＝＝

8 月 7 日（金）第 14 回数値解析Ｉ 提出課題 19TM054 浅野 駿介

提出日：2020/08/17

＜作成プログラム＞

```
#include<stdio.h>
#include<math.h>
#define N 4

//行列を出力する関数
void print_matrixA(double A[][N + 1]) {

	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N + 1; j++) {
			printf("%lf ", A[i][j]);
		}
		printf(" ¥n");
	}
	printf(" ¥n");
}

void print_matrixX2(double X[]) {
	for (int i = 0; i < N; i++) {
		printf("%lf ", X[i]);
	}
	printf(" ¥n");
}

double check_err(double newX[], double X[]) {

	double max_err = 0.0;

	for (int i = 0; i < N; i++) {
		if (fabs(newX[i] - X[i]) > max_err)
			max_err = fabs(newX[i] - X[i]);
	}
	return(max_err);
```

```
}

void my_jacobi(double A[][N + 1], double X[], int kmax, double err) {
	double newX[N];

	int i, j;
	int k = 0;

	for (k = 0; k < kmax; k++) {
		printf("%d-->", k);//何回目かの計算か
		print_matrixX2(X);//k 回目の結果を出力

		for (i = 0; i < N; i++) {

			newX[i] = A[i][N] / A[i][i];

			for (j = 0; j < N; j++) {

				//i と j の大小で場合分け
				if (j < i)
					newX[i] = newX[i] - A[i][j] * X[j+1] / A[i][i];
				if (j > i)
					newX[i] = newX[i] - A[i][j] * X[j] / A[i][i];
			}
		}

		//許容誤差内で収まったかどうか判断する
		if (check_err(newX, X) < err)
			k = kmax;

		for (i = 0; i < N; i++) {
			X[i] = newX[i];
		}
	}
}
```

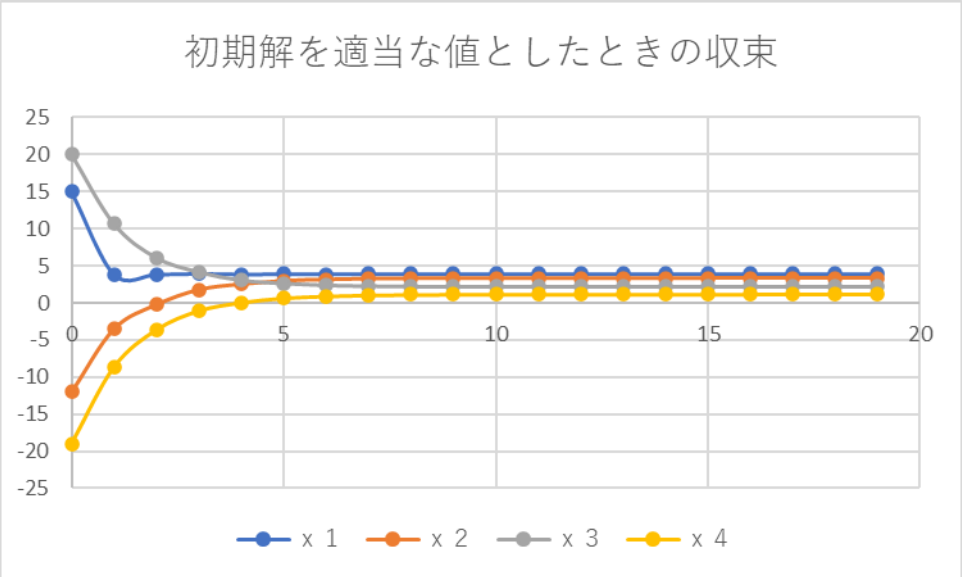
```
void main() {

	//行列の値
	double A[N][N + 1] = { {8.0,1.0,2.0,1.0,40.0},
	                       {2.0,4.0,1.0,-2.0,20.0},
	                       {2.0,0.0,6.0,1.0,21.0},
	                       {1.0,3.0,-2.0,8.0,17.0} };

	//初期解
	double X[N] = { 15,-12.0,20.0,-19.0 };

	//関数を呼び出す
	print_matrixA(A);
	my_jacobi(A, X, 100, 0.00001);
}
```

＜出力結果＞

初期解を適当な解としたとき
8.000000 1.000000 2.000000 1.000000 40.000000
2.000000 4.000000 1.000000 -2.000000 20.000000
2.000000 0.000000 6.000000 1.000000 21.000000
1.000000 3.000000 -2.000000 8.000000 17.000000

