidade do oleo - co" << endl;
1234    cout << " 5 - Fator de Compressibilidade do gas - Z" << endl;
1235    cout << " 6 - Fator Volume Formacao do Gas - Bg" << endl;
1236    cout << " 0 - Sair" << endl;
1237    cout << linha;
1238    cin >> resp; cin.get();
1239
1240        if((resp < 0) || (resp > 6))
1241        {cout << "Opcao Invalida!" << endl;};
1242
1243        }while ((resp < 0) && (resp > 6));
1244
1245    switch (resp) {
1246
1247case 1: CalculoPressaoBolha();                  break;
1248
1249case 2: CalculoRazaodeSolubilidade();           break;
1250
1251case 3: CalculoFatorVolumeFormacaoOleo();       break;
1252
1253case 4: CalculoCompressibilidadeOleo();         break;
1254
1255case 5: CalculoFatordeCompressibilidadeZ();     break;
1256
1257case 6: CalculoFatorVolumeFormacaoGas();        break;
1258
1259case 0 : return 0;
1260
1261    }
1262   } while (resp != 0);
1263}
1264
1265/// Metodo que solicita a entrada de dados pelo teclado ou pelo disco
1266void CSimuladorBlackOil::EntradadeDados(){
1267    string nomearquivo;
1268    char opcao;
```

```
1269     do {
1270     cout << linha;
1271     cout << "Como␣deseja␣entrar␣com␣os␣dados:␣\n";
1272     cout << "␣1␣-␣Teclado␣\n";
1273     cout << "␣2␣-␣Arquivo␣de␣disco␣\n";
1274     cout << linha;
1275     cin >> opcao; cin.get();
1276 
1277     ///Entrada de dados pelo teclado
1278     if(opcao == '1'){
1279         cout << linha;
1280         cout << "␣Entrada␣de␣dados␣pelo␣teclado" << endl;
1281         cout << linha;
1282         cin >> fblackoil;}
1283     /// Entrada de dados pelo disco
1284     else if (opcao == '2'){
1285         cout << linha;
1286         cout << "␣Entrada␣de␣dados␣por␣arquivo␣de␣disco" << endl;
1287         cout << linha;
1288         cout << "␣Informe␣o␣nome␣do␣arquivo␣com␣a␣extensao␣(ex:␣.txt,␣.
                dat)" << endl;
1289         getline(cin, nomearquivo);
1290         ifstream fin;
1291         fin.open(nomearquivo.c_str());
1292         fin >> fblackoil;}
1293     /// Mensagem de erro se a escolha for errada
1294     if ((opcao != '1') && (opcao!= '2'))
1295     {cout << "Opcao␣Invalida" << endl;}
1296     }while ((opcao != '1') && (opcao!= '2'));
1297 
1298 }
1299 
1300 ///Metodo que calcula a pressao de bolha
1301 void CSimuladorBlackOil:: CalculoPressaoBolha() {
1302     char resp1;
1303     do{
1304     cout << linha;
1305     cout << "␣1␣-␣Pressao␣de␣bolha␣-␣Pb␣" << endl;
1306     cout << linha;
1307     cout << "Qual␣correlacao␣deseja␣usar:␣" << endl;
1308     cout << "␣1␣-␣Standing␣\n";
1309     cout << "␣2␣-␣Vasquez␣e␣Beggs␣" << endl;
1310     cout << linha;
1311     cin >> resp1; cin.get();
1312 
1313         if((resp1!='1') && (resp1!='2'))
1314         {cout << "Opcao␣Invalida!" << endl;};
1315 
```

