数值计算方法

绪论

信息管理与工程学院
冯银波

数值计算方法的起源

- **数值计算（计算数学）是数学的一个分支**
  - 西方数学（古希腊）注重逻辑和推理（非实用性）
  - 其主流是分析、代数、几何，计算曾一度被排斥在主流数学之外
  - 中国古代数学注重计算和实用性，如《周髀算经》、《九章算术》等
  - 数学的产生和发展的最直接动力就是计算

数值计算方法的起源

- **如果不使用计算机进行计算**
  - 解析方法——准确、理论性强、速度慢、在实践中应用性低
  - 画图求解——直观、化繁为简、准确度低、不适用高维问题
  - 算盘等计算工具——不适用大规模问题
- **计算机的诞生焕发了计算数学的青春**
  - 人们对计算的需求增加
  - 新方法、新思路随之诞生
  - 原来分散在各个领域的计算方法需要汇总成一门新的科学——数值计算方法

数值计算方法的起源

- 数值计算方法和计算机一起成为不可或缺的工具
- 两者相辅相成、不可分割

数值计算方法——研究如何用计算机求解

数值计算方法的研究对象

- 方程求根
- 方程组求解
- 函数插值
- 函数逼近、数据拟合、最小二乘问题
- 矩阵特征值、特征向量求解
- 数值积分
- 微分方程求解
- 偏微分方程求解

数值方法的意义

例1： 求解线性方程组 $Ax = b$, 其中A为 $n \times n$ 的方阵。假设行列式 $d = \det A \neq 0$.

- 根据Cramer法则，此方程组的解为：

$$x = \frac{d_i}{d},$$

其中，d_i表示将A中的第i列换成b之后所对应的列式。对于n阶方程组，要计算$n+1$个n阶行列式，按照Laplace展开定理：

$$d = a_{i1}A_{i1} + \cdots + a_{in}A_{in}$$

其中A_{ij}表示元素a_{ij}的代数余子式。

- 设计算一个n阶行列式所需要的乘法运算次数为m_n，则容易推出

$$m_n = n + n * m_{n-1} = \cdots > n!$$

计算方法的意义

- 共需计算至少$(n+1)!$次乘法运算。
  - 当 $n=20$ 时，需要进行至少 $21! = 5.11\times10^{19}$ 次乘法运算。
  - 如果一台计算机每秒可以做十万亿次乘法运算，该计算机一年可以做$365\times24\times60\times60\times10^{13}=3.15\times10^{20}$次乘法。那么，在该计算机上用Cramer法则求解20阶线性方程组至少需要59天。
  - 当$n=25$ 时，需要至少128万年
- 这说明，一个不好的计算方法可以让计算过程变得异常之慢！

数值方法的意义

- 例2：求解矩阵的特征值和特征向量。对矩阵进行Jordan分解，理论上讲，就可以知道它的所有特征值的几何重数、代数重数，以及相应的特征向量。然而，Jordan分解往往是十分不稳定的，尤其对于大规模矩阵而言，矩阵元素有微小变化时，Jordan分解往往会发生很大变化，而且其变换矩阵往往是病态的。
- Jordan分解理论上很完美，用计算机求解时，实际计算结果可能会变得毫无意义

数值方法的意义

- 例3：考虑如下两个方程

$$(x-1)^{11}=0;$$

$$(x-1)^{11}+10^{-10}*x^{11}=0.$$

- 第一个方程有唯一实根 $x=1$.
- 调用Matlab中的root命令对第二个方程求解，可得到10个复根和一个实根$x=0.8902$.

