上机题第三题实验报告

王伟任　　计 33　　2013011333

一、题目要求及分析

第三章上机题 6：编程序生成 Hilbert 矩阵$\boldsymbol{H_n}$，以及 n 维向量$\mathbf{b} = \boldsymbol{H_n x}$，其中 $\mathbf{x}$ 为所有分量都是 1 的向量。用 Cholesky 分解算法求解方程$\boldsymbol{H_n x} = \mathbf{b}$，得到近似解$\hat{\boldsymbol{x}}$，计算残差 $\mathbf{r} = \mathbf{b} - \boldsymbol{H_n}\hat{\boldsymbol{x}}$ 和误差 $\Delta\mathbf{x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$ 的∞-范数。

（1）设 n=10，计算$||\boldsymbol{r}||_\infty$、$||\Delta\boldsymbol{x}||_\infty$；

（2）在右端项上施加10^{-7}的扰动然后解方程组，观察残差和误差的变化情况；

（3）改变 n 的值为 8 和 12，求解相应的方程，观察$||\boldsymbol{r}||_\infty$、$||\Delta\boldsymbol{x}||_\infty$的变化情况，通过这个实验说明了什么问题？

利用 Cholesky 分解算法求解方程$\boldsymbol{H_n x} = \mathbf{b}$，只需要执行一次前代过程和一次回代过程（经过 Cholesky 分解后，原方程可化为$\boldsymbol{LL^T x} = \boldsymbol{b}$，可以先求解方程$\boldsymbol{Ly} = \boldsymbol{b}$，再求解方程$\boldsymbol{L^T x} = \boldsymbol{y}$），复杂度较小。值得注意的是在右端项添加扰动之后解方程组得到的结果与原结果的差异取决于 Hilbert 矩阵的条件数大小，因此为了更加清楚的了解这个情况，另行编写程序计算 Hilbert 矩阵的条件数。

二、实验结果及分析

（1） 计算$||\boldsymbol{r}||_\infty$和$||\Delta\boldsymbol{x}||_\infty$

经计算，得到的近似解$\hat{\boldsymbol{x}}$结果为：

$(1.000000, 1.000000, 0.999998, 1.000021, 0.999900, 1.000273, 0.999551, 1.000433, 0.999773, 1.000050)^T$

误差范数为 0.0004485575，残差范数为 0.00000000000000022204。

结果储存在 test0.txt 中。

可以看到此时的误差和残差都比较小，在可接受范围内。

（2） 添加扰动后得到的结果

由于题目中并未说明扰动加在向量的哪个分量上，因此将扰动依次加在十个分量上，比较得到的结果。得到的结果如下：

• 加到**第一分量**上，得到的近似解$\hat{\boldsymbol{x}}$结果为：

$(1.000010, 0.999505, 1.007916, 0.939977, 1.252074, 0.369861, 1.960152, 0.125622, 1.437169, 0.907713)^T$

误差范数为 0.9601524784，残差范数为 0.00000000000000044409。

• 加到**第二分量**上，得到的近似解$\hat{\boldsymbol{x}}$结果为：

$(0.999505, 1.032666, 0.412016, 5.756054, -19.807451, 54.504253, -82.228016, 77.999415, -37.980665, 9.3123$

误差范数为 83.2280159311，残差范数为 0.00000000000000222045。

• 加到**第三分量**上，得到的近似解$\hat{\boldsymbol{x}}$结果为：

$(1.007919, 0.412018, 12.289088, -94.119847, 429.034954, -1122.582153, 1776.524313, -1662.184795, 851.48$

误差范数为 1775.5243125789，残差范数为 0.00000000000003153033。

• 加到**第四分量**上，得到的近似解$\hat{\boldsymbol{x}}$结果为：

$(0.939948, 5.756044, -94.119870, 825.365160, -3785.900732, 10099.342099, -16156.265775, 15288.884359,$

误差范数为 16157.2657752129，残差范数为 0.00000000000027400304。

• 加到**第五分量**上，得到的近似解$\hat{\boldsymbol{x}}$结果为：

$(1.252213, -19.807399, 429.035051, -3785.900618, 17673.123407, -47713.552725, 77115.179873, -73572.0$

误差范数为 77114.1798725634，残差范数为 0.00000000000280797607。

• 加到**第六分量**上，得到的近似解$\tilde{\boldsymbol{x}}$结果为：

$(0.369477, 54.504111, -1122.582428, 10099.341870, -47713.553127, 130131.341131, -212062.899502, 2037$

误差范数为 212063.8995017390，残差范数为 0.00000000000369249076。

• 加到**第七分量**上，得到的近似解$\tilde{\boldsymbol{x}}$结果为：

$(1.960783, -82.227783, 1776.524755, -16156.265327, 77115.180182, -212062.898543, 348003.155992, -336$