```
1316     }while((resp1!='1') && (resp1!='2'));
1317
1318     switch (resp1) {
1319     /// Calculo da pressao de bolha por Standing
1320     case '1':
1321
1322     cout << "Pb = " << standing_pb.Pb(fblackoil) << " psia" << endl;
1323
1324         break;
1325
1326     /// Calculo da pressao de bolha por Vasquez e Beggs
1327     case '2':
1328
1329     cout << "Pb = " << vb_pb.Pb(fblackoil) << " psia" << endl;
1330
1331         break;
1332     }
1333
1334 }
1335
1336 ///Metodo que calcula a razao de solubilidade
1337 void CSimuladorBlackOil:: CalculoRazaodeSolubilidade() {
1338     char resp2;
1339     do{
1340     cout << linha;
1341     cout << " 2 - Razao de Solubilidade - Rs" << endl;
1342     cout << linha;
1343     cout << "Qual correlacao se desejar usar: " << endl;
1344     cout << " 1 - Standing \n";
1345     cout << " 2 - Vasquez e Beggs " << endl;
1346     cout << linha;
1347     cin >> resp2; cin.get();
1348
1349         if((resp2!='1') && (resp2!='2'))
1350         {cout << "Opcao Invalida!" << endl;};
1351
1352     }while((resp2!='1') && (resp2!='2'));
1353
1354     switch (resp2) {
1355
1356     /// Calculo da Razao de solubilidade por Standing.
1357     /// Sera necessario o calculo de outras propriedades, estas tambem
            serao calculadas por Standing
1358     case '1':
1359         standing_pb.Pb(fblackoil);
1360         cout << "Pb = " << standing_pb.pb << " psia" << endl;
1361         standing_rs.Rs(fblackoil, standing_pb);
1362
```

```
1363        cout << setw(20) << left << "Pressao␣(psia)" << left << setw(15)
                << "Razao␣Solubilidade␣(scf/STB)" << endl;
1364        for(int i = 0; i<fblackoil.vpressao.size(); i++){
1365        cout << setw(20) << left << fblackoil.vpressao[i] << left <<
               setw(15) << standing_rs.vrs[i] << endl;}
1366        standing_rs.Plot(fblackoil);
1367
1368    break;
1369
1370    /// Calculo da Razao de solubilidade por Vasquez e Beggs.
1371    /// Sera necessario o calculo de outras propriedades, estas tambem
           serao calculadas por Vasquez e Beggs
1372    case '2':
1373
1374        cout << "Pb␣=␣" << vb_pb.Pb(fblackoil) << "␣psia" << endl;
1375        vb_rs.Rs(fblackoil, vb_pb);
1376
1377        cout << left << setw(20) << "Pressao␣(psia)" << left << setw(15)
                << "Razao␣Solubilidade␣(scf/STB)" << endl;
1378        for(int i = 0; i<fblackoil.vpressao.size(); i++){
1379            cout << setw(20) << left << fblackoil.vpressao[i] << left <<
                    setw(15) << vb_rs.vrs[i] << endl;}
1380        vb_rs.Plot(fblackoil);
1381
1382    break;
1383    }
1384}
1385
1386///Metodo que calcula o fator volume de formacao do oleo
1387void CSimuladorBlackOil::CalculoFatorVolumeFormacaoOleo() {
1388    char resp3;
1389    do{
1390    cout << linha;
1391    cout << "␣3␣-␣Fator␣Volume␣de␣Formacao␣do␣Oleo␣-␣Bo" << endl;
1392    cout << linha;
1393    cout << "Qual␣correlacao␣se␣desejar␣usar:␣" << endl;
1394    cout << "␣1␣-␣Standing␣\n";
1395    cout << "␣2␣-␣Vasquez␣e␣Beggs␣" << endl;
1396    cout << linha;
1397    cin >> resp3; cin.get();
1398
1399        if((resp3!='1') && (resp3!='2'))
1400        {cout << "Opcao␣Invalida!" << endl;};
1401
1402    }while((resp3!='1') && (resp3!='2'));
1403
1404    switch (resp3) {
1405
```

