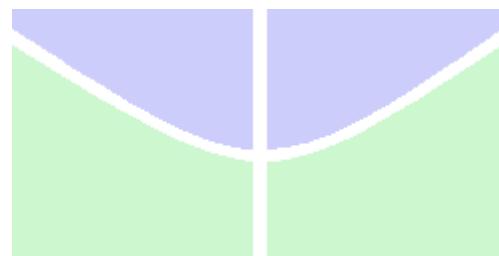


Solução de sistemas de equações lineares:

Métodos Iterativos:

Método de Gauss-Jacobi e Gauss-Seidel

Componentes	Número de matrícula
Eduardo Furtado Sá Corrêa	09/0111575
María Florencia Grenier	15/0082851
Wilson Barroso Neto	13/0018694



# 1 – Introdução

O objetivo é encontrar métodos que permitam aproximar a solução de um sistema linear, de forma de diminuir o número de operações, o que pode ser útil no caso de se tratar de um sistema com um grande número de equações, especialmente se a matriz possuir muitos elementos nulos. Outra utilidade é evitar definir ou armazenar a matriz, ou ainda, evitar os problemas de instabilidade numérica, que podem ocorrer num método direto.

Um método iterativo é basicamente obter uma equação de recorrência e propor uma solução vetor inicial; posteriormente, deve fazer as iterações necessárias até que a diferença entre dois vetores consecutivos atenda uma tolerância pré-definida.

A ideia central é generalizar o método do ponto fixo utilizado nas buscas de raízes de uma equação.

Seja o sistema linear  $\mathbf{A} \mathbf{x} = \mathbf{b}$ , onde:

**A**: matriz de coeficientes  $n \times n$ ;

**x**: vetor de variáveis,  $n \times 1$ ;

**b**: vetor dos termos constantes,  $n \times 1$ .

Este sistema convertido, de alguma forma, num sistema do tipo  $\mathbf{x} = \mathbf{C} \mathbf{x} + \mathbf{g}$  onde  $\mathbf{C}$  é a matriz  $n \times n$  e  $\mathbf{g}$  vetor  $n \times 1$ . Observamos que  $\phi(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{g}$  é uma função de iteração dada na forma matricial.

E então proposto o esquema iterativo:

Partimos de  $\mathbf{x}^{(0)}$  (vetor aproximação inicial) e então construímos consecutivamente os vetores:

$$\begin{aligned} \mathbf{x}^{(1)} &= \mathbf{C} \mathbf{x}^{(0)} + \mathbf{g} = \phi(\mathbf{x}^{(0)}) && \text{primeira aproximação} \\ \mathbf{x}^{(0)} &= \mathbf{C} \mathbf{x}^{(1)} + \mathbf{g} = \phi(\mathbf{x}^{(1)}) && \text{segunda aproximação, etc.} \end{aligned}$$

De um modo geral, a aproximação  $\mathbf{x}^{(k+1)}$  é calculada pela formula  $\mathbf{x}^{(k+1)} = \mathbf{C}\mathbf{x}(k) + \mathbf{g}$ , ou seja  $\mathbf{x}^{(k+1)} = \phi(\mathbf{x}^{(k)})$ ,  $k = 0, 1, \dots$

É importante observar que se a sequencia de aproximações  $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}, \dots$  é tal que  $\lim_{k \rightarrow \infty} \mathbf{x}(k) = \mathbf{a}$ , então  $\mathbf{a} = \mathbf{C}\mathbf{a} + \mathbf{g}$ , ou seja,  $\mathbf{a}$  é solução do sistema linear  $\mathbf{A} \mathbf{x} = \mathbf{b}$ .

## 1.1 Método de Gauss - Jacobi

Dada uma matriz quadrada  $n$  de equações lineares:

$$A\mathbf{x} = \mathbf{b}$$

em que:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Então  $A$  pode ser decomposto num componente diagonal  $D$  e o resto  $R$ :

$$A = D + R \quad \text{Onde} \quad D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \quad R = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}$$

O sistema de equações lineares pode ser reescrito como:

$$D\mathbf{x} = \mathbf{b} - R\mathbf{x}$$

O método de Gauss-Jacobi é um método iterativo que resolve o membro esquerdo da expressão em ordem a  $\mathbf{x}$  ao usar o método resultante da iteração anterior no membro direito. Analiticamente, isto pode ser escrito como:

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - R\mathbf{x}^{(k)})$$

ou, equivalentemente:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n.$$

Nota-se que a computação de  $x_i^{k+1}$  é feita com base em todos os valores obtidos em iterações anteriores.

### 1.1.1 Critério de convergência (critério das linhas)

$$\sum_{j=1, j \neq k}^n \frac{|a_{kj}|}{|a_{kk}|}$$

Seja o sistema linear  $Ax = b$  e seja:  $\alpha_k = \sum_{j=1, j \neq k}^n \frac{|a_{kj}|}{|a_{kk}|}$ .

Se  $\alpha_k < 1$ , sendo  $1 \leq k \leq n$ , então o método de Jacobi gera uma sequência  $\{x(k)\}$  convergente para a solução do sistema dado, independente da escolha da aproximação inicial,  $x(0)$ .

## 1.2 Método de Gauss – Seidel

Procuramos a solução do conjunto de equações lineares, expressadas em termos de matriz como:

A iteração Gauss-Seidel é

$$x^{(k+1)} = (D - L)^{-1} (Ux^{(k)} + b),$$

onde  $A = D + L + U$ ; as matrizes  $D$ ,  $L$ , e  $U$  representam respectivamente os coeficientes da matriz  $A$ : a diagonal, triangular estritamente inferior, e triangular estritamente superior; e  $k$  é o contador da iteração. Esta expressão matricial é utilizada principalmente para analisar o método. Quando implementada, Gauss-Seidel, uma aproximação explícita de entrada por entrada é utilizada:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j < i} a_{ij} x_j^{(k+1)} - \sum_{j > i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n.$$

Diferenciando-se do método de Gauss-Jacob:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n$$

Sendo que o método de Gauss-Seidel apresenta convergência mais rápida que este último.

Note que o cálculo  $x_i^{(k+1)}$  de utiliza apenas os elementos de  $x^{(k+1)}$

que já havia sido calculada e apenas aqueles elementos de  $\mathbf{x}^{(k)}$  já haviam avançado para a iteração  $k + 1$ . Isto significa que nenhum armazenamento adicional é necessário, e que computacionalmente pode ser substituído ( $\mathbf{x}^{(k)}$  por  $\mathbf{x}^{(k+1)}$ ). A iteração geralmente continua até que a solução esteja dentro da tolerância especificada.

### 1.2.1 Critério de Sassenfeld

Seja o sistema linear  $A\mathbf{x} = \mathbf{b}$ , com  $A$  dimensão  $n \times n$  e seja:

$$\beta_1 = \frac{|a_{12}| + |a_{13}| + |a_{14}| + \dots + |a_{1n}|}{|a_{11}|}$$

e para  $j = 2, 3, \dots, n$ :

$$\beta_j = \frac{|a_{j1}| \beta_1 + |a_{j2}| \beta_2 + \dots + |a_{jj-1}| \beta_{j-1} + |a_{jj+1}| + \dots + |a_{jn}|}{|a_{jj}|}$$

Define-se  $\beta = \max_{j=1,n} \beta_j$ .

Se  $\beta < 1$ , então o Método de Gauss-Seidel gera uma sequência convergente para a solução do sistema, qualquer que seja o vetor inicial. Além disso, quanto menor for o valor de  $\beta$  mais rápida é a convergência.

## 2. Metodologia

Implementamos os métodos computacionalmente com a linguagem C.

O código fonte PROJETO.C deve compilar-se e executar-se, por exemplo no sistemas *unix* como

```
clear; gcc projeto.c -o projeto -Wall; ./projeto
```

Ou em sistema *Windows* como

```
cls & gcc projeto.c -o projeto -Wall & projeto.exe
```

O programa foi implementado de forma a poder resolver sistemas lineares de até 10 equações com um máximo de 5000 iterações, se não falta memória no ambiente. Estes valores foram colocado de maneira arbitrária, e podem ser alterados de maneira simples no código.

### 2.1 Organização do programa

O programa consta em uma função principal,

`void main();` onde programamos uma apresentação do projeto e um menu de eleição do método a executar.