误差范数为 348002.1559917043，残差范数为 0.00000000000458877381。

• 加到**第八分量**上，得到的近似解$\tilde{\boldsymbol{x}}$结果为：

$(0.125011, 77.999190, -1662.185228, 15288.883975, -73572.018304, 203741.840814, -336333.327904, 3267$

误差范数为 336334.3279038350，残差范数为 0.00000000000634092778。

• 加到**第九分量**上，得到的近似解$\tilde{\boldsymbol{x}}$结果为：

$(1.437490, -37.980547, 851.487804, -7881.785231, 38202.281951, -106417.069854, 176576.466178, -17229$

误差范数为 176575.4661780928，残差范数为 0.00000000000346078721。

• 加到**第十分量**上，得到的近似解$\tilde{\boldsymbol{x}}$结果为：

$(0.907642, 9.312295, -181.871504, 1707.808229, -8319.720126, 23299.086370, -38829.242698, 38038.87169$

误差范数为 38830.2426978130，残差范数为 0.00000000000078559381。

原结果储存在 test1.txt 中。

可以看出，扰动加在不同的位置对结果的影响有一定差异，但是误差都会变得很大，就连误差最小的情况（加在第一分量上）也有接近 1 的误差，当加在其他分量上误差最大能达到 348002。因此，可以得出结论，Hilbert 矩阵是一个极为病态的矩阵，添加扰动对于结果的影响极大。

以上的结果是从直观上来看 Hilbert 矩阵的病态，从理论上来看，Hilbert 矩阵的条件数是 35352489096888.9531250000，足见其病态性。

文件 test.txt 中储存了 Hilbert 矩阵的逆矩阵。

（3） 改变 n 的值，观察$||\boldsymbol{r}||_\infty$、$||\Delta\boldsymbol{x}||_\infty$的变化情况

当 n=8 时，得到的近似解$\tilde{\boldsymbol{x}}$结果是：

$(1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000001, 1.000000, 1.000000)^T$

误差范数为 0.0000006968，残差范数为 0.00000000000000044409。

当 n=12 时，得到的近似解$\tilde{\boldsymbol{x}}$结果是：

$(1.000000, 1.000006, 0.999805, 1.002674, 0.980302, 1.086898, 0.757043, 1.441095, 0.481531, 1.380577, 0.8414$

误差范数为 0.5184688315，残差范数为 0.00000000000000044409。

原结果分别储存在 test2.txt 和 test3.txt 中。

综合 n=8，n=10，n=12 三种情况下的解可以得出结论，当 n 越大时，方程组的解误差越大。

三、实验代码

采用 C++语言实现。

由于各问的代码基本类似，因此只附第二问和计算条件数的代码。

第二问代码：

```
#include <cstdio>
#include <cmath>

int n = 10;
```

```
double** createHilbert(){
	double** temp = new double*[n];
	for (int i = 0; i < n; i++){
		temp[i] = new double[n];
		for (int j = 0; j < n; j++)
			temp[i][j] = 1.0 / (i + j + 1);
	}
	return temp;
}

double* multiply(double** A, double* bl){
	double* ans = new double[n];
	for (int i = 0; i < n; i++){
		ans[i] = 0;
		for (int j = 0; j < n; j++)
			ans[i] += A[i][j] * bl[j];
	}
	return ans;
}

double** Cholesky(double** Hilbert){
	double** L = new double*[n];
	for (int i = 0; i < n; i++)
		L[i] = new double[n];
	for (int i = 0; i < n; i++)
		for (int j = i + 1; j < n; j++)
			L[i][j] = 0;

	L[0][0] = sqrt(Hilbert[0][0]);
	for (int i = 1; i < n; i++)
		L[i][0] = Hilbert[i][0] / L[0][0];
	for (int j = 1; j < n; j++){
		double ss = 0.0;
		for (int i = 0; i < j; i++)
			ss += L[j][i] * L[j][i];
		L[j][j] = sqrt(Hilbert[j][j] - ss);
		for (int i = j + 1; i < n; i++){
			double s = 0.0;
			for (int ii = 0; ii < j; ii++)
				s += L[i][ii] * L[j][ii];
			L[i][j] = (Hilbert[i][j] - s) / L[j][j];
		}
	}
```

```
        return L;
}

double* solve(double** L, double* b){
        double* y = new double[n];
        for (int i = 0; i < n; i++){
                y[i] = b[i] / L[i][i];
                for (int j = 0; j < i; j++)
                        y[i] = y[i] - y[j] * L[i][j] / L[i][i];
        }
        return y;
}

double* solve2(double** L, double* y){
        double* x = new double[n];
        for (int i = n-1; i >= 0; i--){
                x[i] = y[i] / L[i][i];
                for (int j = i + 1; j < n; j++)
                        x[i] = x[i] - x[j] * L[j][i] / L[i][i];
        }
        return x;
}

double norm(double* A, double* B){
        double mmax = 0.0;
        for (int i = 0; i < n; i++)
                if (fabs(A[i] - B[i]) > mmax) mmax = fabs(A[i] - B[i]);
        return mmax;
}

int main(){
        double** Hilbert = createHilbert();
        double** L = Cholesky(Hilbert);
        double* x0 = new double[n];
        for (int i = 0; i < n; i++) x0[i] = 1;
        double *x, *y, *r, *b;

        FILE* fp = fopen("test1.txt", "w");

        for (int k = 0; k < n; k++){
                b = multiply(Hilbert, x0);
                b[k] = b[k] + 0.0000001;
                y = solve(L, b);
                x = solve2(L, y);
```

```
            r = multiply(Hilbert, x);
            fprintf(fp, "k = %d\n", k);
            fprintf(fp, "Jin si jie x:\n");
            for (int i = 0; i < n; i++)
                fprintf(fp, "%f\t", x[i]);
            fprintf(fp, "\n");
            fprintf(fp, "Wu cha fan shu:    %.10f\n", norm(x, x0));
            fprintf(fp, "Can cha fan shu:    %.20f\n\n", norm(r, b));
        }

        fclose(fp);
        delete[] b;
        delete[] x0;
        delete[] y;
        delete[] x;
        delete[] r;
        for (int i = 0; i < n; i++)
            delete[] Hilbert[i];
        delete[] Hilbert;
        for (int i = 0; i < n; i++)
            delete[] L[i];
        delete[] L;
        return 0;
}
```

