Санкт-Петербургский политехнический
университет им. Петра Великого
Институт прикладной математики и механики
Кафедра Прикладная математика

Лабораторная работа №5

Эмиссионная томография плазмы. Решение ИСЛАУ

по дисциплине: Вычислительные комплексы

Студент группы 3630102/60201 Митрофанова А.Г.

Преподаватель Баженов А.Н.

Санкт-Петербург
2020 год

Содержание

Список иллюстраций

1 Постановка задачи

Ставится задача решения ИСЛАУ

$$Ax = b \tag{1}$$

где A – матрица хорд, b – значения детектора.
Решить полученную ИСЛАУ (1) двумя способами:

- $x = (A^T A)^{-1} A^T b$
- Используя функцию *tosolvty*

2 Теория

Для построения ИСЛАУ правую часть (1) представим в виде интервала:

$$Ax = [\underline{b}, \overline{b}] \tag{2}$$

где $\underline{b}$ – минимум b в некотором окне радиуса K, $\overline{b}$ – максимум b в некотором окне радиуса K, где рассматриваются показатели детектора во временные интервалы с "текущий"$-K$ до "текущий"$+K$.

2.1 Функция *tolsolvty*

Функция

function [tolmax,argmax,envs,ccode] = tolsolvty(infA,supA,infb,supb,varargin)

выдаёт значение максимума распознающего функционала $tolmax$ множества для интервальной системы линейных уравнений (1), у которой матрицы нижних и верхних концов элементов A равны $infA$ и $supA$ соответственно, а векторы нижних и верхних концов правой части b равны $infb$ и $supb$ соответственно. Дополнительно процедура выводит аргумент $argmax$ – доставляющий $tolmax$ вектор значений аргумента, который лежит в допусковом множестве решений при $tolmax >= 0$.
Если $tolmax < 0$, то допусковое множество решений системы пусто. В таком случае ослабляем условия, расширяем интервал $[\underline{b}, \overline{b}]$ так, чтобы допусковое решение было не пусто $[\underline{b} - \Delta b, \overline{b} + \Delta b]$. Для получения решения достаточно взять $\Delta b = |tolmax|$.

2.2 Оценка вариабельности *IVE*

$$IVE(\mathbf{A}, \mathbf{b}) = \sqrt{n} \max_{\mathbb{R}^n} Tol \cdot (\min_{A \in \mathbf{A}} cond_2 A) \cdot \frac{||arg\max_{\mathbb{R}^n} Tol||_2}{||\mathbf{b}||_2} \tag{3}$$

3 Реализация

Реализация осуществлена на языке программирования MATLAB в среде разработке MATLAB Online R2019b.
Для считывания информации о сепаратрисе из g-файла была использована функция $gfile\_extractor\_1t$ [2].

4 Результаты

Рассматривается набор данных 37000, временной интервал 000162.
Число обусловленности матрицы A

$$cond(A) = 4.77 \cdot 10^{16}$$

Число обусловленности матрицы $A^T A$

$$cond(A^T A) = 2.79 \cdot 10^{35}$$

4.1 Простой способ решения

Матрица A сильно разрежена, поэтому собственные числа матрицы $A^T A$ сконцентрированы около нуля

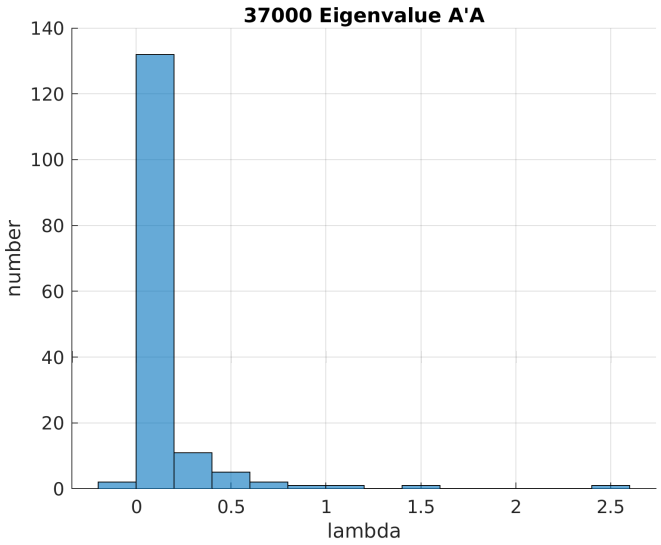

Рис. 1: Гистограмма собственных чисел матрицы $A^T A$

Всего 22 собственных числа больше 0.2.
Ввиду того, что число обусловленности очень большое, надежда найти обратную матрицу почти отсутствует. Поэтому в качестве решения *Matlab* получен вектор, состоящий из *NaN*.

