# UFSC – Universidade Federal de Santa Catarina

Nome: Leonardo Vailatti Eichstaedt
Matrícula: 14100847

Cálculo Numérico para Computação
Profº Sergio Peters

# Trabalho 1

a) Ao analisar o sistema, não podemos afirmar que o sistema possui convergência garantida, pois segundo o Critério de Scarborough, para convergir o sistema deveria possuir diagonal dominante.

O sistema em questão não possui diagonal dominante. Para verificarmos basta observar que para um sistema possuir diagonal dominante devem ocorrer duas condições:

1º) $| a_{ii} | \geq S_i$ , i = 1 ,..., n **e**
2º) $| a_{ii} | > S_i$ , para pelo menos uma linha i de A.

Obs : $S_i \rightarrow$ soma dos coeficientes fora da diagonal principal.

A primeira condição é satisfeita já que em todas as linhas $| a_{ii} | = S_i$, porém a segunda condição nunca é satisfeita pois em nenhuma linha $| a_{ii} | > S_i$.

Como o sistema de equações lineares não satisfaz os critérios de convergência, o processo iterativo poderá oscilar ou até mesmo divergir. Neste caso é recomendado usar fatores de sub ou sobre-relaxação em cada equação.

b) **Valores iniciais:**

Chute da valores inicias como 0s:

```
for k=1:100
        xi(k) = 0;
end
```

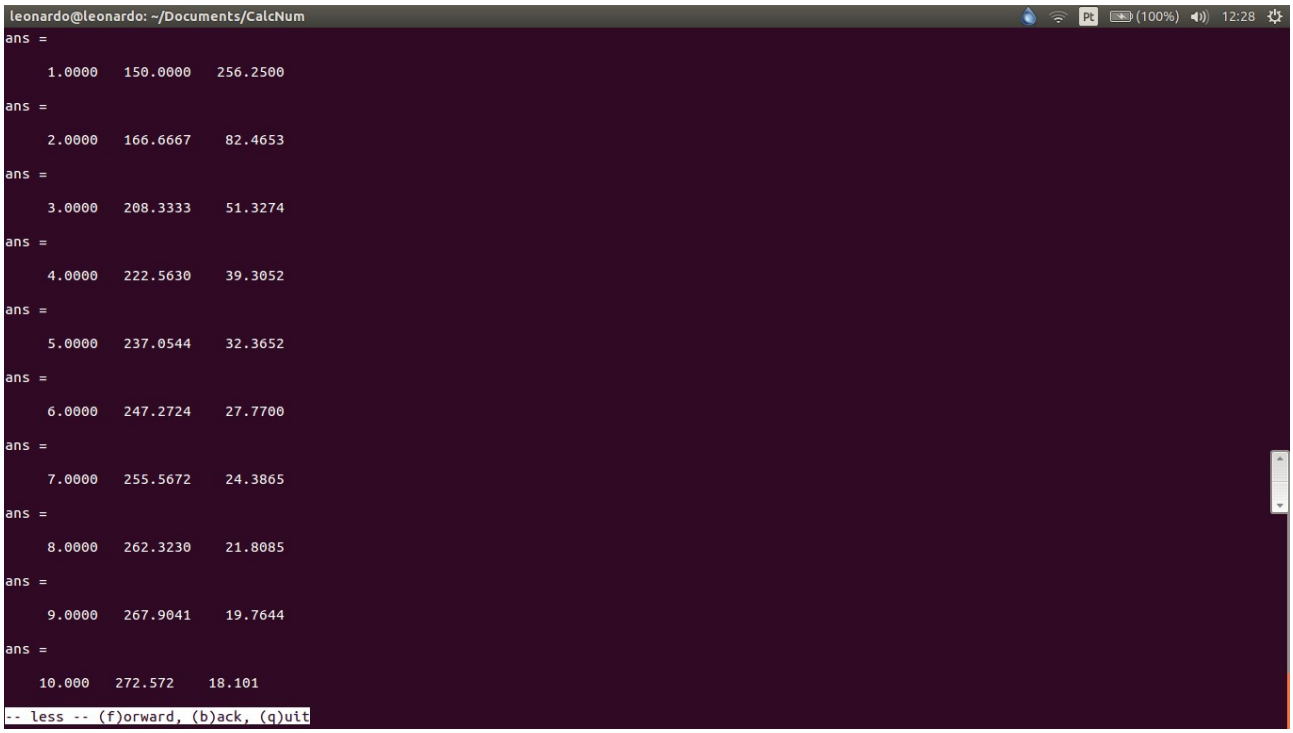
Limite de Iterações = 2000 (o uso de limite como 1000 não foi suficiente para o programa parar pelo critério)