```
1406     /// Calculo do Fator volume de formacao do oleo no ponto de bolha
             por Standing,
1407     case '1':
1408
1409         standing_bo.Bob(fblackoil);
1410         cout << "Fator Volume de Formacao do Oleo no ponto de bolha" <<
                 endl;
1411         cout << "Bob = " << standing_bo.bob << " bbl/STB" << endl;
1412
1413     break;
1414
1415     /// Calculo do Fator volume de formacao do oleo por Vasquez e Beggs.
1416     /// Sera necessario o calculo de outras propriedades, estas tambem
             serao calculadas por Vasquez e Beggs
1417     case '2':
1418
1419         vb_pb.Pb(fblackoil);
1420         vb_rs.Rs(fblackoil, vb_pb);
1421         vb_co.Co(fblackoil,vb_rs);
1422         vb_bo.Bo(fblackoil, vb_pb, vb_rs, vb_co);
1423         cout << "Pressao no ponto de bolha" << endl;
1424         cout << "Pb = " << vb_pb.pb << " psia" << endl;
1425         cout << "Fator Volume de Formacao do Oleo no ponto de bolha" <<
                 endl;
1426         cout << "Bob = " << vb_bo.bob << " bbl/STB" << endl;
1427         cout << left << setw(20) << "Pressao (psia)" << left << setw(15)
                 << "Fator Volume de Formacao do Oleo (bbl/STB)" << endl;
1428         for(int i = 0; i<fblackoil.vpressao.size(); i++){
1429             cout << setw(20) << left << fblackoil.vpressao[i] << left <<
                     setw(15) << vb_bo.vbo[i] << endl;}
1430         vb_bo.Plot(fblackoil);
1431
1432     break;
1433     }
1434 }
1435
1436 ///Metodo que calcula a compressibilidade do oleo
1437 void CSimuladorBlackOil::CalculoCompressibilidadeOleo(){
1438     cout << linha;
1439     cout << " 4 - Compressibilidade do oleo - co" << endl;
1440
1441     /// Calculo da Compressibilidade do oleo por Vasquez e Beggs.
1442     /// Sera necessario o calculo de outras propriedades, estas tambem
             serao calculadas por Vasquez e Beggs
1443     vb_pb.Pb(fblackoil);
1444     vb_rs.Rs(fblackoil, vb_pb);
1445     vb_co.Co(fblackoil,vb_rs);
1446     cout << left << setw(20) << "Pressao (psia)" << left << setw(15) <<
```

```
         "Compressibilidade␣do␣Oleo␣(1/psia)" << endl;
1447     for(int i = 0; i<fblackoil.vpressao.size(); i++){
1448     cout << setw(20) << left << fblackoil.vpressao[i] << left << setw
         (15) << vb_co.vco[i] << endl;}
1449     vb_co.Plot(fblackoil);
1450
1451 }
1452
1453 ///Metodo que calcula o fator de compressibilidade do gas Z
1454 void CSimuladorBlackOil::CalculoFatordeCompressibilidadeZ() {
1455     char resp4;
1456     do{
1457     cout << linha;
1458     cout << "␣5␣-␣Fator␣de␣Compressibilidade␣do␣gas␣-␣Z" << endl;
1459     cout << linha;
1460     cout << "Qual␣correlacao␣se␣desejar␣usar:␣" << endl;
1461     cout << "␣1␣-␣Papay␣\n";
1462     cout << "␣2␣-␣Beggs␣e␣Brill␣" << endl;
1463     cout << linha;
1464     cin >> resp4; cin.get();
1465
1466         if((resp4!='1') && (resp4!='2'))
1467         {cout << "Opcao␣Invalida!" << endl;};
1468
1469     }while((resp4!='1') && (resp4!='2'));
1470
1471     /// Calculo da Pressao e Temperatura pseudo-reduzida por Sutton
1472         sutton_prtr.Pr(fblackoil);
1473         sutton_prtr.Tr(fblackoil);
1474
1475     switch (resp4) {
1476
1477     /// Calculo do Fator Z de compressibilidade do gas por Papay
1478     case '1':
1479
1480     papay_z.Z(sutton_prtr);
1481     cout << left << setw(20) << "Pressao␣(psia)" << left << setw(15) <<
         "Fator␣de␣Compressibilidade␣do␣Gas" << endl;
1482     for(int i = 0; i<fblackoil.vpressao.size(); i++){
1483     cout << setw(20) << left << fblackoil.vpressao[i] << left << setw
         (15) <<  papay_z.vz[i] << endl;}
1484     papay_z.Plot(fblackoil);
1485
1486         break;
1487
1488     /// Calculo do Fator Z de compressibilidade do gas por Beggs e Brill
1489     case '2':
1490
```