As funções que executam os algoritmos dos métodos iterativos contam com um menu para o ingresso dos parâmetros, dados da matriz a resolver, chute inicial e precisão requerida, assim como um número máximo de iterações a utilizar no caso em que o método diverge.

Não contam com parâmetros de entrada, pois estes serão ingressados de forma interativa pelo usuário. As funções estão divididas em 4 partes; a definição das variáveis locais; o menu de carga de dados; as iterações do respectivo método; e o cálculo de erro e apresentação da solução.

```
void gaussJacobi();
void gaussSeidel();
void osDois();
void osDoisForcaBruta();
```

As funções auxiliares, que sustentaram o cálculo dos métodos:

```
float precisao(float *aux_j, float *x, int n);
```

 esta função recebe como

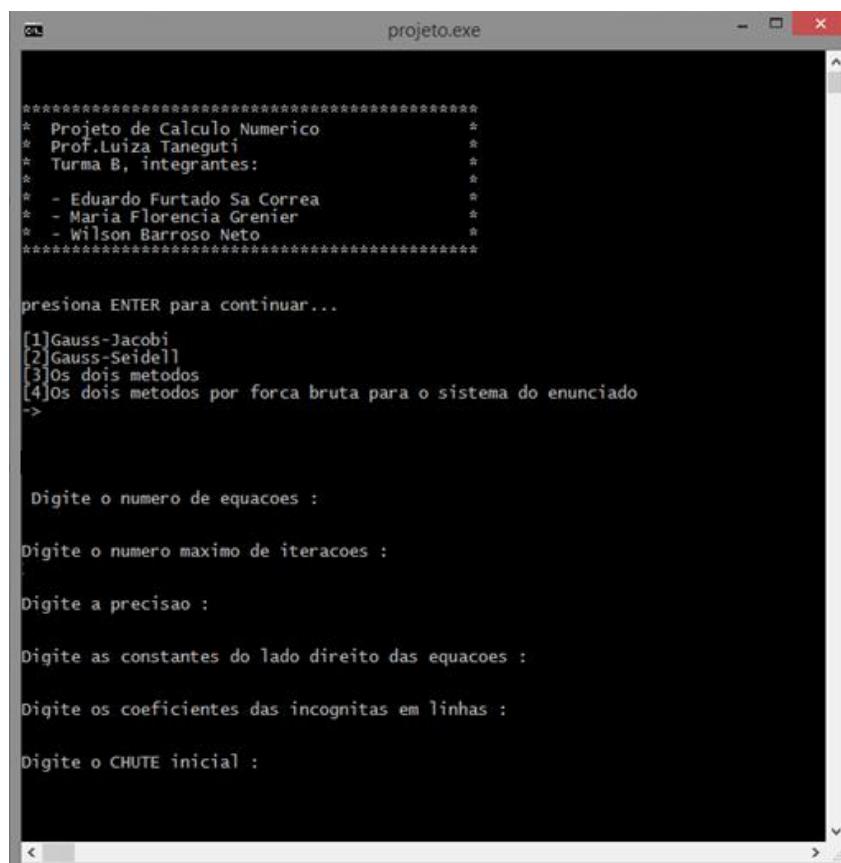
parâmetros dois pontos a arrays contendo os dados da anteúltima e última iteração, e o número de equações do sistema, e devolverá o cálculo de precisão que se utilizará como critério de parada das iterações.

`void mostrarSolucao(float *x, int n, int m, float pres);` auxilia na apresentação das soluções dos métodos na tela, recebendo como parâmetros um ponto ao array que contem a solução da última iteração calculada, o numero de equações do sistema, o número de iterações efetuadas e a precisão da ultima iteração.

`int getMenor(int a, int b);` função simples que cálculo o mínimo entre dois inteiros. Vamos a utilizá-la para a apresentação dos dados quando são executados os dois métodos na opção três do menu do nosso programa.

Dado que nem toda configuração de sistemas lineares converge, implementamos como esforço extra o método `void osDoisForcaBruta();`, que testa todas as configurações possíveis para o sistema linear. Este método não usa uma interface interativa e sua parametrização é feita diretamente no código.

## 2.2 Apresentação do programa



### 3. Código fonte

```
26 int main() {
27     int opcao;
28
29     //criação da tela inicial de apresentação
30     printf("\n\n\n*****\n");
31     printf("* Projeto de Calculo Numerico           *\n");
32     printf("* Prof.Luiza Taneguti                   *\n");
33     printf("* Turma B, integrantes:                 *\n");
34     printf("*                                         *\n");
35     printf("* - Eduardo Furtado Sa Correa          *\n");
36     printf("* - Maria Florencia Grenier            *\n");
37     printf("* - Wilson Barroso Neto                *\n");
38     printf("*****\n\n");
39     printf("presiona ENTER para continuar...");  

40     getchar();
41
42
43
44     // Menú de escolha de opção
45     label_menu_escolha_de_opcao:
46     printf("\n[1]Gauss-Jacobi \n[2]Gauss-Seidell \n[3]Os dois metodos\n[4]Os dois
47         metodos por força bruta para o sistema do enunciado\n->");  

48     scanf("%d", &opcao);
49
50     if(opcao == 1)
51     {
52         gaussJacobi();
53     }
54     else if(opcao == 2)
55     {
56         gaussSeidel();
57     }
58     else if(opcao == 3)
59     {
60         osDois();
61     }
62     else if(opcao == 4)
63     {
64         osDoisForcaBruta();
65     }
66     else
67     {
68         printf("\n Valor ingresado invalido \n");
69         goto label_menu_escolha_de_opcao;
70     }
71
72     printf("\n\nDone!\nBye.\n\n");
73     return 0;
74 }
```

```

67 void gaussJacobi(int mostrar) {
68
69     int i,j;
70     int m,n,l;
71     float x[10];
72     float a[10][10];
73     float b[10];
74     float c[10];
75     // vetor auxiliar. supõe-se que o programa não vai ter que enfrentar sistemas com
76     // mais de 20 incógnitas
77     // para que facilitar o desenvolvimento e não termos que explicar alocação de
78     // memória dinâmica para nosso
79     // companheiro que não é programador.
80     float aux[20];
81     // float aux[n];
82     float e;
83     float pres;
84
85     // Menú de carga de dados
86     printf("\n Digite o numero de equações : \n");
87     scanf("%d",&n);
88
89     printf("\nDigite o numero maximo de iterações : \n");
90     scanf("%d",&l);
91
92     printf("\nDigite a precisão : \n");
93     scanf("%f",&e);
94
95     printf("\nDigite as constantes do lado direito das equações : \n");
96     for(i=0;i<n;i++) {
97         scanf("%f",&b[i]);
98     }
99
100    printf("\nDigite os coeficientes das incógnitas em linhas : \n");
101    for(i=0;i<n;i++) {
102        for(j=0;j<n;j++) {
103            scanf("%f",&a[i][j]);
104            // guarda o chute como iter 0
105            // r[0][i]= x[i];
106        }
107
108    printf("\nDigite o CHUTE inicial : \n");
109    for(i=0;i<n;i++) {
110        scanf("%f",&x[i]);
111    }
112
113    /* Iterações de Gauss-Jacobi */
114    m=1;
115    label_gauss_jacobi_loop:
116    for(i=0;i<n;i++) {
117        // guarda em aux a iteração ANTERIOR
118        aux[i]= x[i];
119        c[i]=b[i];
120        for(j=0;j<n;j++) {
121            if(i!=j) {
122                c[i]=c[i]-a[i][j]*x[j];
123            }
124        }
125    }
126
127    // resultado de X_i de la iteración m
128    for(i=0;i<n;i++) {
129        x[i]=c[i]/a[i][i];
130        // r[m][i]= x[i];
131    }
132    m++;
133
134    /* Calculo do erro*/
135    pres=precisao(aux, x, n);
136    // Se alcancei a precisão ou cheguei ao número máximo de iterações termino (NÃO
137    // entro no if), se não, continuo iterando (entro no if)
138    if((m<=l) && (e <= pres)){
139        goto label_gauss_jacobi_loop;
140    }
141    else {
142        mostrarSolucao(x,n,m,pres);
143    }
144}