计算条件数代码：

```
#include <cstdio>
#include <cmath>

int n = 10;

double** createHilbert(){
        double** temp = new double*[n];
        for (int i = 0; i < n; i++){
            temp[i] = new double[n * 2];
            for (int j = 0; j < n; j++)
                temp[i][j] = 1.0 / (i + j + 1);
            for (int j = n; j < 2 * n; j++)
                temp[i][j] = 0;
            temp[i][n + i] = 1.0;
        }
        return temp;
}
```

```
double norm(double** h){
	double mmax = 0;
	for (int i = 0; i < n; i++){
		double s = 0;
		for (int j = 0; j < n; j++)
			s += fabs(h[i][j]);
		if (s > mmax) mmax = s;
	}
	return mmax;
}

int main(){
	double** HI = createHilbert();

	double** Hilbert = new double*[n];
	for (int i = 0; i < n; i++){
		Hilbert[i] = new double[n];
		for (int j = 0; j < n; j++)
			Hilbert[i][j] = HI[i][j];
	}

	for (int i = 1; i < n; i++)
		for (int j = 0; j < i; j++){
			double bk = HI[i][j] / HI[j][j];
			for (int k = j; k < 2 * n; k++)
				HI[i][k] = HI[i][k] - HI[j][k] * bk;
		}
	for (int i = n-2; i >= 0; i--)
		for (int j = n-1; j > i; j--){
			double bk = HI[i][j] / HI[j][j];
			for (int k = j; k < 2 * n; k++)
				HI[i][k] = HI[i][k] - HI[j][k] * bk;
		}
	for (int i = 0; i < n; i++){
		double bk = HI[i][i];
		for (int j = 0; j < 2 * n; j++)
			HI[i][j] = HI[i][j] / bk;
	}

	double** invHilbert = new double*[n];
	for (int i = 0; i < n; i++){
		invHilbert[i] = new double[n];
		for (int j = 0; j < n; j++)
			invHilbert[i][j] = HI[i][j + n];
```

```
    }

    FILE* fp = fopen("test.txt", "w");
    fprintf(fp, "Hilbert:\n");
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++)
            fprintf(fp, "%26f\t", Hilbert[i][j]);
        fprintf(fp, "\n");
    }
    fprintf(fp, "invHilbert:\n");
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++)
            fprintf(fp, "%26f\t", invHilbert[i][j]);
        fprintf(fp, "\n");
    }
    fclose(fp);

    printf("%.10f\n", norm(Hilbert) * norm(invHilbert));

    for (int i = 0; i < n; i++)
        delete[] HI[i];
    delete[] HI;
    for (int i = 0; i < n; i++)
        delete[] invHilbert[i];
    delete[] invHilbert;
    for (int i = 0; i < n; i++)
        delete[] Hilbert[i];
    delete[] Hilbert;
    return 0;
}
```