求解线性最小二乘系统

[密集线性问题和分解](#)

本页介绍如何使用 Eigen 求解线性最小二乘系统。一个超定方程组，比如 $Ax = b$，没有解。在这种情况下，搜索最接近解的向量x是有意义的，因为差异$Ax - b$尽可能小。这个x称为最小二乘解（如果使用欧几里得范数）。

本页讨论的三种方法是 SVD 分解、QR 分解和正规方程。其中，SVD分解通常最准确但最慢，正规方程最快但最不准确，QR分解介于两者之间。

使用 SVD 分解

BDCSVD类中的solve()方法可直接用于求解线性平方系统。仅计算奇异值是不够的（此类的默认值）；您还需要奇异向量，但薄 SVD 分解足以计算最小二乘解：

例子：

```cpp
#include <iostream>
#include <Eigen/Dense>

using namespace std;
using namespace Eigen;

int main()
{
    MatrixXf A = MatrixXf::Random(3, 2);
    cout << "Here is the matrix A:\n" << A << endl;
    VectorXf b = VectorXf::Random(3);
    cout << "Here is the right hand side b:\n" << b << endl;
    cout << "The least-squares solution is:\n"
        << A.bdcSvd(ComputeThinU | ComputeThinV).solve(b) << endl;
}
```

输出：

```
Here is the matrix A:
  0.68  0.597
-0.211  0.823
 0.566 -0.605
Here is the right hand side b:
 -0.33
 0.536
-0.444
The least-squares solution is:
-0.67
0.314
```

这是来自页面线性代数和分解的示例。如果您只需要解决最小二乘问题，但对 SVD 本身不感兴趣，则更快的替代方法是CompleteOrthogonalDecomposition。

使用 QR 分解

QR 分解类中的 solve() 方法也计算最小二乘解。有三个 QR 分解类：HouseholderQR（无旋转，快速但不稳定，如果您的矩阵不是 rull 秩），ColPivHouseholderQR（列旋转，因此有点慢但更稳定）和 FullPivHouseholderQR（完全旋转，最慢，比ColPivHouseholderQR）。这是列旋转的示例：

例子：

```
MatrixXf A = MatrixXf::Random(3, 2);
VectorXf b = VectorXf::Random(3);
cout << "The solution using the QR decomposition is:\n"
     << A.colPivHouseholderQr().solve(b) << endl;
```

输出：

```
The solution using the QR decomposition is:
-0.67
0.314
```

使用正规方程

找到$Ax = b$ 的最小二乘解等效于求解正规方程 $A^TAx = A^Tb$。这导致以下代码

例子：

```
MatrixXf A = MatrixXf::Random(3, 2);
VectorXf b = VectorXf::Random(3);
cout << "The solution using normal equations is:\n"
     << (A.transpose() * A).ldlt().solve(A.transpose() * b) << endl;
```

输出：

```
The solution using normal equations is:
-0.67
0.314
```

这种方法通常是最快的，尤其是当 A "又高又瘦"时。但是，如果矩阵 A 甚至是轻度病态，这也不是一个好方法，因为 A^TA的条件数是A的条件数的平方。这意味着与上面提到的更稳定的方法相比，使用正规方程会损失大约两倍的精度。