Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SME0104 – Cálculo Numérico

Trabalho 1 – Método Iterativo de Gauss-Seidel para Resolução de Sistemas de Equações Lineares

Elias Italiano Rodrigues – 7987251

São Carlos, 20 de maio de 2014

1 Introdução

1.1 Métodos Iterativos

Muitos sistemas lineares são demasiadamente grandes para serem resolvidos por métodos diretos. Se n for grande, a matriz A requer muita memória para ser armazenada num computador.

Por outro lado, métodos iterativos são mais rápidos e geralmente são aplicados a sistemas lineares grandes, principalmente onde A é uma matriz esparsa.

Um método é iterativo quando fornece uma sequência de aproximantes $\mathbf{x}^{(k+1)}$ para uma solução $\mathbf{x}$, cada qual obtido do anterior $\mathbf{x}^{(k)}$ pela aplicação de um mesmo processo.

1.2 O Método de Gauss-Seidel

Para a obtenção de um método iterativo de um sistema linear $A\mathbf{x} = \mathbf{b}$, decomponhe-se a matriz A na forma $A = M + N$, onde M é uma matriz não-singular e diagonal, triangular ou tri-diagonal. Dessa forma, obtém-se:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ (M+N)\mathbf{x} &= \mathbf{b} \\ M\mathbf{x} + N\mathbf{x} &= \mathbf{b} \\ M\mathbf{x} &= \mathbf{b} - N\mathbf{x} \\ \mathbf{x} &= M^{-1}\mathbf{b} - M^{-1}N\mathbf{x} \end{aligned}$$

E o método iterativo dá-se por:

$$\mathbf{x}^{(k+1)} = M^{-1}\mathbf{b} - M^{-1}N\mathbf{x}^{(k)} \text{ para } k = 0, 1, 2, \ldots$$

No método de Jacobi, escolhe-se M de modo que seja matriz diagonal e N a matriz com os demais termos. Assim, tem-se:

$$\begin{cases} x_1^{(k+1)} &= \left[b_1 - (a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \cdots + a_{1n}x_n^{(k)})\right]/a_{11} \\ x_2^{(k+1)} &= \left[b_2 - (a_{21}x_2^{(k)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)})\right]/a_{22} \\ \vdots &= \vdots \\ x_n^{(k+1)} &= \left[b_n - (a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{nn-1}x_{n-1}^{(k)})\right]/a_{nn} \end{cases}$$

donde explicitando $x_i^{(k+1)}$, vem:

$$x_i^{(k+1)} = \left[b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right]/a_{ii} \text{ para } i = 1, 2, \ldots, n \text{ e } k = 0, 1, 2, \ldots$$

Observando-se que no método de Jacobi:

- para calcular a componente $x_i^{(k+1)}$ utiliza-se os valores $x_i^{(k)}$

- como a sequência $\{x_i^{(k+1)}\}$ é convergente, então os valores $x_j^{(k+1)}, j = 1, 2, \ldots, i-1$ constituem uma melhor aproximação para o x_i do que os valores $x_j^{(k)}, j = 1, 2, \ldots, i-1$

Com base nisso, define-se um método mais otimizado por:

$$x_i^{(k+1)} = \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right] / a_{ii} \text{ para } i = 1, 2, \ldots, n \text{ e } k = 0, 1, 2, \ldots$$

Esse método é conhecido como Método de Gauss-Seidel.

1.3 O Trabalho

Este trabalho consiste em implementar um programa de computador que execute o método iterativo de Gauss-Seidel para a resolução de sistemas de equações lineares. De acordo com o enunciado do trabalho, os seguintes valores são esperados como entrada para o programa:

- inteiro $n \geq 0$
- vetor $\mathbf{b}$ de valores constantes reais e tamanho n
- matriz $A = (a_{ij}) \in M_{n \times n}(\mathbb{R})$ tal que

$$\begin{cases} a_{i,i} & = & 4 & , i = 1, 2, \ldots, n \\ a_{i,i+1} & = & -1 & , i = 1, 2, \ldots, n-1 \\ a_{i+1,i} & = & -1 & , i = 1, 2, \ldots, n-1 \\ a_{i,i+3} & = & -1 & , i = 1, 2, \ldots, n-3 \\ a_{i+3,i} & = & -1 & , i = 1, 2, \ldots, n-3 \\ a_{i,j} & = & 0 & , \text{para os restantes} \end{cases}$$

  que já satisfaz as condições para o método de Gauss-Seidel, pois trata-se de uma matriz simétrica e definida positiva (SPD).

- constante real ε como majorante superior do erro
- inteiro $itmax \geq 1$ que limita a quantidade máxima de iterações para a execução.

São esperados como saída do programa o vetor solução $\mathbf{x}$ de tamanho n e a quantidade de iterações utilizadas até a convergência.

Seja o vetor resíduo $\mathbf{r} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$, a convergência ocorre quando a $\|\mathbf{r}\|_\infty \leq \varepsilon$ ou quando o número máximo de iterações $itmax$ é atingido.

