线性方程组求解

问题描述

给定如下的线性方程组 ($n = 120$):

$$\begin{gathered} 2x_1 - x_2 = 1 \\ -x_{j-1} + 2x_j - x_{j+1} = j, \quad j = 2, \ldots, n-1 \\ -x_{n-1} + 2x_n = n \end{gathered}$$

要求：
a) 如采用追赶法、高斯-塞德尔迭代法求解上述方程，迭代是否收敛性?
b) 给出系数矩阵的谱半径和条件数。(不能调用 matlab 自带的 `cond` 函数)
c) 用追赶法、高斯-塞德尔迭代法、和 SOR 迭代法求解向量 x。

注：不能采用 matlab 自身所带的代数方程求解函数，也不能调用 matlab 自带的矩阵分解函数。

分析与解答

a) 追赶法与高斯-塞德尔迭代法的收敛性

追赶法:
追赶法是一种专门用于求解三对角线性方程组的直接方法。对于本问题中的系数矩阵 A：

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

该矩阵是**对称正定**的。对于对称正定或严格对角占优的三对角矩阵，追赶法是数值稳定且高效的，能够直接得到方程组的精确解（在不考虑计算机舍入误差的情况下）。因此，追赶法不存在迭代收敛性的问题，它总能给出解。

高斯-塞德尔迭代法:
高斯-塞德尔迭代法的收敛性取决于其迭代矩阵 $B_{GS} = (D-L)^{-1}U$ 的谱半径 $\rho(B_{GS})$ 是否小于1。

其中 D 是 A 的对角部分，L 是 A 的严格下三角部分的相反数，U 是 A 的严格上三角部分的相反数。
对于本题中的系数矩阵 A：

1. 它是**严格对角占优**的（对于 $i = 2, \dots, n-1$，有 $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$），或者至少是**弱对角占优且不可约**的。具体来说，第一行和最后一行是弱对角占优，中间行是严格对角占优。
2. 更重要的是，矩阵 A 是**对称正定** 的。
   对于对称正定矩阵，高斯-塞德尔迭代法**保证收敛**。

从提供的运行结果来看：

```
高斯-塞德尔法求解完成，迭代次数：33774
```

这表明，高斯-塞德尔法能够收敛到预设的 10^{-8} 容差，但其收敛速度较慢，需要 33774 次迭代。

b) 系数矩阵的谱半径和条件数

系数矩阵 A 是一个 $n \times n$ 的三对角矩阵，主对角线元素为 2，次对角线元素为 -1。其特征值 λ_k 可以由解析式给出：

$$\lambda_k = 2 - 2\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, 2, \dots, n$$

由于 $0 < \frac{k\pi}{n+1} < \pi$，所以 $\cos\left(\frac{k\pi}{n+1}\right)$ 的值域在 $(-1, 1)$ 之间，因此所有特征值 λ_k 都是正的。

谱半径 $\rho(A)$:
谱半径定义为 $\rho(A) = \max_k |\lambda_k|$。由于所有特征值均为正，所以谱半径等于最大的特征值。

$$\rho(A) = \lambda_{\max} = 2 - 2\cos\left(\frac{n\pi}{n+1}\right)$$

当 $k = n$ 时，$\cos$ 项最小（最接近-1），因此 λ_k 最大。

条件数 $cond_2(A)$:
对于对称正定矩阵，其 2-范数条件数为：

$$cond_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$$

最小的特征值为：

$$\lambda_{\min} = 2 - 2\cos\left(\frac{\pi}{n+1}\right)$$

当 $k=1$ 时，$\cos$ 项最大（最接近1），因此 λ_k 最小。

根据源代码中的计算和运行结果：

- **系数矩阵 A 的谱半径: 3.999326**
- **系数矩阵 A 的条件数 (2-范数): 5933.107143**

条件数 5933.107143 远大于 1，表明该矩阵是**病态的**。这意味着解对系数矩阵 A 或右端向量 b 的微小扰动可能非常敏感，并且迭代方法的收敛速度可能会较慢，这与高斯-塞德尔法的表现一致。