```

```

145 void gaussSeidel(){
146
147     int i,j,q;
148     int m,n,l;
149     float x[10];
150     float a[10][10];
151     float b[10];
152     // vetor auxiliar. supõe-se que o programa não vai ter que enfrentar sistemas com
153     // mais de 20 incógnitas
154     // para que facilitar o desenvolvimento e não termos que explicar alocação de
155     // memória dinâmica para nosso
156     // companheiro que não é programador.
157     float aux[20];
158     // float aux[n];
159     float e,c,pres;
160
161     //Menú de carga de dados
162     printf("\n Digite o numero de equações : \n");
163     scanf("%d",&n);
164
165     printf("\nDigite o numero maximo de iterações : \n");
166     scanf("%d",&l);
167
168     printf("\nDigite a precisão : \n");
169     scanf("%f",&e);
170
171     printf("\nDigite as constantes do lado direito das equações : \n");
172     for(i=0;i<n;i++) {
173         scanf("%f",&b[i]);
174     }
175
176     printf("\nDigite os coeficientes das incógnitas em linhas : \n");
177     for(i=0;i<n;i++) {
178         for(j=0;j<n;j++) {
179             scanf("%f",&a[i][j]);
180         }
181
182         printf("\nDigite o CHUTE inicial : \n");
183         for(i=0;i<n;i++) {
184             scanf("%f",&x[i]);
185             //guarda o chute como iter 0
186             // r[0][i]= x[i];
187         }
188
189         /*Iteraciones de Gauss-Siedel*/
190         m=1;
191         label_iterar_funcao_GS:
192         for(i=0;i<n;i++) {
193             //guardo en aux la iteración ANTERIOR
194             aux[i]= x[i];
195             c=b[i];
196             for(j=0;j<n;j++) {
197                 if(i!=j) {
198                     c=c-a[i][j]*x[j];
199                 }
200             // resultado de X_i de la iteración m
201             x[i]=c/a[i][i];
202         }
203
204         for(q=0;q<n;q++) {
205             // r[m][q]= x[q];
206         }
207         m++;
208
209         /*Calculo do erro*/
210         pres=precisao(aux, x, n);
211         // Se alcancei a precisão ou cheguei ao número máximo de iterações termine (NÃO
212         // entre no if), se não, continuo iterando (entre no if)
213         if((m<=l) && (e <= pres) )
214         {
215             // printf("\n Entrei no if, presisao= %f/n", pres);
216             goto label_iterar_funcao_GS;
217         }
218         else {
219             mostrarSolucao(x,n,m,pres);
220         }

```

```

222 void osDois(){
223
224     int i;
225     int j;
226     int q;
227     int k;
228     int mGJ;
229     int mGS;
230     int n;
231     int l;
232     int menor;
233
234     float x[10];
235     float y[10];
236     float a[10][10];
237     float b[10];
238
239     float cGJ[10];
240     float GJ[500][10];
241     float GS[500][10];
242
243     // vetor auxiliar. supõe-se que o programa não vai ter que enfrentar
244     // sistemas com mais de 20 incógnitas
245     // para que facilitar o desenvolvimento e não termos que explicar alocação
246     // de memória dinâmica para nosso
247     // companheiro que não é programador.
248     float aux[20];
249     float e;
250     float cGS;
251     float presGJ;
252     float presGS;
253
254     //Menú de carga de dados
255     printf("\n Digite o numero de equações : \n");
256     scanf("%d",&n);
257
258     printf("\nDigite o numero maximo de iteracoes : \n");
259     scanf("%d",&l);
260
261     printf("\nDigite a precicao : \n");
262     scanf("%f",&e);
263
264     printf("\nDigite as constantes do lado direito das equações : \n");
265     for(i=0;i<n;i++) {
266         scanf("%f",&b[i]);
267     }
268
269     printf("\nDigite os coeficientes das incógnitas em linhas : \n");
270     for(i=0;i<n;i++) {
271         for(j=0;j<n;j++) {
272             scanf("%f",&a[i][j]);
273         }
274     }
275
276     printf("\nDigite o CHUTE inicial : \n");
277     for(i=0;i<n;i++) {
278         scanf("%f",&x[i]);
279         y[i]= x[i];
280         //guarda o chute como iter 0
281         GJ[0][i]= x[i];
282         GS[0][i]= y[i];
283     }

```

```

285 /*Iteraciones de Gauss-Jacobi*/
286 mGJ=1;
287
288 label_os_dois_loop:
289 for(i=0;i<n;i++) {
290     //guardo en aux la iteración ANTERIOR
291     aux[i]= x[i];
292     cGJ[i]=b[i];
293     for(j=0;j<n;j++) {
294         if(i!=j) {
295             cGJ[i]=cGJ[i]-a[i][j]*x[j];
296         }
297     }
298 }
299
300 // resultado de X_i de la iteración m
301 for(i=0;i<n;i++) {
302     x[i]=cGJ[i]/a[i][i];
303     GJ[mGJ][i]= x[i];
304 }
305 mGJ++;
306
307 /*Calculo do erro*/
308 presGJ=precisao(aux, x, n);
309 // Se alcancei a precisão ou cheguei ao número máximo de iterações termine
// (NÃO entre no if), se não, continuo iterando (entre no if)
310 if((mGJ<=1) && (e <= presGJ ) {
311     goto label_os_dois_loop;
312 }
313
314 //-----
315 -----/*Iteraciones de Gauss-Siedel*/
316 mGS=1;
317 label_iterar_gauss_seidel:
318
319 for(i=0;i<n;i++) {
320     //guardo en aux la iteración ANTERIOR
321     aux[i]= y[i];
322     cGS=b[i];
323     for(j=0;j<n;j++) {
324         if(i!=j) {
325             cGS=cGS-a[i][j]*y[j];
326         }
327     }
328     // resultado de X_i de la iteración m
329     y[i]=cGS/a[i][i];
330 }
331
332
333 for(q=0;q<n;q++) {
334     GS[mGS][q]= y[q];
335 }
336 mGS++;
337
338 /*Calculo do erro*/
339 presGS=precisao(aux, y, n);
340
341 // Se alcancei a precisão ou cheguei ao número máximo de iterações termine
// (NÃO entre no if), se não, continuo iterando (entre no if)
342 if((mGS<=1) && (e <= presGS) )
343 {
344     goto label_iterar_gauss_seidel;
345 }

```

```

362     printf("\nITERACAO----- Gauss-Jacobi ----- Gauss-
363             Siedell ----- Diferencias----- \n");
364
365     for(i=0;i<menor;i++) {
366         printf(" %d :",i);
367         for(j=0;j<n;j++) {
368             printf(" %f ",GJ[i][j]);
369         }
370         printf(" ");
371         for(k=0;k<n;k++) {
372             printf(" %f ",GS[i][k]);
373         }
374         printf(" ");
375         for(l=0;l<n;l++) {
376             printf(" %f ",fabs(GJ[i][l]-GS[i][l]));
377         }
378         printf("\n");
379     }
380     if(mGJ > mGS){
381         for(i=mGJ;i<mGS;i++) {
382             printf(" %d :",i);
383             for(j=0;j<n;j++) {
384                 printf(" %f ",GJ[i][j]);
385             }
386             printf(" ");
387             for(k=0;k<n;k++) {
388                 printf(" - ");
389             }
390             printf(" ");
391             for(l=0;l<n;l++) {
392                 printf(" - ");
393             }
394             printf("\n");
395     }else if(mGJ<mGS){
396         for(i=mGJ;i<mGS;i++) {
397             printf(" %d :",i);
398             for(j=0;j<n;j++) {
399                 printf(" - ");
400             }
401             printf(" ");
402             for(k=0;k<n;k++) {
403                 printf(" %f ",GS[i][j]);
404             }
405             printf(" ");
406             for(l=0;l<n;l++) {
407                 printf(" - ");
408             }
409             printf("\n");
410     }
411 }
412 ...