- 在用计算机求解时，一点点的误差都有可能导致最终结果出现很大偏差

数值方法的意义

- 例4：求解超越方程 $e^x = 3x$.
- 例5：求平方根，如$\sqrt{7}$.
- 例6：计算定积分

$$\int_0^{\infty} e^{-x^2} dx$$

- **计算机实质上只会做加减乘除等基本运算。数值计算方法就是在研究怎样通过计算机所执行的基本运算，求得各类问题的数值解或近似数值解**

  - 好的数值方法：计算快且准、占用内存小

本课程主要内容

- 方程求根问题
- 线性方程组问题
- 线性最小二乘问题
- 矩阵特征值问题
- 数值积分
- 常微分方程求解

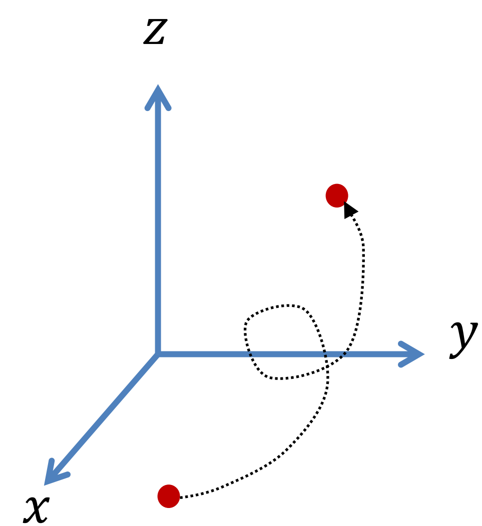
数值线性代数的主要研究内容，以矩阵分解为核心

矩阵计算的重要性

- 科学与工程计算的核心：矩阵计算。包括矩阵乘积、矩阵求逆、计算矩阵特征值/特征向量/奇异值、矩阵分解。
- 很多大规模科学与工程问题最终都要归结为矩阵计算问题

矩阵计算的重要性

- 一个物体的位置可以通过向量来表示，而物体在不同时间的位置变化可以通过矩阵乘法来描述
- 以游乐园大摆锤为例，大摆锤既有左右摆动，还有平面旋转，某游客在摆锤中的运动轨迹如何刻画？



矩阵计算的重要性

- 如果游客在t时刻的坐标(x, y, z)，锤杆末端（转盘中心）的坐标是$(0, b, c)$，已知从t到t'锤杆顺时针摆动了α角度，游客在转盘上顺时针旋转了β角度，那么游客在t'的坐标(x', y', z')可以表示为：

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{b}{\sqrt{b^2+c^2}} & \frac{-c}{\sqrt{b^2+c^2}} \\ 0 & \frac{c}{\sqrt{b^2+c^2}} & \frac{b}{\sqrt{b^2+c^2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \sin\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{b}{\sqrt{b^2+c^2}} & \frac{c}{\sqrt{b^2+c^2}} \\ 0 & \frac{-c}{\sqrt{b^2+c^2}} & \frac{b}{\sqrt{b^2+c^2}} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- 理论上，给定摆锤和转盘的转速，可以计算出游客在任意时刻的坐标。

矩阵计算的重要性

- 矩阵A将线形空间中的每一个点（向量）都做了一次线性变化（伸缩、旋转）
- 矩阵的特征向量就是被矩阵变换之后保持不变的向量，特征值表示特征向量被伸缩的倍数

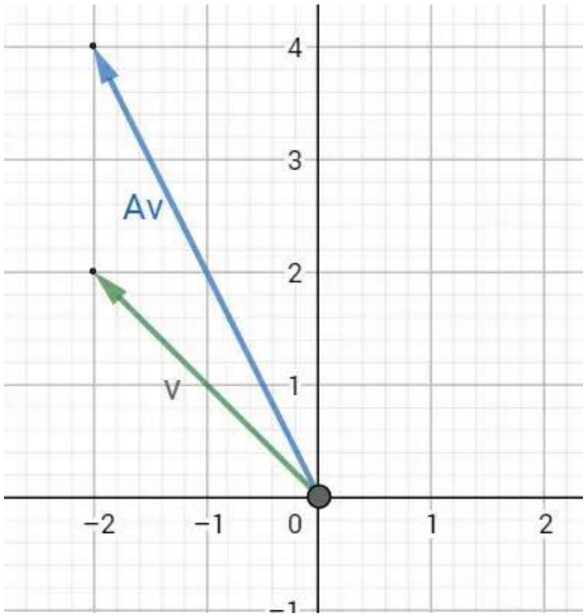
设$\boldsymbol{A}$是n阶方阵，若存在n维非零向量$\boldsymbol{x}$，使