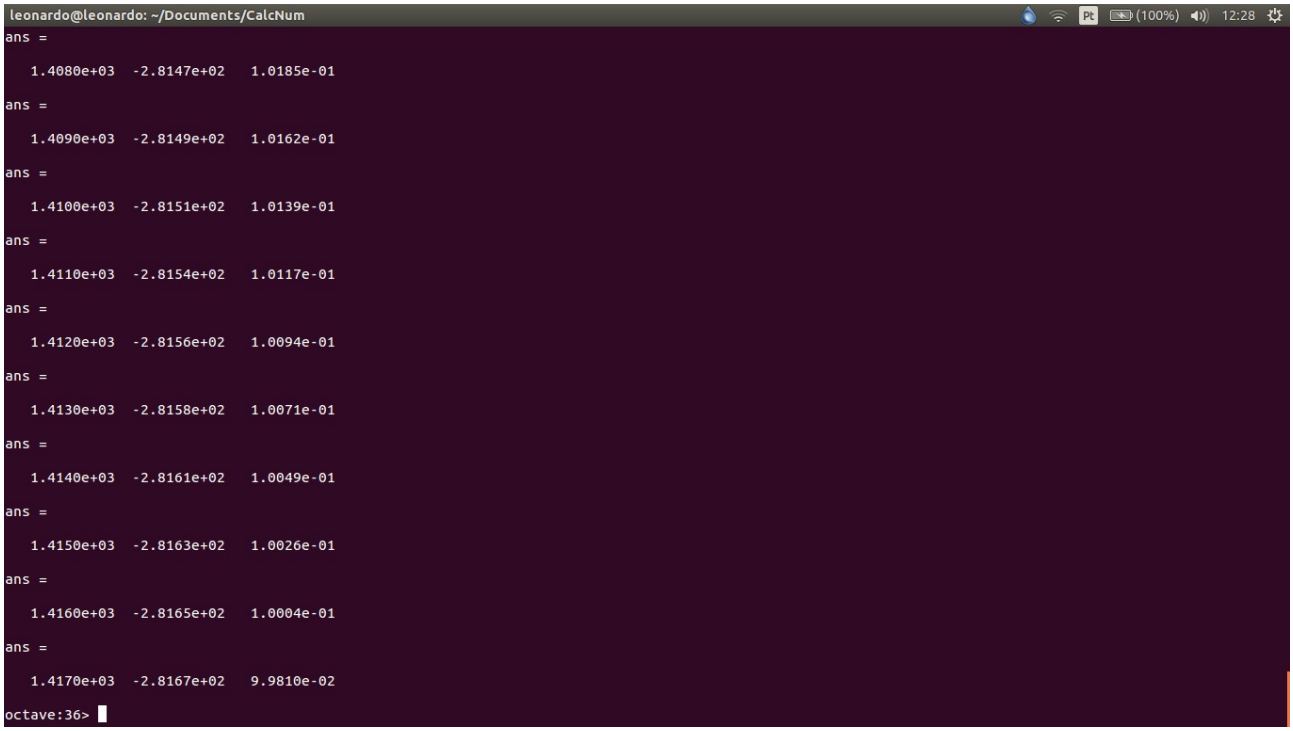
```
limite = 2000;
```

Critério de Parada = 1.e-1 (erro máximo de uma casa decimal. O enunciado pede uso de critério de parada maiores, para efetuar menos iterações.)

```
criterio = 1.e-1;
```

Sem o uso de fatores de relaxação foram necessários muitas iterações para que o valor tivesse convergência.