4.2 Решение *tolsolvty*

Для нахождения интервала b выбрано окно радиуса $K = 1$.
При первой попытке решения с помощью функции ***tolsolvty*** получили $tolmax = -169.0752$. Т.к. $tolmax < 0$, то допусковое множество решений интервальной линейной системы пусто.

Рис. 2: График первой попытки решения с помощью функции ***tolsolvty***

Расширим b, выбрав $\Delta b = 169.0752$.
При второй попытке решения с помощью функции ***tolsolvty*** получили $tolmax = 0.0028$. Т.к. $tolmax >= 0$, то допусковое множество решений интервальной линейной системы непусто.

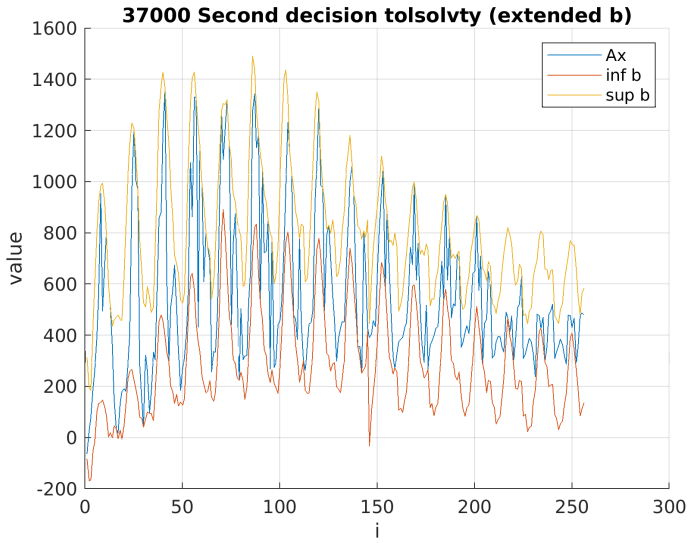

Рис. 3: График второй попытки решения с помощью функции ***tolsolvty***

Посмотрим на решение на графике:

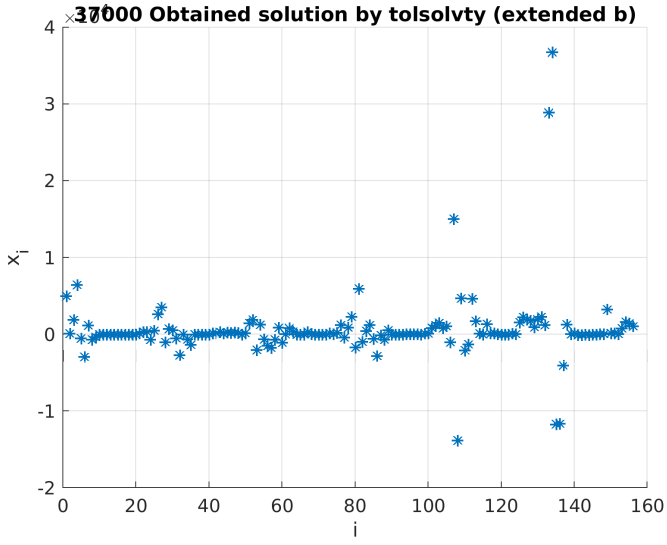

Рис. 4: График полученного решения от i с помощью функции ***tolsolvty***

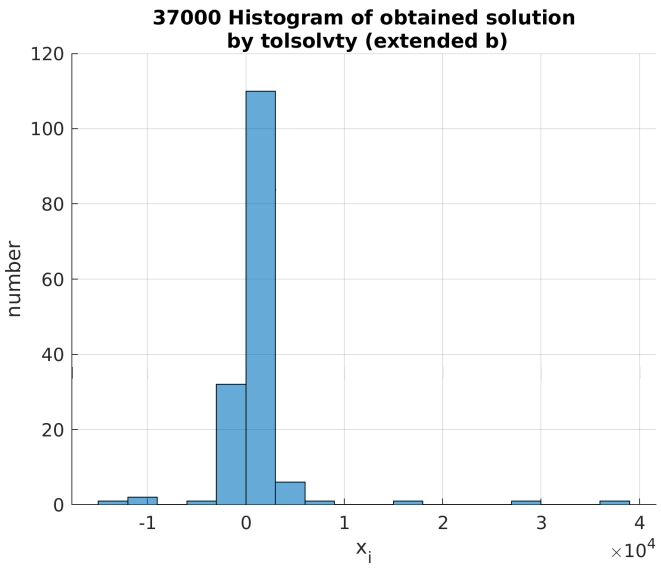

Рис. 5: Гистограмма решения, полученного с помощью функции ***tolsolvty***

4.3 Оценка числа обусловленности интервальной матрицы A

В качестве оценки радиуса элементов матрицы A возьмём 10% от их величины. Выбор такого значения обусловлен тем, что это точность знания сепаратрисы, по которой построена матрица A.
Рассмотрим значения оценки числа обусловленности для разного количества повторений при постоянном радиусе элементов 10%

```
fix radius, different number of iteration
rad = 0.1 HeurMinCond(A, 10) = 2.008224358626789e+16
rad = 0.1 HeurMinCond(A, 20) = 3.112295113402922e+16
rad = 0.1 HeurMinCond(A, 30) = 2.892131296414938e+16
rad = 0.1 HeurMinCond(A, 40) = 2.096838109302854e+16
rad = 0.1 HeurMinCond(A, 50) = 2.569437970419934e+16
rad = 0.1 HeurMinCond(A, 60) = 2.006277703217819e+16
rad = 0.1 HeurMinCond(A, 70) = 2.245900186140562e+16
rad = 0.1 HeurMinCond(A, 80) = 2.066637154114504e+16
rad = 0.1 HeurMinCond(A, 90) = 2.421265791587686e+16
rad = 0.1 HeurMinCond(A, 100) = 2.214231732078458e+16
```

Рис. 6: Значения числа обусловленности при изменении числа итераций

Рассмотрим значения оценки числа обусловленности для разных радиусов элементов A при постоянном числе итераций равным 100.

```
fix number of iteration, different radius
rad = 0.1 HeurMinCond(A, 100) = 2.68794841006769e+16
rad = 0.15 HeurMinCond(A, 100) = 2.665801131825306e+16
rad = 0.2 HeurMinCond(A, 100) = 2.661785974662393e+16
rad = 0.25 HeurMinCond(A, 100) = 2.38554481082195e+16
rad = 0.3 HeurMinCond(A, 100) = 2.945721174980176e+16
rad = 0.35 HeurMinCond(A, 100) = 2.359754161438948e+16
rad = 0.4 HeurMinCond(A, 100) = 2.617619681269532e+16
rad = 0.45 HeurMinCond(A, 100) = 2.046281938804106e+16
rad = 0.5 HeurMinCond(A, 100) = 2.1568321922126e+16
```

Рис. 7: Значения числа обусловленности при изменении радиуса элементов

Для полученного решения $maxtol = 0$, то и $IVE(A, b) = 0$.

5 Обсуждение

СЛАУ представляет собой матрицу $256xN$, где N – количество элементов разбиения. Матрица A крайне плохо обусловлена. Из-за этого матрица $A^T A$ становится еще хуже обусловлена. Поэтому, получить стандартное решение с помощью формулы нельзя. Вторым методом решение было получено, но при этом сильно ослаблены условия на b и не гарантируется, что $x >= 0$.

6 Литература

Список литературы

[1] А.Н. Баженов, Лабораторный практикум. Методический материал. «Вычислительные комплексы» [Электронный ресурс, облачное хранилище]. Режим доступа: https://cloud.mail.ru/public/4ra6/5wwqBzMBC/LabPractics.pdf (дата обращения: февраль, 2020 г.)

[2] Код функции, извлекающей информацию о сепаратрисе [Электронный ресурс, облачное хранилище]. Режим доступа: https://cloud.mail.ru/public/5o3T/4G4dD71hL (дата обращения: февраль, 2020 г.)