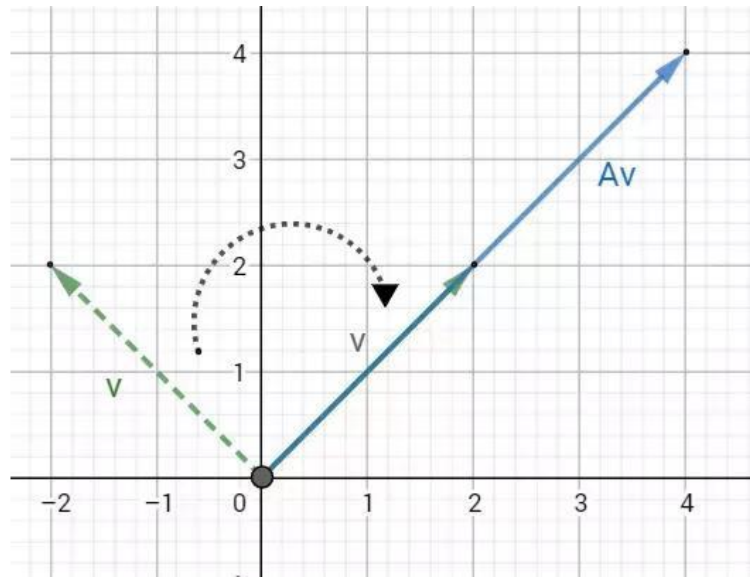
$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

则称数λ为$\boldsymbol{A}$的特征值，$\boldsymbol{x}$为$\boldsymbol{A}$属于λ的特征向量

矩阵计算的重要性

- 对于向量v, 矩阵A将它映射到Av.