Obs: Primeira coluna representa o número do passo, a segunda coluna representa x(1) e a terceira coluna representa o módulo da diferença entre x e xi

Para acelerar o processo de convergência é recomendado o uso de fatores de relaxação.

Nossas equações de formação do sistema ficam de tal forma:

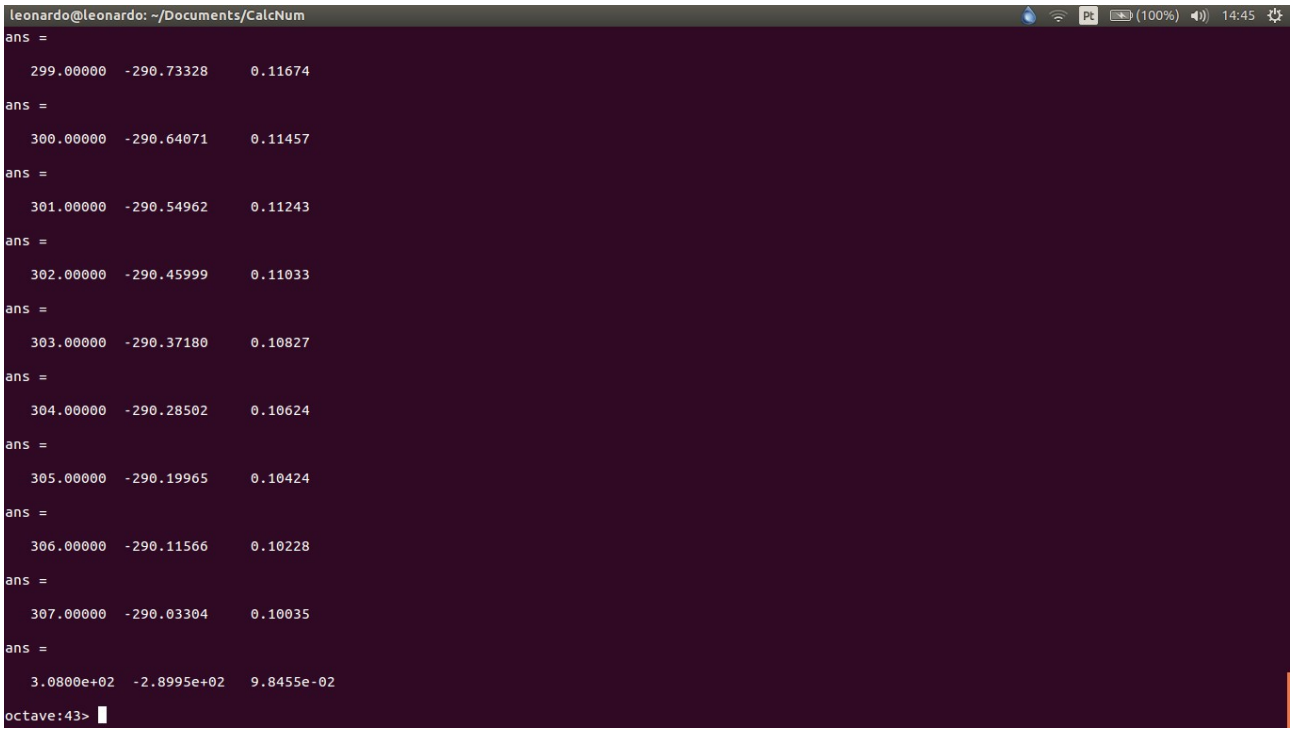
```
i=1;
x(i)= (1 - lambda)*x(i) + lambda*(150-x(i+1));
for i=2:n/2
        x(i)=(1 - lambda)*x(i) + lambda*(100-x(i-1)-x(i+1)-x(i+50))/3;
end
for i=(n/2)+1:n-1
        x(i)=(1 - lambda)*x(i) + lambda*(200-x(i-50)-x(i-1)-x(i+1))/3;
end
i=n;
x(i)=((1 - lambda)*x(i) + lambda*(300-x(i-1)));
```