c) 用追赶法、高斯-塞德尔迭代法和 SOR 迭代法求解向量 x

1. 追赶法:
作为一种直接方法，追赶法求解结果精确且高效。
运行结果显示： `追赶法求解完成。`

2. 高斯-塞德尔迭代法:
如前所述，高斯-塞德尔法收敛但速度很慢。
运行结果显示： `高斯-塞德尔法求解完成，迭代次数: 33774` 。

3. SOR 迭代法:
SOR 法是高斯-塞德尔法的一种加速改进，通过引入松弛因子 ω。最优松弛因子 ω_{opt} 的计算公式为：

$$\omega_{opt} = \frac{2}{1+\sqrt{1-\rho(B_J)^2}}$$

其中 $\rho(B_J)$ 是雅可比迭代矩阵 $B_J = D^{-1}(L+U)$ 的谱半径。对于本问题的矩阵 A，$\rho(B_J) = \cos(\frac{\pi}{n+1})$。
根据运行结果：

```
计算得到的最优松弛因子 omega_opt: 1.949392
SOR法求解完成，迭代次数: 605
```

SOR 法使用计算得到的最优松弛因子，仅用 605 次迭代就达到了收敛容差，远优于高斯-塞德尔法。

解向量摘要与比较:

从运行结果中可以看到三种方法得到的解向量（部分元素）：

```
--- 解向量摘要 (前3个元素, 中间元素, 后3个元素) ---
索引 追赶法 高斯-塞德尔 SOR法
1 2440.000000 2440.000000 2440.000000
2 4879.000000 4878.999999 4879.000000
3 7316.000000 7315.999999 7316.000000
60 110410.000000 110409.999985 110410.000000
118 14101.000000 14100.999999 14101.000000
119 9520.000000 9519.999999 9520.000000
120 4820.000000 4820.000000 4820.000000
```

解向量之间的无穷范数差异：

```
--- 解向量之间的差异 (范数) ---
norm(x_thomas - x_gs, inf): 1.482182e-05
norm(x_thomas - x_sor, inf): 1.968438e-07
norm(x_gs - x_sor, inf): 1.468831e-05
```

分析：

- 追赶法作为直接法，其解可以视为“精确解”(在数值精度范围内)。
- 高斯-塞德尔法收敛到了一个较为精确的解，但仍不如SOR法的精度高。
- SOR 法的解与追赶法的解非常接近（范数为 1.968×10^{-7})，表明 SOR 法在最优松弛因子下能够高效地逼近精确解。

总结

1. **追赶法**对于本问题中的三对角系统是稳定且直接有效的。
2. **高斯-塞德尔法**理论上收敛，因为系数矩阵是对称正定的。它在 33774 次迭代后达到了预设精度。然而，由于矩阵的条件数较大 ($n = 120$ 时约为 5933)，其收敛速度非常缓慢。
3. **SOR 法**通过使用最优松弛因子（约为 1.949)，显著加快了收敛速度，仅用 605 次迭代就达到了所需精度，其解与追赶法的精确解非常吻合，并且优于在更多迭代次数下高斯-塞德尔法得到的解。
4. 系数矩阵的谱半径约为 3.999，条件数约为 5933.1，表明矩阵是病态的，这解释了迭代法（尤其是未经加速的高斯-塞德尔法）收敛缓慢的原因。