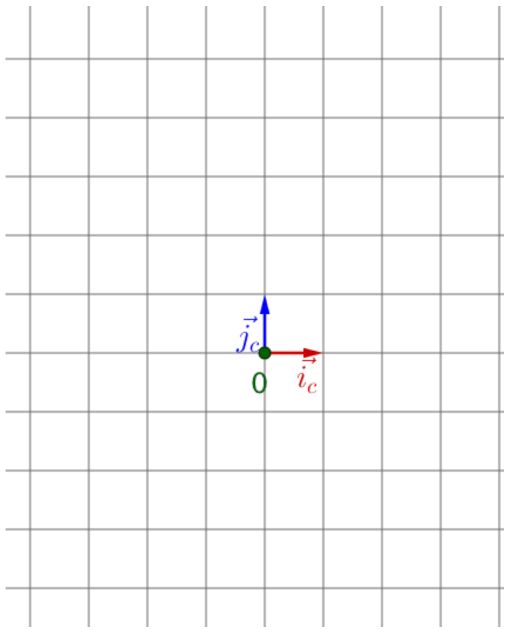




- 对于某个向量v而言，在矩阵A的作用下，仍然保持方向不变，但有可能被伸缩（右图）

矩阵计算的重要性

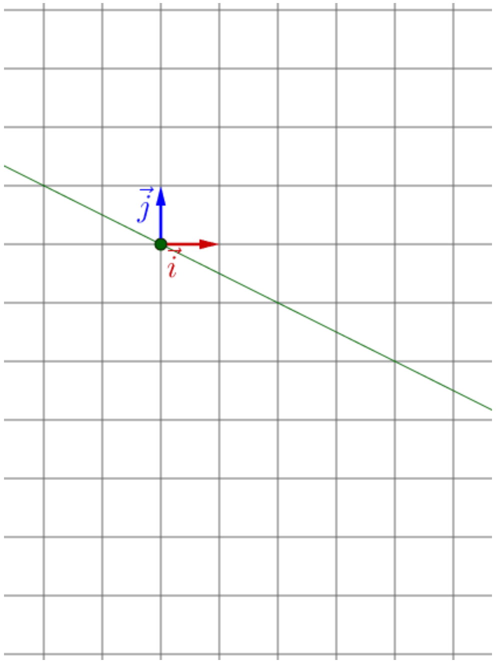
- 对整个空间而言，矩阵A将它整体做了旋转或者伸缩



- 上面所示变换，空间的维度保持不变

矩阵计算的重要性

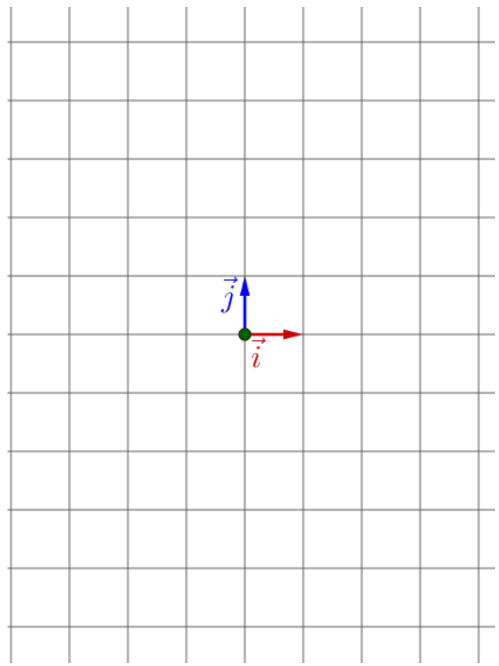
- 对整个空间而言，矩阵A将它整体做了旋转或者伸缩