$\lambda$ ( $0 < \lambda < 2$ )

Para encontrar o fator(lambda) que nós dá o menor número de equações realizamos testes sucessivos, alterando o valor da váriavel até encontrarmos o melhor valor.

| Lambda | Nº Iterações |
|---|---|
| 0.5 | 2000+ |
| 1.5 | 660 |
| 1.6 | 540 |
| 1.7 | 423 |
| **1.8** | **308** |
| 1.9 | 815 |

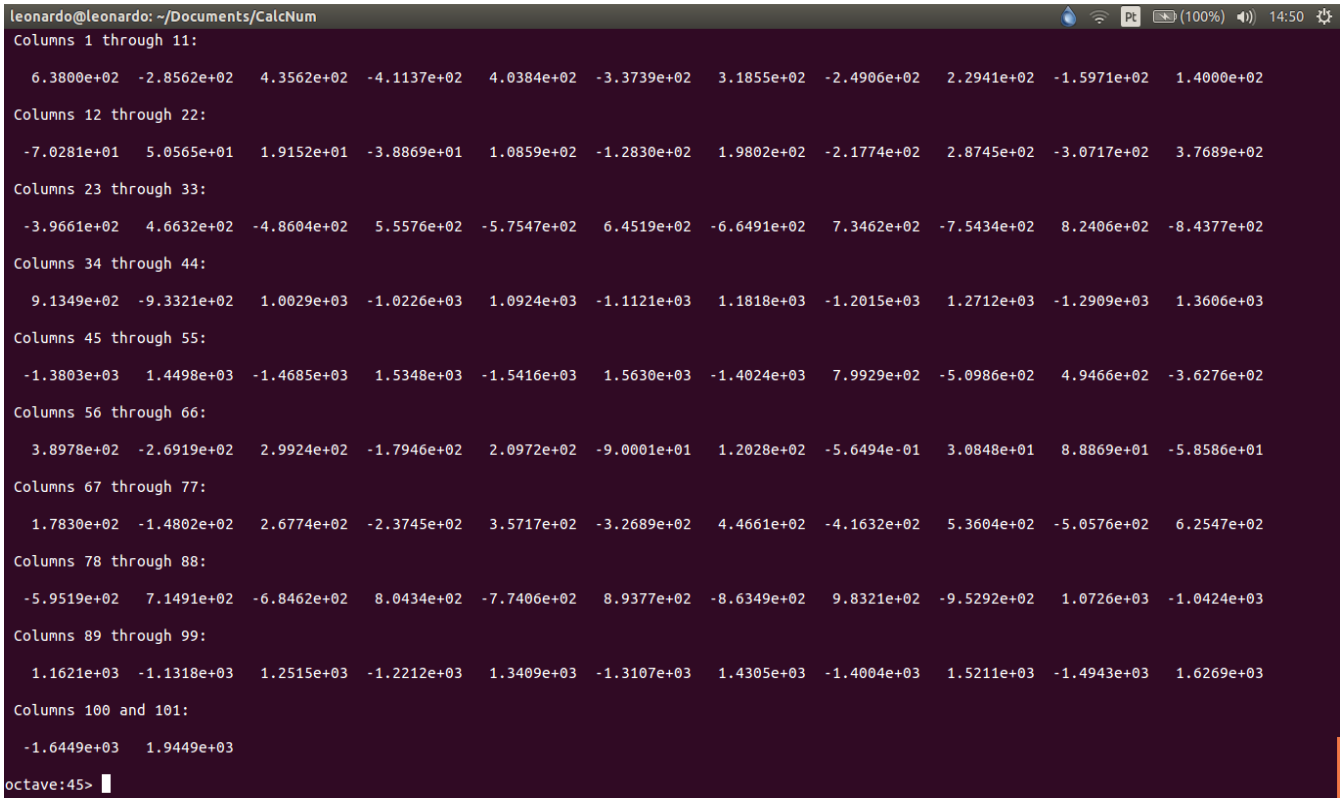
Portanto, é recomendado o uso de um fator de sobre-relaxação($1 \leq \lambda < 2$) para que seja obtida a convergência com o menor número de iterações.