```
1491     bb_z.Z(sutton_prtr);
1492     cout << left << setw(20) << "Pressao␣(psia)" << left << setw(15) <<
            "Fator␣de␣Compressibilidade␣do␣Gas" << endl;
1493     for(int i = 0; i<fblackoil.vpressao.size(); i++){
1494     cout << setw(20) << left << fblackoil.vpressao[i] << left << setw
            (15) << bb_z.vz[i] << endl;}
1495     bb_z.Plot(fblackoil);
1496 
1497         break;
1498     }
1499 
1500 }
1501 
1502 ///Metodo que calcula o fator volume de formacao do gas
1503 void CSimuladorBlackOil::CalculoFatorVolumeFormacaoGas() {
1504 
1505     cout << linha;
1506     cout << "␣6␣-␣Fator␣Volume␣Formacao␣do␣Gas␣-␣Bg" << endl;
1507     cout << linha;
1508     cout << "Necessario␣o␣calculo␣de␣Z,␣qual␣correlacao␣deseja␣usar:␣"
            << endl;
1509     char resp4;
1510     do{
1511     cout << linha;
1512     cout << "␣Fator␣de␣Compressibilidade␣do␣gas␣-␣Z" << endl;
1513     cout << "␣1␣-␣Papay␣\n";
1514     cout << "␣2␣-␣Beggs␣e␣Brill␣" << endl;
1515     cout << linha;
1516     cin >> resp4; cin.get();
1517 
1518         if((resp4!='1') && (resp4!='2'))
1519         {cout << "Opcao␣Invalida!" << endl;};
1520 
1521     }while((resp4!='1') && (resp4!='2'));
1522 
1523     /// Calculo da Pressao e Temperatura pseudo-reduzida por Sutton
1524         sutton_prtr.Pr(fblackoil);
1525         sutton_prtr.Tr(fblackoil);
1526 
1527     switch (resp4) {
1528 
1529     /// Calculo do Fator volume de formacao do gas usando o Fator Z
            calculado por Papay
1530     case '1':
1531 
1532     papay_z.Z(sutton_prtr);
1533     bg.Bg(fblackoil, papay_z);
1534     cout << left << setw(20) << "Pressao␣(psia)" << left << setw(15) <<
```

```
            "Fator␣Volume␣de␣Formacao␣do␣Gas␣(bbl/scf)" << endl;
1535        for(int i = 0; i<fblackoil.vpressao.size(); i++){
1536        cout << setw(20) << left << fblackoil.vpressao[i] << left << setw
               (15) << bg.vbg[i] << endl;}
1537        bg.Plot(fblackoil);
1538
1539            break;
1540
1541        /// Calculo do Fator volume de formacao do gas usando o Fator Z
               calculado por Beggs e Brill
1542        case '2':
1543
1544        bb_z.Z(sutton_prtr);
1545        bg.Bg(fblackoil, bb_z);
1546        cout << left << setw(20) << "Pressao␣(psia)" << left << setw(15) <<
               "Fator␣Volume␣de␣Formacao␣do␣Gas␣(bbl/scf)" << endl;
1547        for(int i = 0; i<fblackoil.vpressao.size(); i++){
1548        cout << setw(20) << left << fblackoil.vpressao[i] << left << setw
               (15) << bg.vbg[i] << endl;}
1549        bg.Plot(fblackoil);
1550
1551            break;
1552        }
1553}
```

Apresenta-se na listagem 1.39 o programa que usa a classe main.

Listing 1.39: Arquivo de implementação da função `main()`.

```
1558#include "CSimuladorBlackOil.h"
1559
1560using namespace std;
1561
1562int main()
1563{
1564    ///Objeto da classe CSimuladorBlackOil
1565    CSimuladorBlackOil simulador;
1566    ///Executa o simulador
1567    simulador.Executar();
1568  return 0;
1569}
```