- 上面所示变换，空间的维度降低了

矩阵计算的重要性

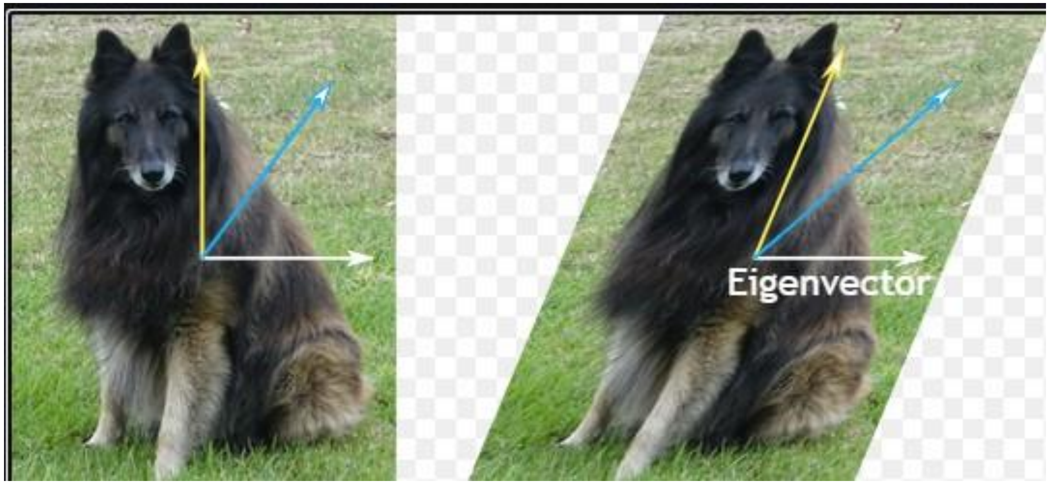
- 对整个空间而言，矩阵A将它整体做了旋转或者伸缩

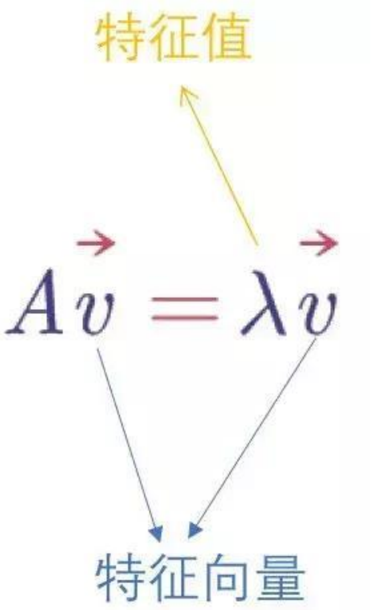


- 上面所示变换，整个空间被映射到了0点

矩阵计算的重要性

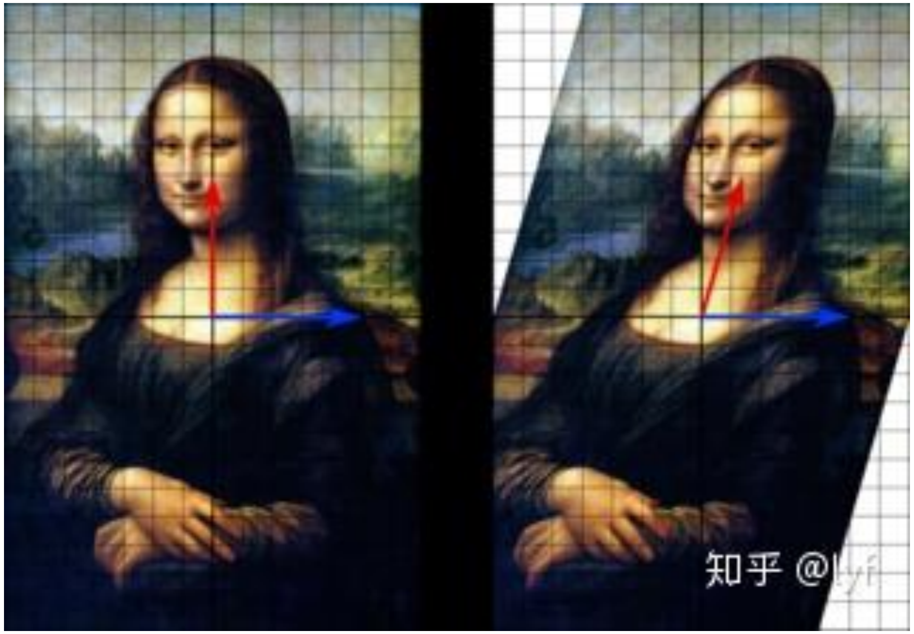
- 如果矩阵A没有对向量v进行旋转，只是伸缩了λ倍，则该向量v是A的一个特征向量，λ即为特征值





