## Trabalho 1 Métodos Iterativos para Sistemas Lineares

Eduardo Brunaldi dos Santos — Jorge Ashkar Ferreira Simondi 8642515 — 8517081

Victor Luiz da Silva Mariano Pereira 8602444

2018

1

Introdução

1 Método Iterativo de Gauss-Seidel	

## 2 Códigos Fonte

2.1 Função principal (main.c)

```
1
          Trabalho 1 - Métodos Iterativos para Sitemas Lineares
2
3
              Cálculo Numérico
                                 SME-0104
4
              Prof.: Murilo Francisco Tomé
5
6
7
              Eduardo Brunaldi dos Santos
                                                       8642515
8
              Jorge Ashkar Ferreira Simondi
                                                       8517081
9
              Victor Luiz da Silva Mariano Pereira
                                                       8602444
10
11
12
    #include <stdio.h>
    #include <stdlib.h>
13
    #include <gauss_seidel.h>
14
15
    int main (int argc, char *argv[]){
16
        // Variáveis de entrada
17
        int n;
                           // Quantidade de equações do sistema linear
18
        int itmax;
                           // Número máximo de iterações
19
20
        long double **A; // Matriz de sistemas lineares
21
        long double *b;
                           // Vetor resultado das equações
22
        long double *x;
                           // Vetor inicial
23
        long double e;
                           // Erro permitido, precisão
24
        // Iteradores
25
        int i;
26
        int j;
27
28
        // Dimensão da matriz (número de equações do sistema linear)
29
        scanf("%d", &n);
30
31
32
        // Alocação da matriz
        A = malloc(sizeof(long double *) * n);
33
        for (i = 0; i < n; i++)
34
            A[i] = malloc(sizeof(long double) * n);
35
36
        // Pegando valores de A
37
        for (i = 0; i < n; i++)
38
39
            for (j = 0; j < n; j++)
                scanf("%Lf", &(A[i][j]));
40
41
        // Alocação do vetor resultado
42
43
        b = malloc(sizeof(long double) * n);
44
        // Pegando os valores de b
45
        for(i = 0; i < n; i++)
46
            scanf("%Lf", &b[i]);
47
48
49
        // Alocação do vetor chute
        x = malloc(sizeof(long double) * n);
50
51
        // Pegando os valores do x(0), o vetor inicial
52
        for(i = 0; i < n; i++)
53
            scanf("%Lf", &x[i]);
54
55
         // Pegando o valor da precisão (erro permitido)
```

```
57
        scanf("%Lf", &e);
58
        // Pegando a quantidade máxima de iterações
59
        scanf("%d", &itmax);
60
61
        // Calcula um valor aproximado para a resposta
62
        // usando o método de Gauss-Seidel
63
        x = gauss_seidel(A, b, x, n, e, itmax);
64
65
        // Imprime a solução na tela
66
        imprime_vetor(x, n);
67
68
69
        // Liberando a memória
70
        free(x);
71
        free(b);
        for (i = 0; i < n; i++)
72
73
            free(A[i]);
        free(A);
74
75
        return 0;
76
77
    2.2
         Biblioteca auxiliar
    2.2.1 Header (gauss_seidel.h)
1
          Trabalho 1 - Métodos Iterativos para Sitemas Lineares
2
3
                                 SME-0104
              Cálculo Numérico
4
              Prof.: Murilo Francisco Tomé
5
6
7
              Eduardo Brunaldi dos Santos
                                                       8642515
              Jorge Ashkar Ferreira Simondi
                                                       8517081
8
              Victor Luiz da Silva Mariano Pereira 8602444
9
10
11
    #ifndef GAUSS_SEIDEL_H
12
    #define GAUSS_SEIDEL_H
13
14
15
     * Função para imprimir de forma mais legível uma matriz quadrada
16
17
     * Oparam A Matriz a ser impressa
     * @param n dimensão da matriz
18
19
    void imprime_matriz(long double **A, int n);
20
21
22
     * Função para imprimir um vetor de forma mais legível
23
24
     * Oparam v vetor a ser impresso
     * @param n tamanho do vetor
25
26
    void imprime_vetor(long double *v, int n);
27
28
29
     * Função para retornar a norma infinita de um vetor obtido pela subtração
30
     * de dois vetores
31
     * @param xk Vetor x(k+1)
```

```
33
     * @param x Vetor x(k)
34
     * @param n Dimensão dos vetores
35
                  norma do vetor obtido pela subtração
36
37
    long double norma_infinita(long double *xk, long double *x, int n);
38
39
     * Função para resolver o sistema linear usando o método de gauss-seidel
40
     * Oparam A Matriz de funções do sistema linear