0-->15.000000 -12.000000 20.000000 -19.000000
1-->3.875000 -3.500000 10.666667 -8.625000
2-->3.848958 -0.229167 6.104167 -3.593750
3-->3.951823 1.791667 4.175347 -1.033854
4-->3.861437 2.543403 3.075087 0.076823
5-->3.903700 2.997938 2.639395 0.673123
6-->3.881269 3.177743 2.388500 0.928765
7-->3.889561 3.278386 2.285958 1.064286
8-->3.885677 3.321460 2.229824 1.124039
9-->3.886857 3.343833 2.205507 1.154643
10-->3.886314 3.354028 2.192948 1.168617
11-->3.886432 3.359057 2.187221 1.175545

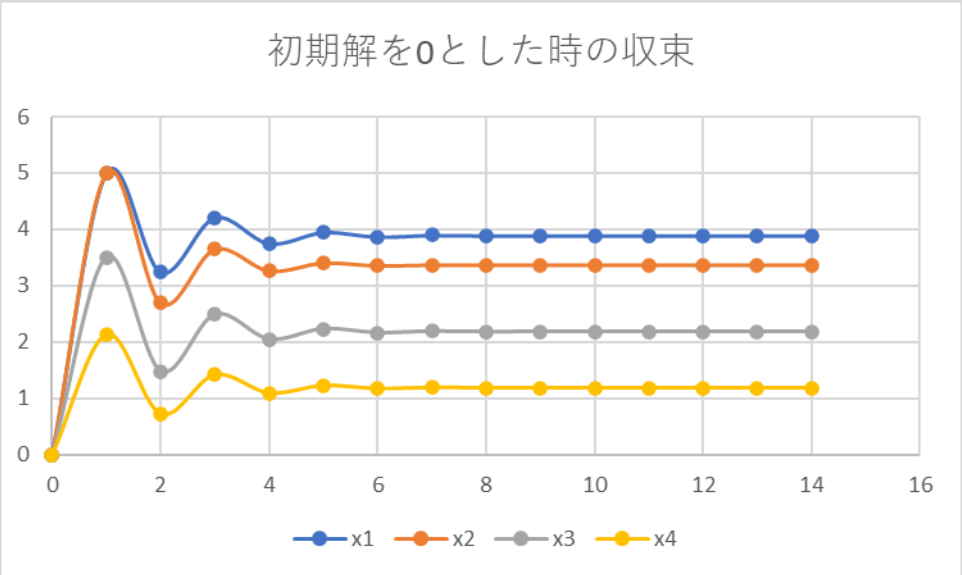
12-->3.886369 3.361439 2.184390 1.178796
13-->3.886373 3.362581 2.183054 1.180373
14-->3.886367 3.363132 2.182411 1.181125
15-->3.886365 3.363394 2.182102 1.181486
16-->3.886365 3.363521 2.181954 1.181659
17-->3.886364 3.363581 2.181883 1.181742
18-->3.886364 3.363610 2.181849 1.181782
19-->3.886364 3.363624 2.181833 1.181801



初期解を 0 とした時
8.000000 1.000000 2.000000 1.000000 40.000000
2.000000 4.000000 1.000000 -2.000000 20.000000
2.000000 0.000000 6.000000 1.000000 21.000000
1.000000 3.000000 -2.000000 8.000000 17.000000

0-->0.000000 0.000000 0.000000 0.000000
1-->5.000000 5.000000 3.500000 2.125000
2-->3.234375 2.687500 1.479167 0.718750
3-->4.204427 3.645833 2.484375 1.414063
4-->3.746419 3.263021 2.049045 1.091146
5-->3.943468 3.401801 2.230469 1.221517
6-->3.864468 3.352241 2.162480 1.168728

7-->3.894259 3.367624 2.187798 1.187222
8-->3.883695 3.362849 2.179588 1.180428
9-->3.887193 3.363892 2.182312 1.182405
10-->3.886135 3.363678 2.181635 1.181748
11-->3.886413 3.363626 2.181816 1.181864
12-->3.886360 3.363665 2.181814 1.181832
13-->3.886359 3.363630 2.181806 1.181820
14-->3.886367 3.363643 2.181820 1.181824



＜理解した内容、感想、注意点など＞
・ヤコビ法よりもガウスザイデル法のほうが収束する速度が速い.
・ヤコビ法よりもガウスザイデル法のほうがグラフの変化が小さいことが分かった.