- 上图中的白线为特征向量

矩阵计算的重要性

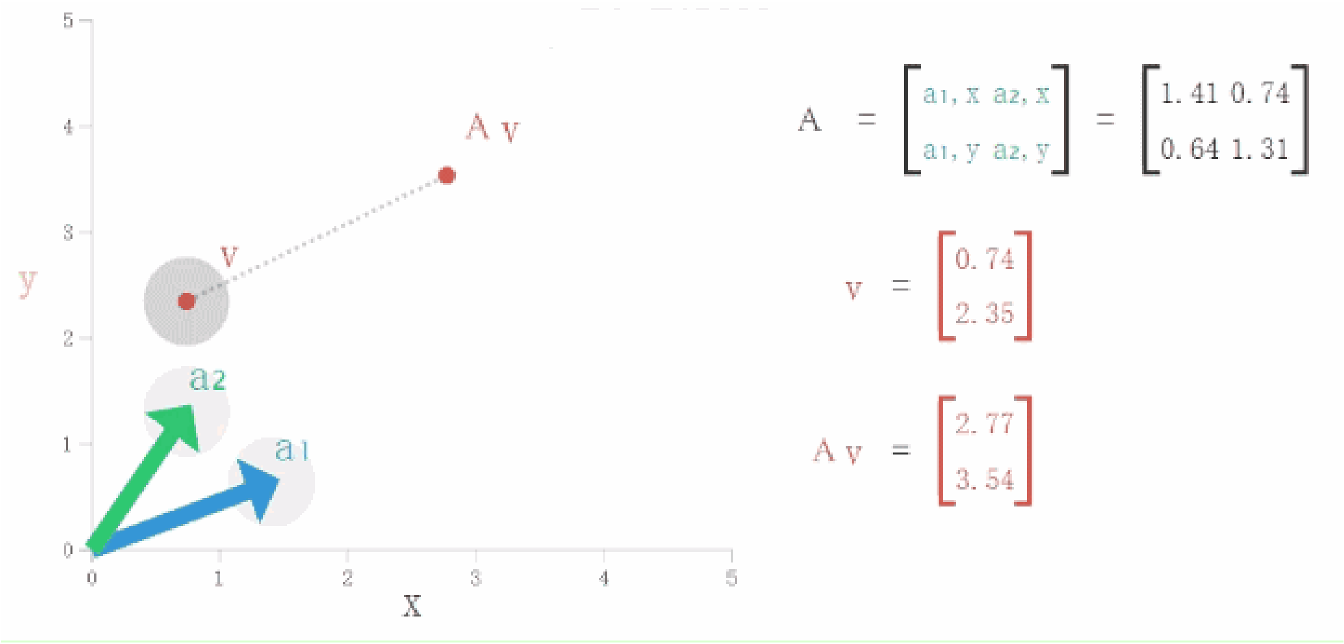


- 上图中的蓝线为特征向量

矩阵计算的重要性

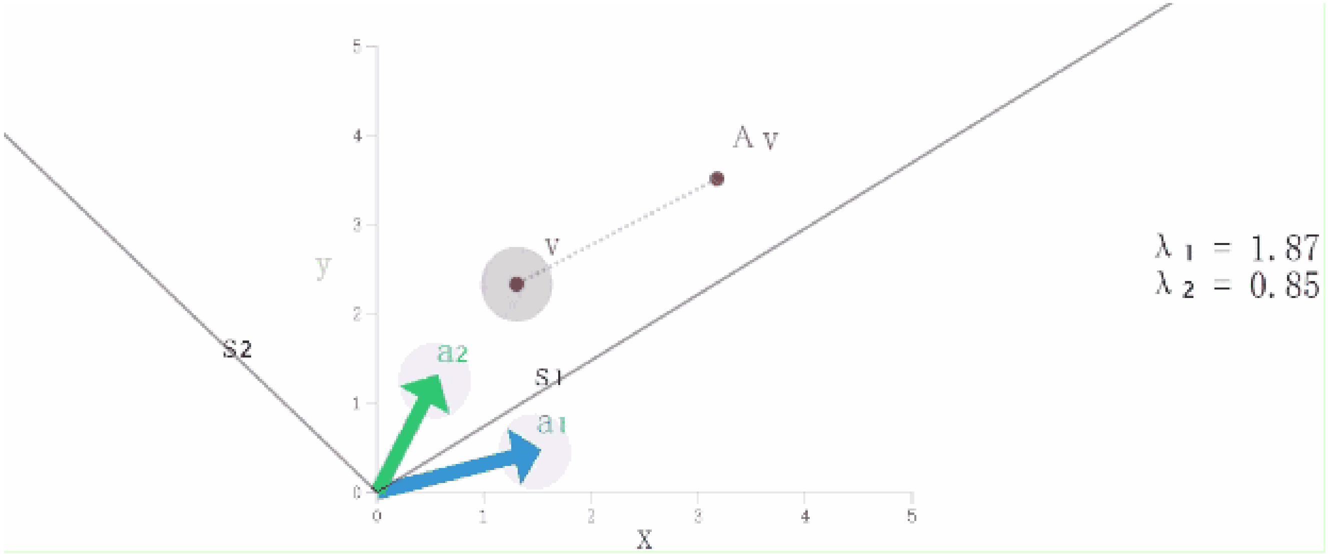
- 通过改变向量v的位置，看看Av的变化

$$A = (a_1, a_2)$$



矩阵计算的重要性

- 通过改变向量v的位置，看看Av的变化



矩阵分解是设计算法的主要技巧

- 结合具体问题的特点，设计相应的算法，方能事半功倍。
- 如果线性方程组的系数矩阵为上三角矩阵或者下三角矩阵，则该方程组易于求解，如：

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- 可依次求出$x_1, x_2, \cdots, x_n$

