数理アルゴリズムとシミュレーション　演習課題3　 提出日：2020年10月29日
201811319 永崎遼太

課題2

A

$$
\begin{pmatrix}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{pmatrix}
$$

b

$$
\begin{pmatrix}
u_{1,0} + u_{0,1} \\
u_{2,0} \\
u_{3,0} + u_{4,1} \\
u_{0,2} \\
0 \\
u_{4,2} \\
u_{0,3} + u_{1,4} \\
u_{2,4} \\
u_{4,3} + u_{3,4}
\end{pmatrix}
$$

課題3

（3-1）

次のようなコードを作成した（n = 4 の場合）。

ーーーーー

```
load n_4
n = 4;
x = zeros(n*n, 1); % 初期状態の設定
maxit = 2000; % 最大反復回数
tol = 1e-04; % 判定条件
r = b - A*x;
p = r;
t = 0;
t2 = 0;
for i = 1:maxit
    v = 0;
    Ap = A*p;
    alpha = dot(p,r)/dot(p,Ap);
    x = x + alpha*p;
    r = r - alpha*Ap;
    beta = -dot(r,Ap)/dot(p,Ap);
    p = r + beta*p;
    v = norm(r);
    t(i) = v;
    t2(i) = i;
    if v < tol
        break;
    end
```

```
end
t2
semilogy(t)
ylim([tol inf])
xlabel('Number of Iterations')
ylabel('2-Norm of r_k')
```

