# 实验六

1. **实验内容**

6.3对物理实验中所得的下列数据

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 1.5 | 2 | 2.5 | 3.0 | 3.5 | 4 | 4.5 |
|  | 33.40 | 79.50 | 122.65 | 159.05 | 189.15 | 214.15 | 238.65 | 252.2 |
|  | 5.0 | 5.5 | 6.0 | 6.5 | 7.0 | 7.5 | 8.0 |  |
|  | 267.55 | 280.50 | 296.65 | 301.65 | 310.40 | 318.15 | 325.15 |  |

1. 用公式作曲线拟合。
2. 用指数函数作曲线拟合。
3. 比较上述两条拟合曲线，哪条更好？
4. **解题思路**

（注：程序均采用Java实现）

1. 参考教材P200算法6.2（用法方程方法求解曲线拟合的最小二乘问题）。
2. 首先根据式（6.28）形成矩阵A，继而计算出和向量，得到方程。
3. 解上述所得方程。可以采用分解然后执行前代和回代过程，也可采用高斯LU分解然后执行前代和回代过程。得到的向量即为多项式函数的各项系数（低次到高次）。
4. 用指数函数拟合时，可以对函数式两边同时取对数，转化为一次多项式函数的曲线拟合问题。
5. **算法设计**

|  |
| --- |
| **public** **class** Exp6  {  **public** **static** **void** print(**double** A[][], **int** M, **int** N)  {  **for** (**int** i = 0; i < M; i++)  {  **for** (**int** j = 0; j < N; j++)  {  System.*out*.print(A[i][j] + " ");  }  System.*out*.println();  }  }  **public** **static** **double**[] solve(**double** A[][], **double** b[], **int** N)  {  **double** a[][] = **new** **double**[N][N];  **for** (**int** i = 0; i < N; i++)  {  **for** (**int** j = 0; j < N; j++)  {  a[i][j] = A[i][j];  }  }  // cholesky分解  **for** (**int** j = 0; j < N; j++)  {  **for** (**int** k = 0; k < j; k++)  {  a[j][j] = a[j][j] - a[j][k] \* a[j][k];  }  a[j][j] = Math.*sqrt*(a[j][j]);  **for** (**int** i = j + 1; i < N; i++)  {  **for** (**int** k = 0; k < j; k++)  {  a[i][j] = a[i][j] - a[i][k] \* a[j][k];  }  a[i][j] = a[i][j] / a[j][j];  }  }  **double** \_x[] = **new** **double**[N]; // 第一次回带的中间结果  **double** x[] = **new** **double**[N];  **for** (**int** i = 0; i < N; i++)  {  \_x[i] = b[i];  **for** (**int** j = 0; j < i; j++)  \_x[i] -= (a[i][j] \* \_x[j]);  \_x[i] = \_x[i] / a[i][i];  }  **for** (**int** i = N - 1; i >= 0; i--)  {  x[i] = \_x[i];  **for** (**int** j = N - 1; j > i; j--)  x[i] -= (a[j][i] \* x[j]);  x[i] = x[i] / a[i][i];  }  **return** x;  }  **public** **static** **void** calculate1(**double** x[], **double** y[], **int** M)  {  // 计算矩阵A  **final** **int** N = 3;  **double** A[][] = **new** **double**[M][N];  **for** (**int** i = 0; i < M; i++)  {  A[i][0] = 1;  A[i][1] = x[i];  A[i][2] = x[i] \* x[i];  }  // 计算矩阵G  **double** G[][] = **new** **double**[N][N];  **for** (**int** i = 0; i < N; i++)  {  **for** (**int** j = 0; j < N; j++)  {  **for** (**int** k = 0; k < M; k++)  {  G[i][j] += A[k][i] \* A[k][j];  }  }  }  // 计算向量b  **double** b[] = **new** **double**[N];  **for** (**int** i = 0; i < N; i++)  {  **for** (**int** j = 0; j < M; j++)  {  b[i] += A[j][i] \* y[j];  }  }  // 计算出系数  **double** answer[] = **new** **double**[N];  answer = *solve*(G, b, N);  // 计算均方误差  **double** sum = 0.0;  **for** (**int** i = 0; i < M; i++)  {  **double** result = answer[0] + answer[1] \* x[i] + answer[2] \* x[i]  \* x[i];  sum += (result - y[i]) \* (result - y[i]);  }  sum = Math.*sqrt*(sum / M);  System.*out*.println("第一个拟合的方程为：y=" + answer[0] + "+" + answer[1] + "x+"  + answer[2] + "x^2");  System.*out*.println("均方误差为：" + sum);  }  **public** **static** **void** calculate2(**double** x[], **double** y[], **int** M)  {  // 计算矩阵A  **final** **int** N = 2;    **for** (**int** i=0;i<M;i++)  y[i] = Math.*log*(y[i]);    **double** A[][] = **new** **double**[M][N];  **for** (**int** i = 0; i < M; i++)  {  A[i][0] = 1;  A[i][1] = x[i];  }  // 计算矩阵G  **double** G[][] = **new** **double**[N][N];  **for** (**int** i = 0; i < N; i++)  {  **for** (**int** j = 0; j < N; j++)  {  **for** (**int** k = 0; k < M; k++)  {  G[i][j] += A[k][i] \* A[k][j];  }  }  }  // 计算向量b  **double** b[] = **new** **double**[N];  **for** (**int** i = 0; i < N; i++)  {  **for** (**int** j = 0; j < M; j++)  {  b[i] += A[j][i] \* y[j];  }  }  // 计算出系数  **double** answer[] = **new** **double**[N];  answer = *solve*(G, b, N);  // 计算均方误差  **double** sum = 0.0;  **for** (**int** i = 0; i < M; i++)  {  **double** result = Math.*exp*(answer[0]) \*Math.*exp*(answer[1] \* x[i]);  sum += (result - y[i]) \* (result - y[i]);  }  sum = Math.*sqrt*(sum / M);  System.*out*.println("第二个拟合的方程为：y=" + Math.*exp*(answer[0]) + "\*e^(" + answer[1] + "t)");  System.*out*.println("均方误差为：" + sum);  }  **public** **static** **void** main(String[] args)  {  **final** **int** M = 15;  **double** x[] = { 1, 1.5, 2, 2.5, 3, 3.5, 44, 4.5, 5, 5.5, 6, 6.5, 7, 7.5,  8 };  **double** y[] = { 33.4, 79.5, 122.65, 159.05, 189.15, 214.15, 238.65,  252.2, 267.55, 280.5, 296.65, 301.65, 310.4, 318.15, 325.15 };  *calculate1*(x, y, M);  *calculate2*(x, y, M);  }  } |

1. **实验结果和结论**
2. 用公式作曲线拟合：

拟合结果为：；

。

1. 用指数函数作曲线拟合：

拟合结果为：；

。

1. 通过比较知：用2次多项式函数拟合曲线比指数函数拟合曲线好。我们也可以通过拟合曲线图像直观说明这个问题：