矩阵分解是设计算法的主要技巧

- 对于线性方程组 $Ax = b$，如果能将矩阵A 分解为： $PA = LU$, 其中P为置换矩阵，L 是下三角矩阵， U 是上三角矩阵；然后求解两个三角形方程组

$$Ly = Pb,\ Ux = y.$$

- x 即为所求.
- 因此，如何求解线性方程组的问题就转化了如何实现上述的矩阵分解问题。
- 矩阵分解是矩阵计算中常用的技巧

数值计算中的误差

- 在计算机上求解一个数值问题，通常不能期望得到准确解。为了使得所得到的解尽可能的准确，有必要理解在数值计算过程中误差是如何产生的

误差的来源

- **描述误差（模型误差）**
  - 实际问题与数学模型之间的差距
- **观测误差**
  - 数据本身存在的误差，用这些数据作为参数进行计算自然会有误差
- **截断误差**
  - 在数值计算中，常用收敛的无穷级数的前有限项来代替无穷级数。如：

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$$

被抛弃的那“无穷多项”所导致的误差叫做截断误差，这类误差可以尽可能地减小，但无法完全消除

误差的来源

- **舍入误差**
  - 计算机的字长是有限的，只能用有限位小数来代替无穷小数或用位数较少的小数来代替位数较多的有限小数，如：

$$\pi = 3.14159265359\ldots, \quad \frac{1}{3} = 0.3333\ldots$$

  - 通过四舍五入/截断，用有限项代替无穷项，必然导致误差。对于单一次运算来说，这些误差也许很小，但正常的计算问题，往往包含大量运算，这些舍入误差积少成多，就会变得十分巨大。
- **数值计算主要研究截断误差和舍入误差带来的影响**

舍入/截断误差举例

- 例7:在MATLAB中计算0.1+0.2

```
>> 0.3-0.3

ans =

     0

>> 0.1+0.2-0.3

ans =

       5.55111512312578e-17
```

- 计算机的字长/精度的限制，0.1转化为二进制后被舍入（或截断）

○ 2进制 ○ 4进制 ○ 8进制 ◉ 10进制 ○ 16进制 ○ 32进制 [10进制]

转换数字 0.1

◉ 2进制 ○ 4进制 ○ 8进制 ○ 10进制 ○ 16进制 ○ 32进制 [2进制]

转换结果 0.0001100110011001100110011001100110011001100110011001101

舍入/截断误差举例

你输入的	计算机存储的（单精度）	计算机存储的（双精度）
0.1	0.100000001490116119384	0.10000000000000000555
0.2	0.200000002980232238770	0.20000000000000001110
0.3	0.300000011920928955078	0.29999999999999998890
0.1+0.2	0.300000011920928955078	0.30000000000000004441

舍入/截断误差举例

- 例8:在MATLAB中计算下列矩阵的特征值

```
M =

     0    -2     2
    -1     0    -1
    -2    -2     0

>> eig(M)

ans =

   1.0e-04 *

  -0.0617 + 0.1068i
  -0.0617 - 0.1068i
   0.1233 + 0.0000i
```

- 请手动计算上述矩阵的特征值

计算解和真实解之间的差距

- 假设我们想要计算函数 f 在 x 点的真实值 $f(x)$。实际上我们往往只能得到计算解$\hat{y}$。
- $\hat{y}$和$f(x)$之间的差距由两个原因导致：
  - 问题本身很敏感（病态）：原始数据 x 可能是从数据中观测到的，也可能是上次运算得到的。所以，x 本身很可能存在扰动，从x 变为 $x+\delta x$，这意味着我们实际上在计算$f(x+\delta x)$。如果原始数据发生一点扰动，即使计算过程中没有误差，也会导致最终结果的变化很大，这说该问题本身就很敏感。
  - 计算方法不当：给定x值（即使x 是精确的），由于截断误差、舍入误差以及算法不当等因素，我们很难计算得到理论上的真实值 $f(x)$。