```

```

int getMenor(int a, int b){
    if(a<b)
        return a;
    else
        return b;
}

float precisao(float *aux_j, float *x, int n)
{
    int i;
    double dif[n];
    double max_dif;
    float retorno_max_diff;

    for (i =0; i< n; i++)
    {
        //fabs dá o valor absoluto de um double
        dif[i] = fabs( ( x[i] - aux_j[i] ) );
    }

    max_dif = dif[0];

    for (i =1; i < n; i++) {
        if (max_dif < dif[i])
        {
            max_dif = dif[i];
        }
    }

    retorno_max_diff = (float)max_dif;
    return retorno_max_diff;
}

void mostrarSolucao(float *x, int n, int m, float pres){
    int i;
    int max_x;
    float erroRelativo;
    printf("\n A Solução é : \n");
    for(i=0;i<n;i++) {
        // mostro o resultado da iteração final
        printf("x(%d) = %f\n",i,x[i]);
    }

    max_x = x[0];
    for (i =1; i < n; i++)
    {
        if (max_x < fabs(x[i]))
        {
            max_x = fabs(x[i]);
        }
    }

    erroRelativo = (float)((double)pres/(double)max_x);
    //mostro o erro relativo na ultima iteração realizada
    printf("\n dr(%d): %f \n",m-1,erroRelativo);
}

```

## 4. Resultados

Dados do problema

$$x(0) = [0 \ 0 \ 0 \ 0]^T \text{ e } \epsilon=0,005$$

$$\begin{aligned} 3.210x + 7.890y + 4.950z + -6.350w &= 31.070 \\ 8.050x + -4.700y + 5.050z + 1.250w &= 38.270 \\ 0.320x + 9.050y + -4.770z + -4.780w &= -20.670 \\ 2.020x + 7.770y + -3.040z + -6.720w &= 0.000 \end{aligned}$$

Inicialmente verificamos se o sistema aceitava algum critério de convergência.

### Critério de Linhas

$$( |7.890| + |4.950| + |-6.350| ) / |3.10| > 1$$

$$( |8.050| + |5.050| + |1.250| ) / |-4.700| > 1$$

$$( |0.320| + |9.050| + |-4.780| ) / |-4.770| > 1$$

$$( |2.020| + |7.770| + |-3.040| ) / |-6.720| > 1$$

Não atende ao critério.

### Critério de Sassenfel

$$\beta_1 = ( |7.890| + |4.950| + |-6.350| ) / |3.10| > 1$$

$$\beta_2 = ( |8.050| + |5.050| + |1.250| ) / |-4.700| > 1$$

$$\beta_3 = ( |0.320| + |9.050| + |-4.780| ) / |-4.770| > 1$$

$$\beta_4 = ( |2.020| + |7.770| + |-3.040| ) / |-6.720| > 1$$

$$\text{Max}(\beta_1, \beta_2, \beta_3, \beta_4) > 1$$

Não atende ao critério

Logo, em nenhum dos dois métodos podemos garantir convergência.

Ao executar o programa, também apresentou divergência, a seguir estão parte dos resultados:

ITERAÇÃO----- Gauss-Jacobi ----- Gauss-Siedell ----- Diferencias-----

0 : 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
1 : 9.679128 -8.142553 -0.000000 4.324267 9.679128 8.435528 27.992067 -6.990232 0.000000 16.578081 27.992067 11.314499  
2 : 38.247303 9.585599 -23.939106 -10.444098 -68.048286 -96.475868 -276.348907 92.881210 106.295589 106.061467 252.409801 103.325308  
3 : 2.373264 28.866636 73.001312 48.922245 856.693848 1186.947998 3397.679443 -1081.640015 854.320584 1158.081362 3324.678131 1130.562260  
4 : -77.067909 87.371117 -32.786095 -13.712079 -10286.876953 -14264.127930 -40902.351562 13126.093750 10209.809044 14351.499046 40869.565468  
13139.805829  
5 : -181.641312 -179.016296 202.414993 197.302628 124109.765625 172105.562500 493340.312500 -158146.875000 124291.406937 172284.578796  
493137.897507 158344.177628  
6 : 527.858582 -49.289051 -1014.390015 -548.759888 -1496619.625000 -2075349.500000 -5949306.500000 1907402.000000 1497147.483582 2075300.210949  
5948292.109985 1907950.759888  
7 : 609.522034 -339.920837 1437.817383 958.610657 18048476.000000 25027756.000000 71745344.000000 -23001788.000000 18047866.477966  
25028095.920837 71743906.182617 23002746.610657  
8 : 524.307556 2835.664795 -2582.833496 -2033.254150 -217654128.000000 -301820032.000000 -865207616.000000 277388992.000000 217654652.307556  
301822867.664795 865205033.166504 277391025.254150  
9 : -6999.520996 -2426.057861 12090.685547 7985.633789 2624784384.000000 3639777536.000000 10433908736.000000 -3345151744.000000  
2624791383.520996 3639779962.057861 10433896645.314453 3345159729.633789  
10 : 3125.404053 3118.211426 -28504.263672 -17122.921875 -31653398528.000000 -43893633024.000000 -125826965504.000000 40340619264.000000  
31653401653.404053 43893636142.211426 125826936999.736328 40340636386.921875  
11 : 2428.016113 -29835.949219 47897.316406 34561.914062 381721837568.000000 529332011008.000000 1517400883200.000000 -486484574208.000000  
381721835139.983887 529332040843.949219 1517400835302.683594 486484608769.914062  
12 : 67854.562500 64806.625000 -151045.000000 -104118.804688 -4603346157568.000000 -6383440232448.000000 -18298984988672.000000  
5866726096896.000000 4603346225422.562500 6383440297254.625000 18298984837627.000000 5866726201014.804688  
13 : -132328.625000 -73773.359375 440885.531250 277974.625000 55513710592000.000000 76980619837440.000000 220675251896320.000000 -  
70749352099840.000000 55513710724328.625000 76980619911213.359375 220675251455434.468750 70749352377814.625000  
14 : 51357.777344 320990.750000 -890958.000000 -588493.125000 -669463531749376.000000 -928342215753728.000000 -2661217034305536.000000  
853196595527680.000000 669463531800733.750000 928342216074718.750000 2661217033414578.000000 853196596116173.125000  
15 : -579213.500000 -1025864.375000 2155432.500000 1500270.000000 8073346126184448.000000 11195273564913664.000000 3209274725498800.000000 -  
10289060161519616.000000 8073346126763662.000000 11195273565939528.000000 32092747252833368.000000 10289060163019886.000000  
16 : 2165553.000000 1722886.250000 -6323287.000000 -4131968.750000 -97359916393562112.000000 -135008586098016256.000000 -  
38702015587758992.000000 124080187842232320.000000 97359916395727664.000000 135008586099739136.000000 387020155871265728.000000  
124080187846364288.000000

Na conformação fornecida, o sistema diverge.

Obtendo esses resultados, optamos por permutar linhas e colunas a mão e verificar se algum dos métodos garantia convergência. Ao não obter bons resultados, decidimos criar um programa que gerava todos as permutações possíveis de linhas e colunas.

Então, como temos um sistema de 4 linhas e 4 colunas, temos  $4! \times 4!$  permutações possíveis.

Obtivemos a seguinte permutação do sistema que converge pelo método de Gauss - Seidel

Tomando os dados iniciais,  $x(0) = [0 \ 0 \ 0 \ 0]^T$  e  $e=0,005$  e aplicando na seguinte conformação:

$$\begin{array}{cccccc} 8.050 & -4.700 & 5.050 & 1.250 & = 38.270 \\ 3.210 & 7.890 & 4.950 & -6.350 & = 31.070 \\ 0.320 & 9.050 & -4.770 & -4.780 & = -20.670 \\ 2.020 & 7.770 & -3.040 & -6.720 & = 0.000 \end{array}$$

Obtivemos convergência para Gauss-Seidel na trigésima iteração e divergência no método G-J com número de iterações 400 como pode ver-se a seguir:

ITERAÇÃO	Gauss-Jacobi	Gauss-Siedell	Diferencias
0	0.000000 0.000000 0.000000 0.000000	0.000000 0.000000 0.000000 0.000000	0.000000 0.000000 0.000000 0.000000
1	4.754037 3.937896 4.333333 -0.000000	4.754037 2.003744 8.453916 -0.078521	0.000000 1.934152
2	4.334754 -0.714887 12.123532 4.021916	0.632729 -1.686508 1.254696 -2.327430	3.702025 0.971621
3	-3.893317 -2.194788 -0.762551 -5.008037	3.343663 -0.082774 6.732910 -2.136459	7.236980 2.112014
4	4.728624 1.969727 4.926567 -3.363073	0.813707 -2.336682 2.095533 -3.405171	3.914918 4.306408
5	3.335701 -3.783385 11.757794 1.470214	2.603928 -1.176721 5.687767 -3.150893	0.731773 2.606663
6	-5.059197 -3.612525 -4.094304 -8.690845	0.988174 -2.568397 2.684170 -3.886936	6.047372 1.044128
7	6.562847 1.570337 5.849046 -3.845568	2.174180 -1.758910 5.037140 -3.658898	4.388667 3.329247
8	2.598741 -5.496689 11.606600 1.142466	1.135303 -2.628918 3.088285 -4.095501	1.463438 2.867771
9	-5.913770 -3.481618 -7.065917 -10.824983	1.917717 -2.075959 4.627409 -3.917222	7.831487 1.405659
10	8.834846 2.064748 8.178694 -2.606785	1.247342 -2.625349 3.361439 -4.181265	7.587504 4.690096
11	1.233589 -6.885612 11.455671 1.343192	1.761761 -2.252904 4.367176 -4.050971	0.528172 4.632708
12	-6.661180 -2.669977 -9.993815 -12.773005	1.328053 -2.602564 3.544111 -4.213297	7.989233 0.067413
13	11.447967 2.637920 11.620565 -0.568469	1.665442 -2.354100 4.200816 -4.121674	9.782524 4.992021