2 Desenvolvimento

2.1 Implementação

O programa foi desenvolvido em linguagem `C` em sistema operacional `Linux`. Além das bibliotecas convencionais `stdlib.h` e `stdio.h`, foi utilizada também a `math.h` para usar a função `fabs()` que retorna o valor absoluto de um número real do tipo `double`.

Para representar os números reais no programa foi utilizado o tipo de dado de ponto flutuante `double` que é próprio da linguagem `C`.

O código do programa consiste nas seguintes funções:

- `int main(int argc, char **argv)`
  Função principal do programa que faz a leitura das entradas, impressão dos resultados e é responsável por chamar as demais funções.

- `int gauss_seidel(double **A, double *x, double *b, int n, double epsilon, int itmax)`
  Faz a implementação da fórmula do método iterativo de Gauss-Seidel usando os parâmetros de entrada que representam um sistema de equações lineares do tipo $A\mathbf{x} = \mathbf{b}$ de tamanho $n \times n$. Retorna o vetor solução na variável `x` e a quantidade de iterações utilizadas até a convergência no retorno padrão.

- `double norma_infinita(double *v, int n)`
  Calcula e retorna a norma infinita de um vetor `v` de tamanho `n` que é dada por

$$\|\mathbf{v}\|_\infty = \max_{1 \leq i \leq n} \{|v_i|\}$$

2.2 Código-fonte

main.c

```c
/**
 * Elias Italiano Rodrigues, 7987251
 *
 * SME0104 Calculo Numerico: Trabalho 1
 * 2014/05/20
 */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

/**
 * Calcula e retorna a norma infinita de um vetor v de tamanho n
 */
double norma_infinita(double *v, int n)
{
    int i;
    double abs, max = fabs(v[0]);

    for (i = 1; i < n; i++) {
        abs = fabs(v[i]);
        if (abs > max)
            max = abs;
    }
    return max;
}
```

```c
/**
 * Resolve o sistema linear do tipo Ax = b de tamanho n x n
 * e precisao epsilon pelo metodo de Gauss-Seidel com no maximo itmax iteracoes
 */
int gauss_seidel(double **A, double *x, double *b, int n, double epsilon, int itmax)
{
    double *x_ = (double *)malloc(sizeof(double) * n); // vetor iteracao (k+1)
    double *r = (double *)malloc(sizeof(double) * n); // vetor residuo x(k+1) - x(k)
    double soma; // valor dos somatorios da formula
    int iteracoes = 0; // quantidade de iteracoes ate convergir
    int i, j; // variaveis de iteracao

    // Retorna valor dos somatorios da formula
    double somatorios()
    {
        soma = 0.0;
        for (j = 0; j < i; j++)
            soma += A[i][j] * x_[j];
        for (j = i + 1; j < n; j++)
            soma += A[i][j] * x[j];
        return soma;
    }

    do {
        // Conta e confere iteracoes
        iteracoes++;
        if (iteracoes > itmax) {
            iteracoes--;
            break;
        }

        // Aplica formula do metodo de Gauss
        for (i = 0; i < n; i++)
            x_[i] = (b[i] - somatorios()) / A[i][i];

        // Calcula vetor residuo r
        for (j = 0; j < n; j++)
            r[j] = x_[j] - x[j];

        // Copia valores de x_ em x para a proxima iteracao
        for (j = 0; j < n; j++)
            x[j] = x_[j];

    } while (norma_infinita(r, n) > epsilon);

    free(x_);
    free(r);
    return iteracoes;
}
```

```c
/**
 * Programa principal
 */
int main(int argc, char **argv)
{
    int n;          // tamanho n do vetor b e matriz quadrada A
    double **A;     // matriz A
    double *x;      // vetor solucao x
    double *b;      // vetor b
    double epsilon; // epsilon da precisao para o erro
    int itmax;      // numero maximo de iteracoes
    int iteracoes; // quantidade de iteracoes para convergir
    int i, j, sinal; // variaveis auxiliares

    printf("SME0104 Calculo Numerico\nTrabalho 1: Metodo de Gauss-Seidel\n\n");

    // Leitura das entradas
    printf("n: "); scanf("%d", &n);
    printf("itmax: "); scanf("%d", &itmax);
    printf("epsilon: "); scanf("%lf", &epsilon);
    printf("\nvetor b (informe os n valores):\n");
    b = (double *)malloc(sizeof(double) * n);
    for (i = 0; i < n; i++) {
        printf("b[%d]: ", i + 1); scanf("%lf", b+i);
    }

    // Preparando o vetor x e construindo a matriz A segundo o enunciado do trabalho
    x = (double *)calloc(n, sizeof(double));
    A = (double **)malloc(sizeof(double *) * n);
    for (i = 0; i < n; i++)
        A[i] = (double *)calloc(n, sizeof(double));
    for (i = 0; i < n - 3; i++) {
        A[i][i] = 4.0;
        A[i][i+1] = A[i+1][i] = -1.0;
        A[i][i+3] = A[i+3][i] = -1.0;
    }
    for ( ; i < n - 1; i++) {
        A[i][i] = 4.0;
        A[i][i+1] = A[i+1][i] = -1.0;
    }
    for ( ; i < n; i++)
        A[i][i] = 4.0;

    printf("\nSistema a ser resolvido:\n\n");
    sinal = n / 2;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            printf(" %9lf ", A[i][j]);
        if (i == sinal)
            printf(" | x[%d] =  %9lf\n", i + 1, b[i]);
        else
            printf(" | x[%d]    %9lf\n", i + 1, b[i]);
    }
```

```
    printf("\nResolvendo o sistema...\n");
    iteracoes = gauss_seidel(A, x, b, n, epsilon, itmax);

    printf("\nQuantidade de iteracoes: %d\n", iteracoes);

    printf("\nVetor solucao do sistema:\n");
    printf("x = (");
    for (i = 0; i < n; i++)
        printf(" %9lf", x[i]);
    printf(" )\n");

    free(b);
    free(x);
    while (--n >= 0)
        free(A[n]);
    free(A);

    return EXIT_SUCCESS;
}
```