ーーーーー

tに各反復ごとのr_kの2ノルムの値を格納している。これを用いて片対数グラフをsemilogyにより描画すると図1のようになる。

同様にしてn=50, 200の場合のグラフを描画すると、図2、3のようになる。

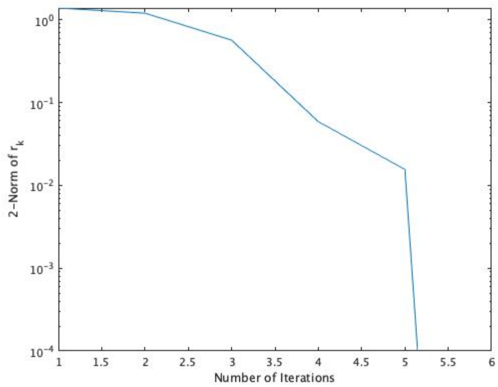

図1 n=4の場合の各反復ごとのr_kの2ノルム

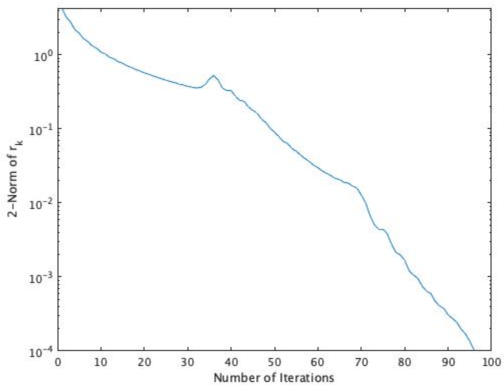

図2 n=50の場合の各反復ごとのr_kの2ノルム

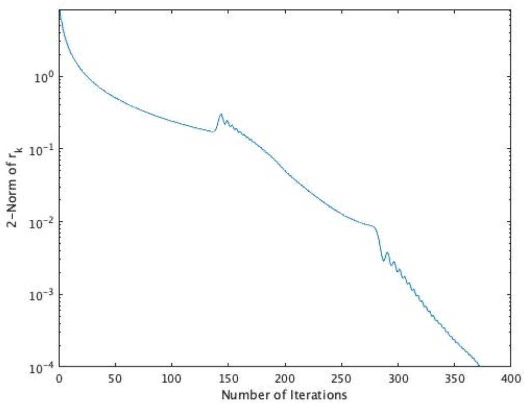

図3 n=200の場合の各反復ごとのr_kの2ノルム

なお、それぞれの場合についてplotを用いて描画したグラフはそれぞれ図4、5、6のようになる。

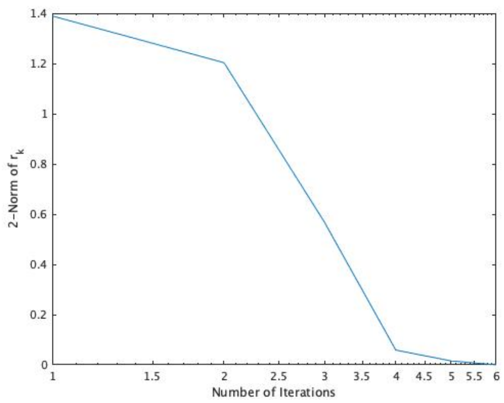

図4 n=4の場合のplotによるグラフ

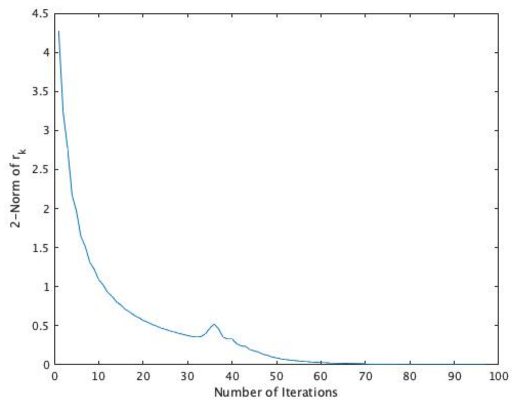

図5 n=50の場合のplotによるグラフ

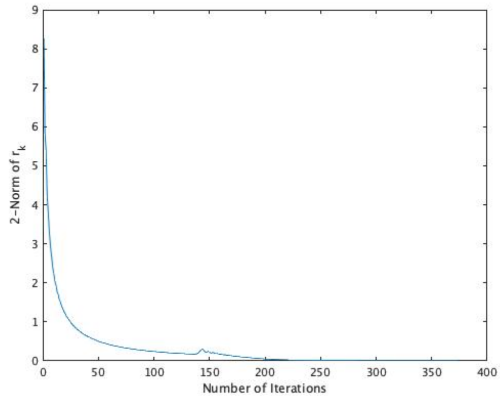

図6 n=200の場合のplotによるグラフ

図4、5、6の場合、反復回数が大きくなるにつれて収束の様子が見にくくなってしまうので、横軸を反復回数の数字としながら、より収束の様子が見やすくすることができるsemilogyによる片対数グラフが適切だと考えた。

（3-2）

次のようなコードを作成した（n = 4 の場合）。

ーーーーー

```
load n_4
n = 4;
x = zeros(n*n, 1); % 初期状態の設定
maxit = 2000; % 最大反復回数
tol = 1e-04; % 判定条件
r = b - A*x;
p = r;
for i = 1:maxit
    v = 0;
    Ap = A*p;
    alpha = dot(p,r)/dot(p,Ap);
    x = x + alpha*p;
    r = r - alpha*Ap;
    beta = -dot(r,Ap)/dot(p,Ap);
    p = r + beta*p;
    v = norm(r);
    if v < tol
        break;
    end
end
X = reshape(x,[n,n]);
u = zeros(n+2,n+2);
% 境界条件の設定
u(:,1) = linspace(0,1,n+2);
u(:,n+2) = linspace(1,0,n+2);
u(1,:) = linspace(0,1,n+2);
```

```
u(n+2,:) = linspace(1,0,n+2);
for i = 2:n+1
   for j = 2:n+1
      u(i,j) = X(i-1, j-1);
   end
end
x = linspace(0, 1, n+2);
y = linspace(0, 1, n+2);
surf(x, y, u);
```

ーーーーー

これを実行すると、図7、8のようなグラフが描画される。同様にして、同様にしてn=50, 200の場合のグラフを描画すると、図9、10のようになる。

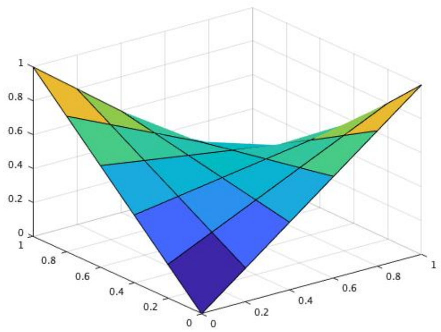

図7 n=4の場合のsurfによるグラフ(1)

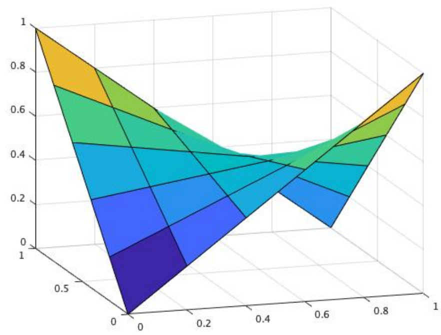

図8 n=4の場合のsurfによるグラフ(2)