14 : -0.907459 -8.467624 10.675851 1.234378 1.384311 -2.577985 3.665370 -4.222822 2.291770 5.889639  
 7.010481  
 15 : -7.078739 -1.397239 -13.029919 -14.893021 1.605202 -2.413333 4.093939 -4.159920 8.683941 1.016094  
 17.1238  
 16 : 14.424888 3.006359 16.131746 2.151094 1.422714 -2.557338 3.745445 -4.223630 13.002174 5.563697  
 12.386301  
 17 : -3.944634 -10.320226 8.849325 0.514460 1.567149 -2.448739 4.025021 -4.181120 5.511782 7.871487  
 4.824304  
 18 : -6.902761 0.404935 -16.027138 -17.121765 1.448568 -2.541682 3.798129 -4.221588 8.351329 2.946617  
 19.82526  
 19 : 17.703409 3.021418 21.796188 5.643641 1.542923 -2.470293 3.980457 -4.193163 16.160486 5.491710  
 17.815731  
 20 : -8.031632 -12.396953 5.597970 -1.045105 1.465810 -2.530431 3.832701 -4.219036 9.497442 9.866522  
 1.765270  
 21 : -5.833422 2.852366 -18.678604 -19.280664 1.527408 -2.483615 3.951582 -4.200169 7.360829 5.335981  
 22.63018  
 22 : 21.130920 2.512313 28.674799 9.994394 1.477234 -2.522601 3.855343 -4.216792 19.653686 5.034914  
 24.819456  
 23 : -13.319611 -14.605336 0.502121 -3.715218 1.517427 -2.491953 3.932844 -4.204333 14.837037 12.113383  
 3.4307  
 24 : -3.511399 6.051815 -20.547552 -21.118380 1.484767 -2.517262 3.870151 -4.215053 4.996166 8.569077  
 24.41770  
 25 : 24.456726 1.261123 36.742378 15.237223 1.510984 -2.497223 3.920670 -4.206856 22.945742 3.758346  
 32.821708  
 26 : -19.925249 -16.800316 -6.902433 -7.811802 1.489719 -2.513669 3.879827 -4.213788 21.414968 14.286646  
 10.78  
 27 : 0.488268 10.087726 -21.050007 -22.292273 1.506815 -2.500579 3.912756 -4.208409 1.018547 12.588305  
 24.9627  
 28 : 27.310581 -0.995657 45.844284 21.333326 1.492966 -2.511275 3.886144 -4.212901 25.817616 1.515618  
 41.95814  
 29 : -27.899364 -18.765469 -17.101597 -13.680879 1.504113 -2.502730 3.907606 -4.209379 29.403477  
 16.262739 21.  
 30 : 6.650492 15.007133 -19.432013 -22.347553 **1.495091** **-2.509689** **3.890267** **-4.212293** 5.155401  
 17.516822 23.3222  
 31 : 29.176353 -4.562304 55.646545 28.141779 - - - - - - - -  
 32 : -37.188213 -20.194704 -30.566065 -21.678329 - - - - - - - -  
 33 : 15.504527 20.797058 -14.752597 -20.701223 - - - - - - - -  
 34 : 29.365627 -9.775279 65.575829 35.380981 - - - - - - - -  
 35 : -47.584843 -20.674877 -47.698193 -32.140755 - - - - - - - -  
 36 : 27.596262 27.354830 -5.876726 -16.631350 - - - - - - - -  
 37 : 26.994328 -16.987741 74.750496 42.582844 - - - - - - - -  
 38 : -58.669682 -19.669924 -68.758255 -45.343410 - - - - - - - -  
 39 : 43.444698 34.451508 8.516646 -9.274250 - - - - - - - -  
 40 : 20.965942 -26.544533 81.905525 49.041058 - - - - - - - -  
 41 : -69.740807 -16.508528 -93.766289 -61.442356 - - - - - - - -  
 42 : 63.478588 41.688484 29.904661 2.366341 - - - - - - - -  
 43 : 9.966385 -38.745003 85.315063 53.755375 - - - - - - - -  
 44 : -79.735008 -10.378331 -122.376053 -80.397972 - - - - - - - -  
 45 : 87.948868 48.447872 59.860218 19.392687 - - - - - - - -  
 46 : -7.522981 -53.790928 82.719040 55.375240 - - - - - - - -  
 47 : -87.142555 -0.330504 -153.718842 -101.877655 - - - - - - - -  
 48 : 116.812904 53.838116 99.951469 42.962692 - - - - - - - -  
 49 : -33.186230 -71.716866 71.262772 52.147587 - - - - - - - -  
 50 : -89.920593 14.700160 -186.216507 -125.136169 - - - - - - - -  
 51 : 149.586868 56.637791 151.589676 74.208160 - - - - - - - -  
 52 : -68.797577 -92.300507 47.462215 41.876263 - - - - - - - -  
 53 : -85.412659 35.853806 -217.365494 -148.873703 - - - - - - - -  
 54 : 185.164078 55.241539 215.813690 114.113342 - - - - - - - -  
 55 : -116.098831 -114.951180 7.211040 21.902466 - - - - - - - -  
 56 : -70.285004 64.275444 -243.497620 -171.073196 - - - - - - - -  
 57 : 221.598648 47.615116 292.998199 163.344833 - - - - - - - -