2.3 Compilação e Execução

Para compilar o código-fonte foi utilizado o `gcc` (GNU project C and C++ compiler):
`gcc -o t1_gauss-seidel main.c -lm`

A execução do programa, se este estiver no diretório atual, dá-se por:
`./t1_gauss-seidel`

O dados de entrada são pedidos na tela inicial do programa.

3 Resultados

Executando o programa para os casos de testes descritos nos itens do trabalho e tomando $\varepsilon = 0.000001$, $itmax = 1000$ e $\mathbf{x}^{(0)} = \mathbf{0}$, foram obtidas as seguintes soluções:

- Caso de teste com $b_i = \sum_{j=1}^{n} a_{ij}$ para $i = 1, 2, \ldots, n$ com solução esperada $x_i = 1$ para $i = 1, 2, \ldots, n$

  - para $n = 50$

```
Quantidade de iteracoes:  557
Vetor solucao do sistema:
x = ( 0.999993 0.999990 0.999988 0.999984 0.999980 0.999977 0.999974 0.999971
0.999968 0.999965 0.999963 0.999960 0.999958 0.999956 0.999954 0.999952
0.999951 0.999949 0.999948 0.999947 0.999946 0.999946 0.999945 0.999945
0.999945 0.999945 0.999946 0.999946 0.999947 0.999948 0.999949 0.999951
0.999952 0.999954 0.999956 0.999957 0.999960 0.999962 0.999964 0.999967
0.999969 0.999972 0.999974 0.999977 0.999980 0.999983 0.999986 0.999989
0.999992 0.999994 )
```

- para $n = 100$

```
Quantidade de iteracoes:  1000
Vetor solucao do sistema:
x = ( 0.999317 0.998988 0.998715 0.998294 0.997934 0.997598 0.997234 0.996881
0.996539 0.996194 0.995855 0.995521 0.995192 0.994867 0.994549 0.994236
0.993929 0.993628 0.993334 0.993046 0.992766 0.992493 0.992228 0.991971
0.991721 0.991480 0.991248 0.991024 0.990809 0.990603 0.990406 0.990219
0.990041 0.989873 0.989715 0.989567 0.989429 0.989301 0.989184 0.989077
0.988980 0.988894 0.988819 0.988754 0.988700 0.988657 0.988625 0.988603
0.988593 0.988593 0.988603 0.988625 0.988657 0.988700 0.988753 0.988817
0.988891 0.988976 0.989071 0.989176 0.989291 0.989416 0.989551 0.989695
0.989849 0.990012 0.990184 0.990365 0.990554 0.990753 0.990959 0.991174
0.991397 0.991627 0.991865 0.992110 0.992362 0.992621 0.992887 0.993158
0.993435 0.993719 0.994007 0.994301 0.994599 0.994902 0.995209 0.995520
0.995835 0.996153 0.996474 0.996800 0.997122 0.997452 0.997792 0.998104
0.998438 0.998825 0.999076 0.999378 )
```

Analisando os valores numéricos obtidos, conclui-se que o programa converge para a resposta esperada $x_i = 1$ para $i = 1, 2, \ldots, n$

- Caso de teste com $b_i = 1/i$ para $i = 1, 2, \ldots, n$

  - para $n = 40$

```
Quantidade de iteracoes:  389
Vetor solucao do sistema:
x = ( 0.748938 0.852286 0.911644 1.143467 1.248563 1.317490 1.414722 1.481012
1.528366 1.574063 1.607399 1.629788 1.645767 1.653826 1.654288 1.648471
1.636424 1.618402 1.594907 1.566175 1.532434 1.493949 1.450937 1.403592
1.352066 1.296556 1.237267 1.174168 1.107492 1.037716 0.964136 0.887048
0.808637 0.725354 0.636676 0.554129 0.462555 0.351604 0.280870 0.192106
)
```

4 Referências

Anotações pessoais de aula da disciplina SME0104 Cálculo Numérico ministrada pela professor Murilo Francisco Tomé no 1º semestre de 2014.