7 Приложение

7.1 main.m

```
%%
clear all
point_ = get_point_interface();
element_ = get_element_interface();
detector_ = get_detector_interface();
%%
[flux, RBDRY, ZBDRY, NBDRY, R, Z, time, rdim, zdim] = gfile_extractor_1t(34363, 000162, 65);
%%
% change the start of the crawl and change the detour of the separatrix to counter-hourly
% since the last point is equal to the first, let's take this into account
%a new beginning - the 32nd element
[RBDRY,ZBDRY] = circle_spin_and_reverse(RBDRY,ZBDRY,NBDRY, 32);

[arr, ind_arr] = min(flux);
```

```
[flux_min, min_j] = min (arr);
min_i = ind_arr(min_j);
magnet_axis = [R(min_j), Z(min_i)];
%%
num_sectors = 8;
num_circle = 6;
cur_cut_y = RBDRY;
cur_cut_z = ZBDRY;
cur_magnet_axis = magnet_axis;

[R_segments_arr, Z_segments_arr, lines_start, lines_end] = get_web_grid(cur_cut_y, cur_cut_z
    , cur_magnet_axis, num_sectors, num_circle);
elements = create_element_from_grid(R_segments_arr, Z_segments_arr, lines_start, lines_end);
%% calculate partition grids of the separatrix in the cross section plane x = H
% H - distance from the center of the tokamak to the section plane
min_x_sep = min(RBDRY);
max_x_sep = max(RBDRY);
H = 0;
h = length(elements);
cut_elements = [];
left_cut_elem = [];
right_cut_elem = [];
for i = 1 : h
    % the result of the intersection of the sector rotation shape and the x = H plane
    [elem, count] = element_.get_cut(elements(i), H);
    if(count == 2)
        left_cut_elem = [left_cut_elem, elem(1)];
        right_cut_elem = [right_cut_elem, elem(2)];
    else
        left_cut_elem = [left_cut_elem, elem];
    end
end

cut_elements = [left_cut_elem, right_cut_elem];
%% magic constants
% angle between the pinhole camera direction and the center direction (between 8 and 9 beams
    )
ang = acos((708^2 + 720^2 - 31^2) / (2 * 708 * 720));
% position of the detector edge (1st column)
spd_start = [0, -0.708];
% position of the 16th column
spd_end = [0.72 * sin(ang), 0.72 * -cos(ang)];
% direction vector of the pinhole camera in the Equatorial plane
spd_vect = (spd_end - spd_start) / norm(spd_end - spd_start);
% step between columns in the detector plane, 2 numbers
spd_xy_step = [2.3375 - 0.88 , 3.81 - 2.3375 + 0.88 ] * 1e-03;
% the center of the detector
pp = spd_start + spd_vect * ((spd_xy_step(1) + spd_xy_step(2)) * 8 + 0.52 * 1e-03) / 2;
% aperture offset from the center of the detector
aperture_xy_offset = 0.0395;
% coordinate of the aperture
aperture_xy = [pp(1) - spd_vect(2) * aperture_xy_offset, pp(2) + spd_vect(1) *
    aperture_xy_offset];
spd_z_start = (27.52 - 0.49) / 2 * 1e-03;
spd_z_step = -1.72 * 1e-03;
spd_xy = spd_start + spd_vect * (spd_xy_step(2) / 2 + 0.26 * 1e-03);
%% detectors parametres
detector = detector_.create();
detector.start = point_.create(spd_start(1), spd_start(2));
detector.end = point_.create(spd_end(1), spd_end(2));
detector.step = spd_xy_step;
detector.direction = point_.create(spd_vect(1), spd_vect(2));
detector.center = point_.create(pp(1), pp(2));
detector.aperture_offset = 0.0395;
detector.aperture_pos = point_.create(aperture_xy(1), aperture_xy(2));
detector.z_start = spd_z_start;
detector.z_step = spd_z_step;
%%
element_num = length(elements);
hord_matrix = zeros(256, element_num);