58	:	-176.616272	-138.575821	-54.148842	-10.880331	-	-	-	-	-	-
59	:	-40.494938	101.008224	-259.528381	-188.822418	-	-	-	-	-	-
60	:	255.857758	31.267670	382.475311	222.023865	-	-	-	-	-	-
61	:	-251.403992	-161.423889	-141.668182	-59.961689	-	-	-	-	-	-
62	:	8.689964	146.841278	-258.710388	-198.129242	-	-	-	-	-	-
63	:	283.549438	3.253607	482.059143	289.433075	-	-	-	-	-	-
64	:	-340.699158	-180.914627	-260.511353	-129.078781	-	-	-	-	-	-
65	:	82.596542	202.103348	-232.418137	-193.744720	-	-	-	-	-	-
66	:	298.639526	-39.781258	587.470886	363.651642	-	-	-	-	-	-
67	:	-443.477539	-193.454483	-415.522156	-221.988098	-	-	-	-	-	-
68	:	186.944855	266.393280	-170.000565	-169.014694	-	-	-	-	-	-
69	:	293.178833	-101.490944	691.664978	441.117004	-	-	-	-	-	-
70	:	-556.899597	-194.256332	-610.596497	-342.116821	-	-	-	-	-	-
71	:	327.506042	338.241882	-58.750370	-115.788025	-	-	-	-	-	-
72	:	257.072174	-185.635498	784.072937	516.116516	-	-	-	-	-	-
73	:	-675.643555	-177.180313	-847.820801	-492.066071	-	-	-	-	-	-
74	:	509.577698	414.699524	115.945099	-24.421989	-	-	-	-	-	-
75	:	177.933197	-295.777222	849.790894	580.221558	-	-	-	-	-	-
76	:	-791.130920	-134.620239	-1126.338379	-672.935791	-	-	-	-	-	-
77	:	737.234009	490.853363	370.194580	116.069183	-	-	-	-	-	-
78	:	41.082184	-434.838379	868.762573	621.689209	-	-	-	-	-	-
79	:	-890.662476	-57.471714	-1440.911011	-883.444458	-	-	-	-	-	-
80	:	1012.305542	559.281799	720.839355	317.660126	-	-	-	-	-	-
81	:	-170.238480	-604.492249	815.029846	624.869812	-	-	-	-	-	-
82	:	-956.500793	64.773285	-1780.154907	-1118.821045	-	-	-	-	-	-
83	:	1333.045166	609.466492	1184.224976	592.682678	-	-	-	-	-	-
84	:	-474.338776	-804.357788	656.162048	569.681824	-	-	-	-	-	-
85	:	-964.960510	243.748428	-2124.451904	-1369.457886	-	-	-	-	-	-
86	:	1692.446167	627.195496	1774.384644	952.833130	-	-	-	-	-	-
87	:	-890.136108	-1030.975342	353.004089	431.238220	-	-	-	-	-	-
88	:	-885.594116	491.685608	-2443.568115	-1619.328247	-	-	-	-	-	-
89	:	2076.195313	594.012024	2500.508057	1407.729858	-	-	-	-	-	-
90	:	-1435.664795	-1276.545898	-140.060318	179.738586	-	-	-	-	-	-
91	:	-680.604248	820.556702	-2694.053223	-1844.199707	-	-	-	-	-	-
92	:	2460.261475	486.781250	3363.562012	1962.920654	-	-	-	-	-	-
93	:	-2125.900391	-1527.433960	-874.095825	-219.227661	-	-	-	-	-	-
94	:	-304.652252	1240.797119	-2816.559082	-2009.706787	-	-	-	-	-	-
95	:	2808.170654	277.481964	4351.948730	2617.252441	-	-	-	-	-	-
96	:	-2969.748047	-1762.452515	-1903.557861	-803.777466	-	-	-	-	-	-
97	:	294.712128	1759.516479	-2733.289063	-2069.394531	-	-	-	-	-	-
98	:	3068.055908	-66.645233	5436.123535	3359.518066	-	-	-	-	-	-
99	:	-3966.060059	-1950.983521	-3282.848633	-1614.014282	-	-	-	-	-	-
100	:	1175.720703	2378.103027	-2345.887451	-1962.905518	-	-	-	-	-	-
101	:	3169.654541	-582.420898	6562.143555	4164.332520	-	-	-	-	-	-
102	:	-5098.552246	-2051.029053	-5061.103027	-2689.229980	-	-	-	-	-	-
103	:	2399.818848	3089.134766	-1534.205444	-1614.556152	-	-	-	-	-	-
104	:	3021.507568	-1309.311035	7644.205566	4987.231445	-	-	-	-	-	-
105	:	-6329.536621	-2007.334106	-7274.775879	-4063.733154	-	-	-	-	-	-
106	:	4027.464111	3872.537354	-156.501236	-932.638733	-	-	-	-	-	-
107	:	2508.736084	-2287.030029	8556.379883	5759.056641	-	-	-	-	-	-
108	:	-7592.460938	-1749.815186	-9937.620117	-5761.007324	-	-	-	-	-	-
109	:	6111.848145	4690.962402	1948.190796	190.108749	-	-	-	-	-	-
110	:	1491.899658	-3551.877930	9123.889648	6379.793457	-	-	-	-	-	-
111	:	-8783.345703	-1192.584595	-13027.637695	-7785.875000	-	-	-	-	-	-
112	:	8690.066406	5484.424805	4954.631348	1874.297485	-	-	-	-	-	-
113	:	-192.384430	-5131.518555	9114.547852	6712.178223	-	-	-	-	-	-
114	:	-9751.374023	-233.974106	-16470.724609	-10114.396484	-	-	-	-	-	-
115	:	11771.273438	6164.340332	9041.840820	4249.292969	-	-	-	-	-	-
116	:	-2728.229248	-7037.873047	8231.265625	6575.551270	-	-	-	-	-	-
117	:	-10289.076172	1241.910278	-20120.806641	-12681.301758	-	-	-	-	-	-

Assim, tentamos com diferentes condições iniciais;

- $X^{(0)} = [8 \ 6 \ 9 \ 4]^T$ , com precisão 0,005 com uma cota de 400 iterações, mostrando só 50, dado que achamos que era representativa da divergência do método Gauss-Seidel

Obtivemos

ITERAÇÃO	Gauss-Jacobi	Gauss-Siedell	Diferenças
0	8.000000 6.000000 9.000000 4.000000	8.000000 6.000000 9.000000 4.000000	0.000000 0.000000 0.000000 0.000000
1	1.990062 -1.743980 12.245282 5.270833	1.990062 0.701128 1.788685 0.599715	0.000000 2.445108 10.456597 4.671118
2	-4.764462 -0.312098 -4.123853 -6.957806	3.948174 1.692086 7.207581 -0.117295	8.712636 2.004185 11.331434 6.840511
3	8.239236 2.863743 10.393963 0.072514	1.238651 -1.182309 2.290806 -2.031029	7.000585 4.046052 8.103157 2.103543
4	-0.105660 -5.876755 10.246711 1.085847	2.942033 -0.330855 5.938266 -2.184549	3.047693 5.545900 4.308444 3.270396
5	-5.273782 -1.573754 -7.911696 -11.462175	1.174835 -2.023767 2.761635 -3.236142	6.448617 0.450014 10.673331 8.226033
6	10.578279 1.822170 12.479898 0.174174	2.342511 -1.352225 5.167868 -3.197207	8.235767 3.174395 7.312031 3.371382
7	-2.038138 -8.055229 8.325604 -0.359004	1.219046 -2.373428 3.115979 -3.787447	3.257184 5.681801 5.209625 3.428442
8	-5.116164 -0.745120 -10.726623 -13.692859	2.001677 -1.879565 4.696954 -3.696365	7.117841 1.134445 15.423577 9.996494
9	13.174342 1.728773 16.297979 2.453081	1.284085 -2.506179 3.368682 -4.035707	11.890257 4.234952 12.929298 6.488788
10	-4.841728 -9.672695 6.038880 -1.413857	1.804193 -2.157563 4.405048 -3.945110	6.645921 7.515132 1.633832 2.531253
11	-4.462197 0.981178 -12.926416 -15.371328	1.343522 -2.547419 3.543692 -4.144696	5.805719 3.528597 16.470108 11.226632
12	15.822869 1.491949 21.299101 5.640837	1.687249 -2.307498 4.221951 -4.070796	14.135620 3.799447 17.077150 9.711633
13	-8.612342 -11.322262 2.572800 -3.153963	1.390360 -2.552759 3.662652 -4.190607	10.002702 8.769502 1.089852 1.036644
14	-2.980722 3.289303 -14.565300 -16.844080	1.616633 -2.390351 4.106026 -4.135379	4.597355 5.679654 18.671326 12.708701
15	18.427279 0.732122 27.253473 9.496330	1.424738 -2.545994 3.742512 -4.208577	17.002540 3.278117 23.510962 13.704907
16	-13.389997 -13.014520 -2.557657 -5.943284	1.573275 -2.437276 4.032096 -4.169225	14.963273 10.577244 6.589754 1.774058
17	-0.317139 6.206901 -15.301322 -17.915972	1.448975 -2.536713 3.795664 -4.214606	1.766113 8.743614 19.096986 13.701366
18	20.758890 -0.752455 34.041786 14.003426	1.546286 -2.464494 3.984687 -4.187362	19.212604 1.712040 30.057099 18.190787
19	-19.215137 -14.594566 -9.734433 -10.029858	1.465641 -2.528347 3.830830 -4.215831	20.680779 12.066220 13.565263 5.814027
20	3.897112 9.790423 -14.594748 -18.247274	1.529300 -2.480632 3.954157 -4.197316	2.367812 12.271055 18.548905 14.049959
21	22.459324 -3.176922 41.455425 19.094019	1.476917 -2.521791 3.853998 -4.215343	20.982407 0.655131 37.601427 23.309361
22	-26.071920 -15.840528 -19.321507 -15.675797	1.518518 -2.490387 3.934436 -4.202915	27.590438 13.350140 23.255942
11.472881			
23	10.060607 14.050825 -11.760897 -17.412025	1.484462 -2.516995 3.869215 -4.214413	8.576145 16.567820 15.630112 13.197613
24	23.039309 -6.790174 49.115063 24.590843	1.511628 -2.496383 3.921667 -4.206142	21.527681 4.293792 45.193396 28.796985
25	-33.840179 -16.458033 -31.646278 -23.144352	1.489473 -2.513620 3.879189 -4.213516	35.329653 13.944413 35.525467
18.930836			
26	18.591478 18.932753 -5.969401 -14.885624	1.507202 -2.500118 3.913385 -4.208044	17.084276 21.432870 9.882786 10.677580
27	: 21.864153 -11.861068 56.418026 30.179949	1.492784 -2.511302 3.885716 -4.212792	20.371369 9.349766 52.532310 34.392741
28	-42.250061 -16.063433 -46.946812 -32.664539	1.504349 -2.502470 3.908007 -4.209188	43.754410 13.560963 50.854818
28.455351			
29	29.898626 24.291456 3.755223 -10.035667	1.494962 -2.509734 3.889981 -4.212255	28.403664 26.801191 0.134758 5.823412
30	18.139196 -18.658991 62.483387 35.375591	- - - -	- - - -
31	-50.830791 -14.171683 -65.300766 -44.388195	- - - -	- - - -
32	44.337540 29.861929 18.516981 -2.124680	- - - -	- - - -
33	10.902628 -27.427664 66.093178 39.478779	- - - -	- - - -
34	-58.852055 -10.189914 -86.534607 -58.335255	- - - -	- - - -
35	62.148602 35.222118 39.509674 9.673873	- - - -	- - - -
36	-0.969224 -38.348648 65.634506 41.533726	- - - -	- - - -
37	-65.259636 -3.418307 -110.110405 -74.323769	- - - -	- - - -
38	83.374687 39.752090 67.949448 26.242697	- - - -	- - - -
39	-18.738295 -51.491936 59.049522 40.286358	- - - -	- - - -
40	-68.608711 6.938298 -134.988907 -91.883072	- - - -	- - - -
41	107.755005 42.590816 104.970215 48.465366	- - - -	- - - -
42	-43.755833 -66.751724 43.801689 34.149750	- - - -	- - - -
43	-66.999855 21.743828 -159.469772 -110.149521	- - - -	- - - -
44	134.593201 42.593849 151.473053 77.142555	- - - -	- - - -
45	-77.379631 -83.765594 16.870596 21.183689	- - - -	- - - -
46	-58.025391 41.884109 -181.012192 -127.745857	- - - -	- - - -
47	162.598663 38.296021 207.919983 112.872818	- - - -	- - - -