41
     * @param b
                    Resultados das equações do sistema linear
42
     * @param x
                    Vetor contendo os resultados iniciais
43
     * @param n
                    Dimensão do sistema linear
44
45
     * @param e
                    Precisão
46
     * @param itmax Número máximo de iterações
47
    long double *gauss_seidel(long double **A, long double *b, long double *x, int n, long
     → double e, int itmax);
49
     #endif
50
    2.2.2 Implementação da biblioteca (gauss_seidel.c)
1
2
          Trabalho 1 - Métodos Iterativos para Sitemas Lineares
3
4
              Cálculo Numérico
                                  SME-0104
              Prof.: Murilo Francisco Tomé
5
6
7
              Eduardo Brunaldi dos Santos
                                                       8642515
              Jorge Ashkar Ferreira Simondi
                                                       8517081
8
              Victor Luiz da Silva Mariano Pereira
9
                                                      8602444
10
11
    #include <stdio.h>
12
     #include <stdlib.h>
13
    #include <math.h>
14
    #include <gauss_seidel.h>
15
16
17
     * Função para imprimir de forma mais legível uma matriz quadrada
18
     * Oparam A Matriz a ser impressa
19
     * @param n dimensão da matriz
20
21
22
    void imprime_matriz(long double **A, int n){
23
        int i;
24
        int j;
25
        for (i = 0; i < n; i++){
26
            for (j = 0; j < n; j++)
27
                printf("%Lf\t", A[i][j]);
28
            printf("\n");
29
30
31
32
33
     * Função para imprimir um vetor de forma mais legível
34
     * Oparam v vetor a ser impresso
35
       Qparam n tamanho do vetor
36
```

```
37
38
    void imprime_vetor(long double *v, int n){
39
        int i;
40
        for (i = 0; i < n; i++)
41
42
            printf("%.16Lf\n", v[i]);
43
44
45
     * Função para retornar a norma infinita de um vetor obtido pela subtração
46
     * de dois vetores
47
     * @param xk Vetor x(k+1)
48
49
     * @param x Vetor x(k)
50
     * @param n Dimensão dos vetores
51
     * @return
                  norma do vetor obtido pela subtração
52
53
    long double norma_infinita(long double *xk, long double *x, int n){
54
        int i;
        long double maximo = 0;
55
56
        for(i = 0; i < n; i++)
57
            if(fabs(xk[i] - x[i]) > maximo)
58
                maximo = fabs(xk[i] - x[i]);
59
60
        return maximo;
61
62
63
64
65
     * Função para resolver o sistema linear usando o método de gauss-seidel
                    Matriz de funções do sistema linear
66
     * @param A
     * @param b
                    Resultados das equações do sistema linear
67
                     Vetor contendo os resultados iniciais
     * @param x
68
                    Dimensão do sistema linear
69
     * @param n
                     Precisão
70
     * @param e
     * @param itmax Número máximo de iterações
71
72
    long double *gauss_seidel(long double **A, long double *b, long double *x, int n, long
73
       double e, int itmax){
        // Variáveis auxiliares
74
        long double somaL;
75
        long double somaU;
76
        long double *x_ant;
77
78
        // Iteradores
79
        int i;
80
        int j;
81
        int it = 0;
82
83
        // Alocando espaço da memória para o vetor auxiliar
84
85
        x_ant = malloc(sizeof(long double) * n);
86
        dof
87
            // Para toda iteração, atualiza o vetor x anterior
88
            for(i = 0; i < n; i++)
89
                x_ant[i] = x[i];
90
91
            for(i = 0; i < n; i++){
                somaL = 0;
93
94
                somaU = 0;
95
                 // Somatório da parte de baixo da matriz
```

```
96
                  for(j = 0; j < i; j++)
                      somaL += A[i][j] * x[j];
97
                  // Somatório da parte de cima da matriz
98
                  for(j = i + 1; j < n; j++)
99
100
                      somaU += A[i][j] * x[j];
101
                  // Aproximação do x
                  x[i] = (b[i] - somaL - somaU)/A[i][i];
102
              }
103
104
              // Calcula a norma infinita e compara com a tolerância
105
106
              if(norma_infinita(x, x_ant, n) <= e){</pre>
107
                  free(x_ant);
108
                  return x;
109
              }
110
111
              it++;
         }while(it < itmax);</pre>
112
113
          // Libera memória
114
         free(x_ant);
115
         return x;
116
117
```