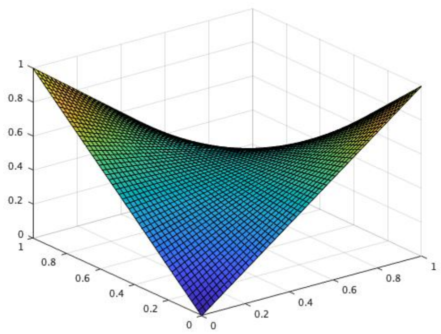

図9 n=50の場合のsurfによるグラフ

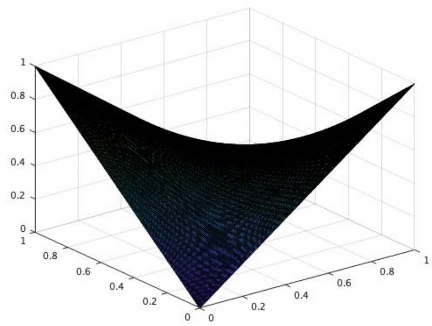

図10 n=50の場合のsurfによるグラフ

図11、12は前回の課題(演習課題2 の課題1) の図である。

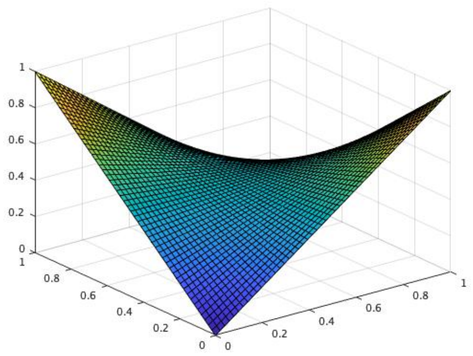

図11 前回の課題(演習課題2の課題1)の図(1)

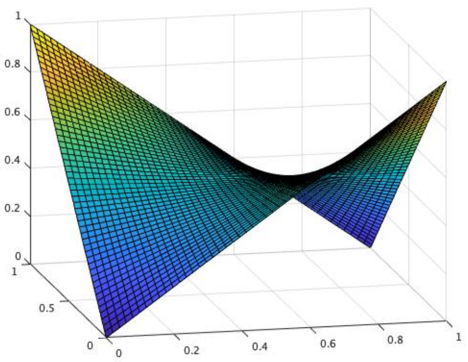

図12 前回の課題(演習課題2の課題1)の図(2)

図7〜10と図11、12を比較して、(3-1) で求めた解xをn次正方行列に変換し，その値をMATLAB の関数surf を用いてグラフに描き, 前回の課題(演習課題2 の課題1) の図1と同じ結果になることが確認できた。

課題4

次のようなコードを作成した。

―――――

```
for n = 10:100
    A = rand(n);
    A = (A + A')/2;
    A = A + 5*eye(n);
    b = rand(n,1);
    x = zeros(n, 1); % 初期状態の設定

    maxit = 2000; % 最大反復回数
    tol = 1e-08; % 判定条件

    r = b - A*x;
    p = r;

    for i = 1:maxit
        v = 0;
        Ap = A*p;
        alpha = dot(p,r)/dot(p,Ap);
        x = x + alpha*p;
        r = r - alpha*Ap;
        beta = -dot(r,Ap)/dot(p,Ap);
        p = r + beta*p;
        v = norm(r);
        t(i) = v;
        t2 = i;
        if v < tol
            break;
        end
    end
    c(n) = t2;
end

plot(c);
xlim([10 100]);
xlabel('Matrix Size')
ylabel('Number of Iterations')
```

―――――

これを実行すると、図13が描画された。

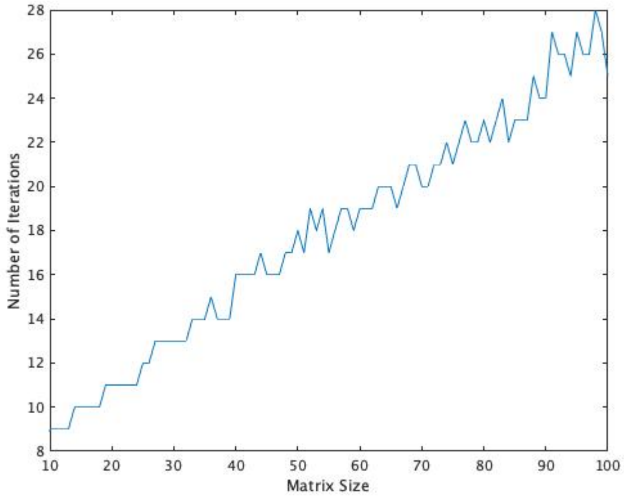

図13 行列サイズごとの反復回数

細かな増減はあるものの、行列サイズが大きくなるほど反復回数が大きくなっていくことがわかる。