48 : -120.847908 -101.816628 -25.209969 -0.902878  
 49 : -38.736546 68.193642 -196.043213 -142.647278  
 50 : 189.702774 27.885670 274.063049 155.891098

- $X^{(0)} = [1 \ 5 \ 7 \ 5]^T$ , com precisão 0,005 com uma cota de 400 iterações, mostrando só 50, dado que achamos que era representativa da divergência do método Gauss-Seidel.

	ITERAÇÃO				Gauss-Jacobi				Gauss-Siedell				Diferencias			
0	: 1.000000	5.000000	7.000000	5.000000	1.000000	5.000000	7.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	: 2.505590	3.163498	8.876309	2.915179	2.505590	2.550958	4.330808	1.743538	0.000000	0.612540	4.545501	1.171641				
2	: 0.580013	-0.304092	7.582158	0.395490	3.255838	1.299458	5.269990	0.097145	2.675825	1.603551	2.312168	0.298345				
3	: -0.241428	-0.736649	3.398978	-3.607281	2.191623	-0.181837	4.038017	-1.378180	2.433051	0.554812	0.639039	2.229102				
4	: 2.751802	-0.999518	6.534355	-2.461956	2.328708	-0.652063	4.633482	-2.150049	0.423094	0.347454	1.900873	0.311907				
5	: 0.453567	-3.262581	5.088698	-3.284532	1.800469	-1.431946	3.891881	-2.875088	1.346902	1.830635	1.196817	0.409443				
6	: 0.166908	-2.082609	1.465167	-5.938049	1.922946	-1.600035	4.307746	-3.220755	1.756038	0.482574	2.842579	2.717295				
7	: 3.541020	-1.828259	6.343748	-3.020658	1.617598	-2.014908	3.846525	-3.583590	1.923422	0.186649	2.497223	0.562932				
8	: 0.176035	-3.913739	4.129168	-3.919302	1.721052	-2.059654	4.132165	-3.733448	1.545018	1.854085	0.002997	0.185854				
9	: 0.487236	-1.878583	0.847222	-6.340303	1.539006	-2.285402	3.841818	-3.917844	1.051770	0.406819	2.994596	2.422459				
10	: 4.110257	-1.894639	7.155428	-2.408917	1.617979	-2.283780	4.034977	-3.979610	2.492277	0.389141	3.120451	1.570693				
11	: -0.466903	-4.162220	3.428391	-4.192133	1.507343	-2.409662	3.850618	-4.075018	1.974247	1.752558	0.422227	0.117115				
12	: 0.824144	-1.396935	0.606057	-6.503855	1.564316	-2.393964	3.979831	-4.098195	0.740172	0.997029	3.373774	2.405660				
13	: 4.568154	-2.012036	8.255742	-1.641641	1.496021	-2.465897	3.861998	-4.148592	3.072133	0.453861	4.393744	2.506951				
14	: -1.344847	-4.421307	2.467490	-4.687992	1.535768	-2.448703	3.947789	-4.155573	2.880614	1.972605	1.480299	0.532419				
15	: 1.352673	-0.835978	0.552499	-6.632637	1.493072	-2.490774	3.872101	-4.182812	0.140398	1.654796	3.319602	2.449825				
16	: 4.949265	-2.297109	9.484541	-0.809932	1.520220	-2.476256	3.928762	-4.183497	3.429045	0.179147	5.555779	3.373565				
17	: -2.411295	-4.677907	1.118743	-5.458933	1.493257	-2.501386	3.879962	-4.198583	3.904552	2.176521	2.761219	1.260350				
18	: 2.168677	-0.176393	0.766673	-6.639752	1.511541	-2.490350	3.917245	-4.197192	0.657136	2.313956	3.150572	2.442560				
19	: 5.201111	-2.769190	10.797827	0.101111	1.494380	-2.505639	3.885691	-4.205754	3.706731	0.263551	6.912136	4.306866				
20	: -3.652249	-4.871071	-0.672982	-6.523178	1.506578	-2.497696	3.910159	-4.203973	5.158827	2.373375	4.583141	2.319205				
21	: 3.345155	0.596046	1.383412	-6.425581	1.495589	-2.507143	3.889714	-4.208950	1.849566	3.103188	2.506302	2.216632				
22	: 5.231946	-3.462393	12.127662	1.068886	-	-	-	-	-	-	-	-				
23	: -5.041497	-4.939043	-2.955914	-7.917017	-	-	-	-	-	-	-	-				
24	: 4.954054	1.471726	2.558015	-5.889019	-	-	-	-	-	-	-	-				
25	: 4.923033	-4.422048	13.359315	2.033652	-	-	-	-	-	-	-	-				
26	: -6.524252	-4.809614	-5.764155	-9.676653	-	-	-	-	-	-	-	-				
27	: 7.064549	2.420618	4.467429	-4.914680	-	-	-	-	-	-	-	-				
28	: 4.127917	-5.694448	14.324826	2.901429	-	-	-	-	-	-	-	-				
29	: -8.007586	-4.393464	-9.101185	-11.823652	-	-	-	-	-	-	-	-				
30	: 9.734320	3.389736	7.308969	-3.369782	-	-	-	-	-	-	-	-				
31	: 2.671266	-7.319985	14.794477	3.539035	-	-	-	-	-	-	-	-				
32	: -9.350289	-3.582326	-12.921938	-14.353503	-	-	-	-	-	-	-	-				
33	: 12.997607	4.296993	11.292998	-1.107079	-	-	-	-	-	-	-	-				
34	: 0.350319	-9.326060	14.467267	3.766680	-	-	-	-	-	-	-	-				
35	: -10.351619	-2.249566	-17.111839	-17.222670	-	-	-	-	-	-	-	-				
36	: 16.849710	5.023873	16.629618	2.028362	-	-	-	-	-	-	-	-				
37	: -3.059978	-11.717881	12.962765	3.350874	-	-	-	-	-	-	-	-				
38	: -10.739701	-0.252865	-21.461885	-20.332724	-	-	-	-	-	-	-	-				
39	: 21.227324	5.407856	23.508446	6.188270	-	-	-	-	-	-	-	-				
40	: -7.797027	-14.466540	9.816334	1.998893	-	-	-	-	-	-	-	-				
41	: -10.160726	2.560276	-25.639822	-23.511410	-	-	-	-	-	-	-	-				
42	: 25.984306	5.235183	32.069939	11.505020	-	-	-	-	-	-	-	-				
43	: -14.094298	-17.494164	4.479954	-0.643891	-	-	-	-	-	-	-	-				
44	: -8.170368	6.343247	-29.158186	-26.490953	-	-	-	-	-	-	-	-				
45	: 30.862831	4.234771	42.366592	18.069014	-	-	-	-	-	-	-	-				
46	: -22.157038	-20.656033	-3.668574	-4.992165	-	-	-	-	-	-	-	-				
47	: -4.229421	11.236158	-31.340628	-28.884247	-	-	-	-	-	-	-	-				
48	: 35.460300	2.074472	54.312481	25.898369	-	-	-	-	-	-	-	-				
49	: -32.128071	-23.719862	-15.304601	-11.512127	-	-	-	-	-	-	-	-				
50	: 2.293794	17.345612	-31.288841	-30.160126	-	-	-	-	-	-	-	-				

- $X^{(0)} = [2 \ 8 \ 5 \ 1]^T$ , com precisão 0,005 com uma cota de 400 iterações, mostrando só 50, dado que achamos que era representativa da divergência do método Gauss-Seidel.