## 2.3 Programa auxiliar (gera\_input.c)

```
1
2
          Trabalho 1 - Métodos Iterativos para Sitemas Lineares
3
              Cálculo Numérico
                                   SME-0104
4
              Prof.: Murilo Francisco Tomé
5
6
7
              Eduardo Brunaldi dos Santos
                                                       8642515
8
              Jorge Ashkar Ferreira Simondi
                                                       8517081
              Victor Luiz da Silva Mariano Pereira
9
                                                       8602444
10
11
    #include <stdio.h>
12
    #include <stdlib.h>
13
14
15
     * Programa feito para gerar parte do input utilizado no trabalho
16
     * com ele é possível gerar a matriz A solicitada, o vetor b (que
17
     * dependendo da letra do enunciado é diferente), o vetor x com o
18
19
     * chute inicial O, número máximo de iterações e a tolerância de erro.
20
     * O programa sempre imprimirá mensagem de como usar se caso não for
     * usado corretamente.
21
     * Se usado corretamente, ele sempre imprimirá na sequência:
22
23
                  A (um elemento por linha)
24
25
                  b (um elemento por linha)
26
                  x (um elemento por linha)
27
                  e (precisão)
                   itmax (número máximo de iterações)
28
29
       para facilitar, recomendo que use da seguinte forma:
30
31
32
                   ./gera_input n exercicio
33
```

```
34
     st sendo que n é a dimensão da matriz, dos vetores x e do vetor b. Já a
35
     * variável exercicio pode assumiros seguintes valores:
                                0 -> para gerar exemplo para o item b) do trabalho
36
37
                              e 1 -> para gerar exemplo para o item c) do trabalho
38
39
     */
40
41
     * Função que gera uma matriz pentadiagonal como a descrita no enunciado
42
     * do trabalho
43
     * @param n Dimensão da matriz
44
     * Oreturn Matriz alocada e com valores setados
45
46
47
    int **gera_matriz_A(int n){
48
        int i;
49
        int j;
50
        int **A;
51
52
        // Alocando memória
53
        A = malloc(sizeof(int *) * n);
54
        for(i = 0; i < n; i++)
55
            A[i] = malloc(sizeof(int) * n);
56
57
        for(i = 0; i < n; i++)
58
            for(j = 0; j < n; j++){
59
60
                 // Caso da diagonal principal
                if(i == j)
61
62
                     A[i][j] = 4;
                 // Caso das diagonais especiais
63
                 else if(j == i+1 || i == j+1 || j == i+3 || i == j+3)
64
                     A[i][j] = -1;
65
                 // Caso do resto
66
67
                 else
                     A[i][j] = 0;
68
            }
69
70
71
        return A;
72
73
74
75
     * Função que gera o vetor b de acordo com a letra b) do enunciado do
     * trabalho
76
     * Oparam A Matriz base para a criação do vetor b
77
     * @param n Tamanho do vetor
78
     * @return Vetor b com os valores solicitados
79
80
81
    int *gera_vetor_b_b(int **A, int n){
        int i;
82
83
        int j;
        int *b;
84
85
        b = calloc(sizeof(int), n);
86
87
        for(i = 0; i < n; i++)
88
            for(j = 0; j < n; j++)
89
90
                b[i] += A[i][j];
91
92
        return b;
93
```

```
94
 95
      * Função que gera o vetor b de acordo com a letra c) do enunciado do
 96
      * trabalho
 97
 98
      * @param n Tamanho do vetor
      * Oreturn Vetor b com os valores solicitados
