

标准雅可比迭代算法程序

常清俊

2017年11月8日

Contents

- 预处理检查用户调用函数时参数格式是否传错
- 对角线
- 开始迭代
- 结果

```
function [ output_args ] = cqj_Jacobi(init_X,coefficient_mat,b_mat,maxNum,epsilon)
```

```
% 雅可比迭代法
```

```
% init_X: 初始值
```

```
% coefficient_mat: 系数矩阵
```

```
% b_mat: b
```

```
% maxNum: 最大迭代次数
```

```
% epsilon: 精度
```

```
output_args = [];
```

```
flag = 0;
```

预处理检查用户调用函数时参数格式是否传错

```
if size(coefficient_mat,1) ~= size(coefficient_mat,2)
```

```
    disp('系数矩阵格式错误(需要方阵)');
```

```
    return;
```

```
end
```

```
if size(b_mat,1) ~= 1 && size(b_mat,2) ~= 1
```

```

        disp('AX=b中的b必须为行向量或者列向量');
        return;
    end
    if size(b_mat,1) == 1
        b_mat = b_mat';
    end
    if size(init_X,1) ~= 1 && size(init_X,2) ~= 1
        disp('初始值必须为行向量或者列向量');
        return;
    end
    if size(init_X,1) == 1
        init_X = init_X';
    end
    if size(init_X,1) ~= size(coefficient_mat,1)
        disp('初始值与系数矩阵维数不一致');
        return;
    end
    if size(init_X,1) ~= size(b_mat,1)
        disp('AX=b中的b向量维数错误(需与系数矩阵的维数一致)');
        return;
    end
    if ~exist('maxNum', 'var')
        maxNum = 500; % 默认值
    end
    if ~exist('epsilon', 'var')
        epsilon = 1e-14; % 默认值
    end
end

```

对角线

```

D = diag(diag(coefficient_mat));
% 可能matlab的源码inv函数对角函数求逆会像普通函数一样处理,
% 这种特殊函数求逆会比较快
D_inv = diag(1./diag(coefficient_mat));

```

开始迭代

下面代码段也行但是对内存的消耗比较大 $x(:,0) = \text{init_X}$; for $k = 1:\text{maxNum}$ $x(:,k) = D.\text{inv}*(b.\text{mat}-(\text{coefficient_mat}-D)*x(:,k-1))$; if $\text{norm}(x(:,k)-x(:,k-1)) < \text{epsilon}$ % 达到精度 $\text{flag} = 1$; break; end end

```
x = init_X;
```

迭代公式:

$$x_{k+1} = D^{-1}(b - (L + U)x_k), k = 0, 1, 2, \dots$$

```
for k = 1:maxNum
    temp = D_inv*(b_mat-(coefficient_mat-D)*x);
    if norm(temp-x) < epsilon
        flag = 1;
        x = temp;
        break;
    end
    x = temp;
end
```

结果

```
text(0,0.95,'方程: ','Color','red','FontSize',14);
str1 = '\left[\begin{array}';
% 正则表达式
str2 = @(X) regexprep(regexprep(mat2str(zeros(1,size(X,2))),...
    '\s\[\]' , '' ), '0', 'c');
mat_start = @(X) [str1 '{' str2(X) '}'];
my_mat2str = @(mat) regexprep(regexprep(regexprep(mat2str(mat),...
    '*','&'),' ','\\\''), '\[\]' , '' );
mat_end = '\end{array}\right]';
str5 = '\times X=';
% LaTeX 输出
```

```

text('Interpreter','latex','String', ['$\$'...
    mat_start(coefficent_mat),my_mat2str(coefficent_mat),mat_end,...
    str5,mat_start(b_mat),my_mat2str(b_mat),mat_end '$$'],...
    'Position',[0.1 0.7],'FontSize',10);
text(0,0.5,'标准雅可比迭代结果: ','Color','red','FontSize',14);
if flag == 1 %好的结果
    output_args.itrNum = k;
    output_args.result = x;
    fprintf('迭代次数: %d\n',output_args.itrNum);
%    disp(x(:,k));
    disp('解: ');
    disp(x);
    text('Interpreter','latex','String', ['$X^*=' ...
        mat_start(x) my_mat2str(x) mat_end '$$'],...
        'Position',[0.1 0.3],'FontSize',10);
    text(0.1,0.1,['迭代次数: ' num2str(output_args.itrNum)],...
        'Color','k','FontSize',10);
    return;
end
text(0.1,0.3,['迭代' num2str(maxNum) '次后未收敛'],...
    'Color','k','FontSize',10);
fprintf('迭代%d次后暂未收敛\n',maxNum);

```

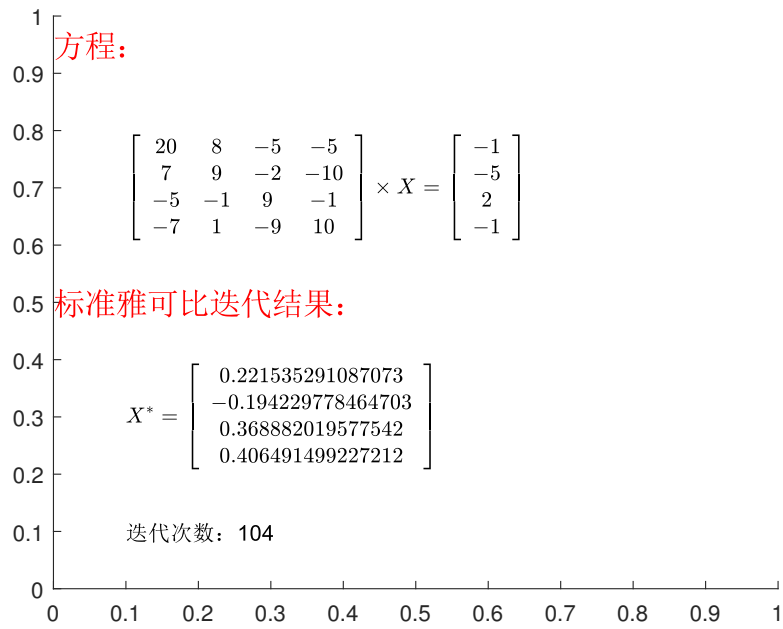
迭代次数: 104

解:

```

    0.2215
-0.1942
    0.3689
    0.4065

```



end