代码实现

```
clear;
clc;

% --- 问题参数 ---
n = 120; % 矩阵维度

% --- 构建系数矩阵 A 和右端向量 b_vec ---
% A 是一个三对角矩阵
main_diag = 2 * ones(n, 1);
sub_diag = -1 * ones(n-1, 1);
super_diag = -1 * ones(n-1, 1);

A = diag(main_diag) + diag(super_diag, 1) + diag(sub_diag, -1);

% 构建 b_vec
b_vec = zeros(n, 1);
b_vec(1) = 1;
for j = 2:n-1
    b_vec(j) = j;
end
b_vec(n) = n;

fprintf('--- 问题 b) --- \n');
% 计算谱半径和条件数 (使用解析特征值)
% 对于 [-1, 2, -1] 类型的三对角矩阵，特征值为 lambda_k = 2 - 2*cos(k*pi/(n+1))
% 这里我们的矩阵对角线是2，次对角线是-1，所以特征值是 2 - 2*cos(k*pi/(n+1))
% k = 1, ..., n
k_vals = (1:n)';
cos_terms = cos(k_vals * pi / (n + 1));
eigenvalues_A = 2 - 2 * cos_terms;

lambda_min_A = min(eigenvalues_A); % 对应 k=1 时，cos 最大
lambda_max_A = max(eigenvalues_A); % 对应 k=n 时，cos 最小 (最负)

% 谱半径 rho(A) = max(|lambda_i|)
% 由于所有特征值 lambda_k = 2 - 2*cos(k*pi/(n+1)) > 0 (因为 cos < 1)
% 所以 rho(A) = lambda_max_A
spectral_radius_A = lambda_max_A;
fprintf('系数矩阵 A 的谱半径: %f\n', spectral_radius_A);
```

```matlab
% 条件数 cond_2(A) = lambda_max(A) / lambda_min(A) (因为 A 是对称正定矩阵)
condition_number_A2 = lambda_max_A / lambda_min_A;
fprintf('系数矩阵 A 的条件数 (2-范数): %f\n', condition_number_A2);
fprintf('\n');

fprintf('--- 问题 c) --- \n');
% 迭代法参数
tol = 1e-8;        % 收敛容差
max_iter = 200000;   % 最大迭代次数
x0 = zeros(n, 1); % 初始解向量

% --- c.1) 追赶法  ---
fprintf('求解中 (追赶法)... \n');
x_thomas = thomas_algorithm(diag(A,-1), diag(A), diag(A,1), b_vec);
fprintf('追赶法求解完成。\n');
% disp('追赶法解向量 x_thomas (部分):');
% disp_solution_summary(x_thomas);

% --- c.2) 高斯-塞德尔迭代法  ---
% 矩阵 A 是对称且严格(或弱)对角占优的，因此高斯-塞德尔法收敛。
fprintf('求解中 (高斯-塞德尔法)... \n');
[x_gs, iter_gs] = gauss_seidel(A, b_vec, x0, tol, max_iter);
fprintf('高斯-塞德尔法求解完成，迭代次数: %d\n', iter_gs);
% disp('高斯-塞德尔法解向量 x_gs (部分):');
% disp_solution_summary(x_gs);

% --- c.3) SOR 迭代法 ---
% 计算最优 omega
% rho_BJ 是Jacobi迭代矩阵的谱半径。对于该类型矩阵, rho_BJ = cos(pi/(n+1))
rho_BJ = cos(pi / (n + 1));
omega_opt = 2 / (1 + sqrt(1 - rho_BJ^2)); % omega_opt = 2 / (1 + sin(pi/(n+1)))
fprintf('计算得到的最优松弛因子 omega_opt: %f\n', omega_opt);

fprintf('求解中 (SOR法)... \n');
[x_sor, iter_sor] = sor_method(A, b_vec, omega_opt, x0, tol, max_iter);
fprintf('SOR法求解完成，迭代次数: %d\n', iter_sor);


fprintf('\n--- 解向量摘要 (前3个元素, 中间元素, 后3个元素) ---\n');
fprintf('%15s %15s %15s %15s\n', '索引', '追赶法', '高斯-塞德尔', 'SOR法');
indices_to_show = [1; 2; 3; round(n/2); n-2; n-1; n];
for idx = indices_to_show'
    fprintf('%15d %15.6f %15.6f %15.6f\n', idx, x_thomas(idx), x_gs(idx), x_sor(idx));
```

```
end

fprintf('\n--- 解向量之间的差异 (范数) ---\n');