```

```
cur_row = 1;
for cut_ind = 1 : 16
    H = detector_.get_plane(detector, cut_ind);
    cut_elements = [];
    for i = 1 : element_num
        [elem, count] = element_.get_cut(elements(i), H);
        if(H < min_x_sep)
            if(count == 2)
                cut_elements = [cut_elements, elem(2)];
            else
                cut_elements = [cut_elements, elem];
            end
        else
            cut_elements = [cut_elements, elem];
        end
    end
    N = length(cut_elements);
    for ray_ind=1 : 16
        [k, b, det_pos, apper_pos] = detector_.get_ray(detector, 16, ray_ind);
        for t = 1 : N
            [hord, intersection] = element_.get_hord(cut_elements(t), k, b);
            hord_matrix(cur_row, cut_elements(t).index) = hord_matrix(cur_row, cut_elements(t).index) + hord;
        end
        cur_row = cur_row + 1;
    end
end

figure('Name', 'HORD MATRIX')
hold on
grid on
imagesc(hord_matrix);
title('37000 Hord Matrix');
colormap('Bone');
cm = colormap;
colormap(flipud(cm));
colorbar;
print('-dpng', '-r300', strcat('37000_hord_matrix', '.png')), pwd;
%%
input_file_name = strcat(num2str(37000), "_SPD16x16.mat");
input_data = load(input_file_name);
sign_bb = input_data.sign_bb(:, :, :);
cnt_meas = size(sign_bb, 3);
tp = cell2mat(input_data.Data(1, 2)) * 1e-3;
tz = cell2mat(input_data.Data(2, 2));
t_start = tz;
t_end = t_start + (cnt_meas - 1) * tp;
t_i = t_start : tp : t_end;
%% overview of integrated luminosity
t_cons_start = 125;
t_cons_end = 200;

dt_cons = 1;
start_efit_time_i = t_cons_start : dt_cons : t_cons_end;

B = [];
for start_efit_time = t_cons_start : t_cons_end
    ind = find(abs(t_i - start_efit_time) < tp / 2);
    b = [];
    for i = 16 : -1 : 1
        b = [b; sign_bb(16: -1 : 1, i, ind(1))];
    end
    b = double(b);
    Bnew = sum(b(:));
    B = [B, Bnew];
end
%% plot overview of integrated luminosity
figure('Name', 'LUMINOSITY OVERVIEW')
hold on
grid on
```

```
plot([t_cons_start : t_cons_end], B, 'LineWidth', 1);
title(['37000 SPD16x16.mat', ' Sum b']);
xlabel('start efit time');
print('-dpng', '-r300', strcat('37000_over_lum', '.png')), pwd;
%% select the "window" by which we calculate the boundaries of b
b_time_window = 1;
input_time_period = 000162;
b_data = [];
for start_efit_time = input_time_period - b_time_window : input_time_period + b_time_window
    ind = find(abs(t_i - start_efit_time) < tp / 2);
    b = [];
    for i = 16 : -1 : 1
        b = [b; sign_bb(16 : -1 : 1, i, ind(1))];
    end
    b = double(b);
    b_data = [b_data, b];
end
N = length(b_data);
inf_b = zeros(N, 1);
sup_b = zeros(N, 1);
for i = 1 : N
    inf_b(i) = min(b_data(i, :));
    sup_b(i) = max(b_data(i, :));
end
%%
b = (sup_b + inf_b) / 2;
A = hord_matrix;
%% simple decision
simple_dec = inv(A' * A) * A' * b;
lambda = eig(A' * A);
disp(strcat("cond(A) = ", num2str(cond(A))));
disp(strcat("cond(A'A) = ", num2str(cond(A' * A))));
disp(strcat("number of lambda > 0.2: ", num2str(length(find(lambda > 0.2)))));

figure('Name', "EIG A'A")
hold on
grid on
histogram(lambda)
xlabel("lambda")
ylabel("number")
title("37000 Eigenvalue A'A")
print('-dpng', '-r300', strcat('37000_lambda', '.png')), pwd;
%% tolsolvty first
b_sup = sup_b;
b_inf = inf_b;
A_inf = A;