Iterações com fator lambda = 1.8

c) Utilizando o método de diagonalização iterativa de Gauss-Seidel com fator de relaxação e critério de parada Max|Δx i |<=1.10 -4, a solução é calculada em 638 iterações.

A solução do sistema está na imagem abaixo, a partir da segunda coluna.



Para calcular o número de operações em ponto flutuante devemos colocar um contador que acrescente, em cada iteração, a quantidade de operações com ponto flutuante foram realizadas. Esse contador pode ser visto no arquivo “**SistemaC.m”**.

Após todas as iterações temos que realizamos: **570372** operações em ponto flutuante.

```
op =  570372
```

d) O erro **relativo** de Arredondamento obtido foi: **0.023412**

Para chegar nesse resultado foi usado um programa escrito em Java (**SistemaD.java**). A função do programa é realizar as iterações para a variável do tipo float e para o tipo double.

Depois basta fazer a operação:

$$\text{Erro Relativo } \% = \left|\frac{\overline{VA} - VE}{VE}\right| .100\%$$

e) O erro **absoluto** de Truncamento máximo na solução S obtida acima, em variavel ‘double’, é **0.00543368057810767,** como observado na imagem abaixo:

```
erroTruncamento =  0.00543368057810767
```

Esse erro foi calculado utilizando o mesmo algoritmo da questão **c)** . Primeiramente com Max|Δx i |<=1.10 -4, armazenando o valor de x para esse caso.

```
passo = 0;
limite = 2000;
dif = 1;
for k=1:100
        xi(k) = 0;
end
x = xi;
n=100;
criterio = 1.e-4;
lambda = 1.8;

while (passo < limite && dif > criterio )

        passo++;

        i=1;
        x(i)= (1 - lambda)*xi(i) + lambda*(150-x(i+1));
        for i=2:n/2
                x(i)=(1 - lambda)*xi(i) + lambda*(100-x(i-1)-x(i+1)-x(i+50))/3;
        end
        for i=(n/2)+1:n-1
                x(i)=(1 - lambda)*xi(i) + lambda*(200-x(i-50)-x(i-1)-x(i+1))/3;
        end
        i=n;
        x(i)=((1 - lambda)*xi(i) + lambda*(300-x(i-1)));

        dif = max(abs(x - xi));

        [passo, x(1)]


        xi = x;


end
```

Depois utilizamos o mesmo algoritmo mas com  Max|Δx i |<=1.10 -8 e armazenamos o valor de xn.

```
limite = 2000;
criterio = 1.e-8;
dif = 1;
passo = 0;
xn = x;

while (passo < limite && dif > criterio )

        passo++;

        i=1;
        xn(i)= (1 - lambda)*xi(i) + lambda*(150-xn(i+1));
        for i=2:n/2
                xn(i)=(1 - lambda)*xi(i) + lambda*(100-xn(i-1)-xn(i+1)-xn(i+50))/3;
        end
        for i=(n/2)+1:n-1
                xn(i)=(1 - lambda)*xi(i) + lambda*(200-xn(i-50)-xn(i-1)-xn(i+1))/3;
        end
        i=n;
        xn(i)=((1 - lambda)*xi(i) + lambda*(300-xn(i-1)));

        dif = max(abs(xn - xi));

        [passo, xn(1)]


        xi = xn;


end
```

Por fim, realizamos a operação para calcular o erro de truncamento propriamente dito:

```
erroTruncamento = max(abs(x - xn))
```

Os nomes dos programas correspondem à questão em que eles são utilizados:

b) - SistemaB
c) – SistemaC
d) – SistemaD
e) - SistemaE