ITERAÇÃO	Gauss-Jacobi	Gauss-Siedell	Diferenças
0	2.000000 8.000000 5.000000 1.000000	2.000000 8.000000 5.000000 1.000000	0.000000 0.000000 0.000000 0.000000
1	6.132918 0.792142 18.643606 7.589285	6.132918 -0.889312 2.055401 -0.114565	0.000000 1.681454 16.588205 7.703850
2	-7.657609 -4.145824 -1.357519 -5.674572	2.963188 1.350626 7.209435 -0.809029	10.620797 5.496451 8.566954 4.865543
3	4.066250 3.338037 1.640316 -6.481335	1.145538 -1.702309 1.991160 -2.524715	2.920712 5.040346 0.350844 3.956620
4	6.680353 -3.961813 17.434216 4.339852	2.903066 -0.524338 6.063284 -2.476532	3.777287 3.437475 11.370932 6.816384
5	-9.169953 -6.224998 -7.084106 -10.459671	1.028782 -2.277772 2.562515 -3.483660	10.198735 3.947226 9.646621 6.976011
6	7.187806 3.694926 2.389227 -6.749384	2.357558 -1.432630 5.264363 -3.329305	4.830248 5.127557 2.875136 3.420079
7	6.460532 -5.917379 18.589359 5.352038	1.132079 -2.504900 2.993082 -3.910006	5.328453 3.412479 15.596277 9.262044
8	-11.193540 -6.045652 -11.823407 -13.309436	2.021045 -1.908980 4.765262 -3.755455	13.214585 4.136672 16.588669 9.553981
9	10.708129 5.197999 5.449481 -5.006325	1.233235 -2.575902 3.292200 -4.097012	9.474894 7.773901 2.157280 0.909314
10	5.147655 -7.866691 19.930550 6.763757	1.820983 -2.165750 4.452073 -3.970803	3.326672 5.700940 15.478477 10.734560
11	-13.392225 -5.216772 -17.024542 -16.564703	1.313231 -2.585276 3.495581 -4.175809	14.705456 2.631496 20.520123 12.388894
12	14.960371 6.735698 10.136683 -2.355954	1.700159 -2.307608 4.253788 -4.081445	13.260212 9.043306 5.882896 1.725491
13	2.693474 -10.404268 20.477327 7.699526	1.371978 -2.573824 3.632124 -4.206678	1.321496 7.830444 16.845203 11.906204
14	-15.362108 -3.808217 -22.941395 -20.483841	1.625981 -2.387936 4.127340 -4.139419	16.988089 1.420281 27.068735 16.344422
15	20.103134 8.095043 16.604303 1.357223	1.413404 -2.558007 3.723009 -4.217051	18.689730 10.653050 12.881294 5.574274
16	-1.146778 -13.565780 19.680424 7.891329	1.579812 -2.434520 4.046254 -4.170479	2.726590 11.131259 15.634170 12.061808
17	-16.737822 -1.591508 -29.389484 -24.933199	1.441897 -2.543725 3.783143 -4.219176	18.179718 0.952217 33.172627 20.714023
18	26.133331 9.119206 25.176403 6.423753	1.550757 -2.462136 3.994041 -4.187522	24.582574 11.581342 21.182362 10.611275
19	-6.713075 -17.319435 16.950933 7.010313	1.461175 -2.532527 3.822760 -4.218356	8.174250 14.786908 13.128174 11.228669
20	-17.080303 1.676469 -36.001755 -29.711794	1.532315 -2.478828 3.960313 -4.197109	18.612618 4.155297 39.962068 25.514685
21	32.931431 9.561033 36.142292 13.090668	1.474076 -2.524332 3.848781 -4.216774	31.457354 12.085365 32.293511 17.307441
22	-14.369573 -21.599304 11.564364 4.603891	1.520530 -2.489085 3.938476 -4.202632	15.890103 19.110219 7.625888 8.806523
23	-15.826289 6.234149 -42.224018 -34.525120	1.482644 -2.518563 3.865835 -4.215242	17.308933 8.752712 46.089854 30.309877
24	40.243252 9.080708 49.697006 21.552269	1.512962 -2.495473 3.924314 -4.205887	38.730290 11.576180 45.772692 25.758156
25	-24.467194 -26.267950 2.664234 0.114519	1.488305 -2.514600 3.876996 -4.214008	25.955498 23.753350 1.212761 4.328527
26	-12.271664 12.312920 -47.260345 -38.932289	1.508082 -2.499496 3.915117 -4.207844	13.779746 14.812415 51.175462
34.724445			
27	47.636131 7.247241 65.884979 31.927691	1.492029 -2.511921 3.884290 -4.213090	46.144102 9.759161 62.000689 36.140781
28	-37.303940 -31.081343 -10.715571 -7.106294	1.504928 -2.502051 3.909139 -4.209042	38.808868 28.579292 14.624710 2.897252
29	-5.567180 20.118221 -50.017887 -42.303669	1.494474 -2.510128 3.889055 -4.212438	7.061654 22.628349 53.906942 38.091231
30	54.446651 3.536233 84.522003 44.215366	- - - - - - - -	- - - -
31	-53.070187 -35.655270 -29.612917 -17.780970	- - - - - - - -	- - - -
32	5.274745 29.797224 -49.056522 -43.782734	- - - - - - - -	- - - -
33	59.724323 -2.668254 105.095230 58.230843	- - - - - - - -	- - - -
34	-71.775192 -39.429737 -55.075336 -32.675400	- - - - - - - -	- - - -
35	21.357138 41.394489 -42.546909 -42.250885	- - - - - - - -	- - - -
36	62.173862 -12.062399 126.642281 73.529648	- - - - - - - -	- - - -
37	-93.152649 -41.631702 -88.065155 -52.548538	- - - - - - - -	- - - -
38	43.852886 54.794586 -28.243971 -36.298901	- - - - - - - -	- - - -
39	60.100647 -25.397736 147.610626 89.315254	- - - - - - - -	- - - -
40	-116.543732 -41.238754 -129.323746 -78.076401	- - - - - - - -	- - - -
41	73.928978 69.650536 -3.486269 -24.211201	- - - - - - - -	- - - -
42	51.366104 -43.438038 165.701080 104.333252	- - - - - - - -	- - - -
43	-140.757248 -36.947956 -179.186600 -109.744835	- - - - - - - -	- - - -
44	112.632042 85.297173 34.764988 -3.971433	- - - - - - - -	- - - -
45	33.362457 -66.892792 177.701279 116.754501	- - - - - - - -	- - - -
46	-163.908157 -27.155096 -237.341797 -147.704880	- - - - - - - -	- - - -
47	160.726486 100.650322 89.831261 26.700813	- - - - - - - -	- - - -
48	3.018957 -96.321495 179.320343 124.052597	- - - - - - - -	- - - -
49	-183.239075 -9.952162 -302.525146 -191.585342	- - - - - - - -	- - - -
50	218.475601 114.093788 165.145569 70.268631	- - - - - - - -	- - - -

## **5. Conclusões**

Como observamos nas tabelas apresentadas na seção anterior o método G-S converge e o método de G-J não para o sistema linear estudado, pelo que concluimos que o método de G-S é o mais conveniente para resolver o sistema.

Em quanto a complexidade algorítmica e tempo de execução dos métodos é similar e a memória requerida para os cálculos também, por isso não achamos vantagem nesse sentido em nenhum dos dois métodos.