 99
100
     long double *gera_vetor_b_c(int n){
101
         int i;
102
         long double *b;
103
104
         b = malloc(sizeof(int)* n);
105
106
         for(i = 0; i < n; i++)
107
108
             b[i] = 1.0/((i+1)*1.0);
109
110
         return b;
111
112
113
      * Função para imprimir um vetor de inteiros, com um elemento por linha
114
      * Oparam v Vetor a ser impresso
115
      * @param n Dimensão do vetor
116
117
     void imprime_vetor(int *v, int n){
118
119
         int i;
120
         for(i = 0; i < n; i++)
121
             printf("%d\n", v[i]);
122
123
124
125
126
      * Função para imprimir uma matriz para usar como input de outro programa
      * Oparam A Matriz a ser impressa
127
128
      * @param n dimensão da matriz
129
130
     void imprime_matriz(int **A, int n){
131
         int i;
         int j;
132
133
         for (i = 0; i < n; i++)
134
             imprime_vetor(A[i], n);
135
136
137
138
      * Função para imprimir um vetor para usar como input de outro programa
139
140
      * no caso o vetor é da forma do item c) do trabalho
141
      * Oparam b vetor a ser impresso
      * @param n dimensão do vetor
142
143
     void imprime_vetor_c(long double *b, int n){
144
         int i;
145
146
         for(i = 0; i < n; i++)
147
             printf("%.16Lf\n", b[i]);
148
149
150
     int main(int argc, char *argv[]){
151
         // Variáveis
152
         int *x;
153
```

```
154
          int *bb;
          int **A;
155
          long double *bc;
156
          int ex;
157
158
          int n;
159
          // Iteradores
160
          int i;
161
162
          if(argc != 3){
163
              printf("Usage: ./programa valor_de_n exercicio, sendo que o exercicio pode ter
164
              \hookrightarrow valores\n\t0 -> b\n\t1 -> c\n");
165
              return -1;
166
167
168
          n = atol(argv[1]);
169
          ex = atol(argv[2]);
170
          if(ex > 1 \&\& ex < 0 || n < 1){
171
              printf("Usage: ./programa valor_de_n exercicio, sendo que o exercicio pode ter
172
              \hookrightarrow valores\n\t0 -> b\n\t1 -> c\n");
              return -1;
173
174
175
          A = gera_matriz_A(n);
176
177
          printf("%d\n", n);
178
          imprime_matriz(A, n);
179
180
          // Como o chute inicial é sempre o vetor nulo, podemos alocar com zeros
          x = calloc(sizeof(int), n);
181
182
          if(ex == 0){
183
              bb = gera_vetor_b_b(A, n);
184
              imprime_vetor(bb, n);
185
186
187
              imprime_vetor(x, n);
188
              printf("0.00001\n");
189
              printf("10000000\n");
190
191
              // Liberando memória
192
              free(bb);
193
          } else if(ex == 1){
194
195
              bc = gera_vetor_b_c(n);
              imprime_vetor_c(bc, n);
196
197
198
              imprime_vetor(x, n);
199
              printf("0.000000001\n");
200
              printf("10000000\n");
201
202
              // Liberando memória
203
204
              free(bc);
          } else{
205
              // Liberando memória
206
              for(i = 0; i < n; i++)
207
208
                  free(A[i]);
209
              free(A);
210
              free(x);
              return -1;
211
```

```
212
         }
213
214
         // Liberando memória
         for(i = 0; i < n; i++)
215
             free(A[i]);
216
         free(A);
217
         free(x);
218
219
220
         return 0;
221
```

Considerações finais	