fprintf('norm(x_thomas - x_gs, inf): %e\n', norm(x_thomas - x_gs, inf));
fprintf('norm(x_thomas - x_sor, inf): %e\n', norm(x_thomas - x_sor, inf));
fprintf('norm(x_gs - x_sor, inf): %e\n', norm(x_gs - x_sor, inf));


% --- 函数定义 ---

% 追赶法
% a: 次对角线 (长度 n-1), A(i,i-1) -> a(i-1) for A's i=2..n
% b: 主对角线 (长度 n)
% c: 超对角线 (长度 n-1), A(i,i+1) -> c(i) for A's i=1..n-1
% d: 右端向量 (长度 n)
function x = thomas_algorithm(a_sub, b_main, c_super, d_rhs)
    N = length(d_rhs);
    cp = zeros(N-1, 1); % 存储修改后的超对角线元素
    dp = zeros(N, 1);   % 存储修改后的右端向量元素
    x = zeros(N, 1);    % 解向量

    % 向前消元
    cp(1) = c_super(1) / b_main(1);
    dp(1) = d_rhs(1) / b_main(1);

    for k = 2:N-1
        denom = b_main(k) - a_sub(k-1) * cp(k-1);
        cp(k) = c_super(k) / denom;
        dp(k) = (d_rhs(k) - a_sub(k-1) * dp(k-1)) / denom;
    end

    % 处理 dp 的最后一个元素
    denom_N = b_main(N) - a_sub(N-1) * cp(N-1);
    dp(N) = (d_rhs(N) - a_sub(N-1) * dp(N-1)) / denom_N;

    % 回代求解
    x(N) = dp(N);
    for k = N-1:-1:1
        x(k) = dp(k) - cp(k) * x(k+1);
    end
end

% 高斯-塞德尔迭代法
```

```matlab
function [x, iter] = gauss_seidel(A, b, x0, tol, max_iter)
    N = length(b);
    x = x0;
    D = diag(diag(A));
    L = -tril(A, -1);
    U = -triu(A, 1);

    % 迭代格式: x_new = inv(D-L) * (U*x_old + b)
    % 或者逐元素更新
    for iter = 1:max_iter
        x_old = x;
        for i = 1:N
            sigma1 = 0;
            if i > 1
                sigma1 = A(i, 1:i-1) * x(1:i-1); % 使用本轮已更新的 x
            end
            sigma2 = 0;
            if i < N
                sigma2 = A(i, i+1:N) * x_old(i+1:N); % 使用上一轮的 x_old
            end
            x(i) = (b(i) - sigma1 - sigma2) / A(i,i);
        end

        if norm(x - x_old, inf) < tol
            return;
        end
    end
    warning('高斯-塞德尔法在达到最大迭代次数 %d 后未收敛到容差 %e', max_iter, tol);
end

% SOR 迭代法
function [x, iter] = sor_method(A, b, omega, x0, tol, max_iter)
    N = length(b);
    x = x0;

    for iter = 1:max_iter
        x_old = x;
        for i = 1:N
            sigma1 = 0;
            if i > 1
                sigma1 = A(i, 1:i-1) * x(1:i-1); % 使用本轮已更新的 x
            end
            sigma2 = 0;
```

```
        if i < N
            sigma2 = A(i, i+1:N) * x_old(i+1:N); % 使用上一轮的 x_old
        end

        % 计算当前分量的高斯-塞德尔迭代值
        x_gs_i = (b(i) - sigma1 - sigma2) / A(i,i);

        % 应用SOR公式
        x(i) = (1 - omega) * x_old(i) + omega * x_gs_i;
    end

    if norm(x - x_old, inf) < tol
        return;
    end
end
warning('SOR法在达到最大迭代次数 %d 后未收敛到容差 %e', max_iter, tol);
end
```