A_sup = A;
[tolmax, argmax, envs, ccode] = tolsolvty(A_inf, A_sup, b_inf, b_sup);
disp(strcat("tolmax = ", num2str(tolmax)));

figure('Name', "FIRST DECISION tolsolvty")
hold on
grid on
x_axis = 1 : length(b_sup);
y_axis = A_sup * argmax;
plot(x_axis, y_axis');
plot(x_axis, b_inf');
plot(x_axis, b_sup');
xlabel("i")
ylabel("value")
title("37000 First decision tolsolvty")
legend("Ax", "inf b", "sup b")
print('-dpng', '-r300', strcat('37000_first_tolsolvty', '.png')), pwd;
%% tolsolvty second
delta_b = - tolmax;
disp(strcat("delta b = ", num2str(delta_b)));
b_sup = b_sup + delta_b;
b_inf = b_inf - delta_b;
[tolmax, argmax, envs, ccode] = tolsolvty(A_inf, A_sup, b_inf, b_sup);
disp(strcat("tolmax = ", num2str(tolmax)));
```

```
221
222 figure('Name', "SECOND DECISION tolsolvty")
223 hold on
224 grid on
225 x_axis = 1 : length(b_sup);
226 y_axis = A_sup * argmax;
227 plot(x_axis, y_axis');
228 plot(x_axis, b_inf');
229 plot(x_axis, b_sup');
230 xlabel("i")
231 ylabel("value")
232 title("37000 Second decision tolsolvty (extended b)")
233 legend("Ax", "inf b", "sup b")
234 print('-dpng', '-r300', strcat('37000_second_tolsolvty', '.png')), pwd;
235 %% plot decision
236 figure('Name', "OBTAINED SOLUTION tolsolvty")
237 hold on
238 grid on
239 plot(argmax, "*", 'LineWidth', 1)
240 xlabel("i")
241 ylabel("x_i")
242 title("37000 Obtained solution by tolsolvty (extended b)")
243 print('-dpng', '-r300', strcat('37000_obt_sol', '.png')), pwd;
244 %% plot hist decision
245 figure('Name', "HISTOGRAM of OBTAINED SOLUTION tolsolvty")
246 hold on
247 grid on
248 histogram(argmax)
249 xlabel("x_i")
250 ylabel("number")
251 title(["37000 Histogram of obtained solution", " by tolsolvty (extended b)"])
252 print('-dpng', '-r300', strcat('37000_hist_obt_sol', '.png')), pwd;
253 %% fix radius, different number of iteration
254 disp("fix radius, different number of iteration")
255 matrix_radius = 0.1;
256 A_inf_1 = A * (1 - matrix_radius);
257 A_sup_1 = A * (1 + matrix_radius);
258 b_inf_1 = inf_b;
259 b_sup_1 = sup_b;
260 for i = 10 : 10 : 100
261     cond_A = HeurMinCond(A_inf_1, A_sup_1, i);
262     disp(strcat("rad = ", num2str(matrix_radius), " HeurMinCond(A, ", num2str(i), ") = ",
    num2str(cond_A)));
263 end
264 %% fix number of iteration, different radius
265 disp("fix number of iteration, different radius")
266 for rad = 0.1 : 0.05 : 0.5
267     A_inf_1 = A * (1 - rad);
268     A_sup_1 = A * (1 + rad);
269     b_inf_1 = inf_b;
270     b_sup_1 = sup_b;
271     cond_A = HeurMinCond(A_inf_1, A_sup_1, i);
272     disp(strcat("rad = ", num2str(rad), " HeurMinCond(A, ", num2str(i), ") = ", num2str(
    cond_A)));
273 end
274 %% calculate IVE
275 A_inf_1 = A * 0.9;
276 A_sup_1 = A * 1.1;
277 b_inf_1 = inf_b;
278 b_sup_1 = sup_b;
279
280 cond_A = HeurMinCond(A_inf_1, A_sup_1);
281 [tolmax_1, argmax_1, envs_1, ccode_1] = tolsolvty(A_inf_1, A_sup_1, b_inf_1, b_sup_1);
282 if (tolmax_1 < 0)
283     delta_b_1 = - tolmax_1;
284     b_inf_1 = b_inf_1 - delta_b_1;
285     b_sup_1 = b_sup_1 + delta_b_1;
286     [tolmax_1, argmax_1, envs_1, ccode_1] = tolsolvty(A_inf_1, A_sup_1, b_inf_1, b_sup_1);
287 end
288
```

```
289 A_IVE_1 = IVE(A_inf_1, A_sup_1, b_inf_1, b_sup_1, tolmax_1, argmax_1, length(argmax_1));
```