问题本身病态

例9： 线性方程组

$$\begin{bmatrix} 2.0002 & 1.9998 \\ 1.9998 & 2.0002 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

的（精确）解为$x = (1,1)^T$. 若方程的右端向量发生了扰动，从b变为$b + \delta b$，$\delta b = (2 \times 10^{-4}, -2 \times 10^{-4})^T$, 则原方程组变为

$$\begin{bmatrix} 2.0002 & 1.9998 \\ 1.9998 & 2.0002 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4.0002 \\ 3.9998 \end{bmatrix},$$

它的（精确）解为$\tilde{x} = (1.5,0.5)^T$.

右端向量变化很小，但方程组的（精确）解却变化很大。这说明该线性方程组问题不是一个良态问题。

使用数值不稳定算法的后果

例10：求$I_n=\int_0^1 \frac{x^n}{x+5}dx$，$n=0,1,\ldots,8$ 的值。

解：$I_n+5I_{n-1}=\int_0^1 \frac{x^n+5x^{n-1}}{x+5}dx=\int_0^1 x^{n-1}dx=\frac{1}{n}$（递推公式）

$$I_n=\frac{1}{n}-5I_{n-1}$$

$$I_0=\int_0^1 \frac{1}{x+5}dx=\ln 6-\ln 5=\ln 1.2\approx 0.182$$

根据递推公式，可以逐步得到

$$I_1\approx 0.09, I_2\approx 0.05, I_3\approx 0.083, I_4\approx -0.165$$（居然为负）

这明显是错误的。这是因为，根据递推公式，每一次迭代前一步的误差都会扩大5倍后传给下一步，误差越来越大。

使用数值稳定算法的结果

- 如果改变递推公式：

$$I_{n-1} = \frac{1}{5n} - \frac{1}{5} I_n$$

- 每次迭代时，前一步的误差会被缩小5倍后传给下一步，初始值的扰动对后面计算的影响越来越小
- 对足够大的 n，可以认为$I_{n-1} = I_n$，带入到递推公式便可算出I_n。这里不妨认为 n=10已经足够大。首先计算出 $I_9 \approx 0.017$，然后逐步计算出 $I_8, I_7, \ldots I_0$，最终得到 $I_0 \approx 0.182$和真实值相差很小

- 前一个算法是数值不稳定算法，后者是数值稳定算法
- **在实际应用中应选用数值稳定的算法，尽量避免使用数值不稳定的算法。**

数值方法的优劣

好的数值计算方法法应当满足：

- 计算量小（算法复杂性低）
- 存算效率高（以后讲）
- 数值稳定（误差小）
- 占用内存小
- 逻辑结构简单

想要数值计算结果精确可靠，既需要问题本身良态，也需要方法是数值稳定的

数值计算中的其它注意事项

- 避免两个相似数相减：

例11：当 $x=1000$，求 $y=\sqrt{x+1}-\sqrt{x}$.

解：当 $x=1000$ 时，y 的精确值为0.01580，若两者相减，如果计算机的精度不够高（比如只保留两位小数），那么将会导致结果出现很大偏差：

$$y=\sqrt{1001}-\sqrt{1000}=31.64-31.62=0.02\,.$$

如果改成

$$y=\frac{1}{\sqrt{x+1}+\sqrt{x}}$$

计算效果则会好很多。

- 类似的问题还有：将$\ln y-\ln x$ 写成$\ln\frac{y}{x}$；不能将太小的数作为分母等
- 现代计算机的计算精度对于这类问题的计算结果还是非常理想的，但是大家还是应当注意这些细节，尤其是对运算量较大的数值计算问题

作业（不需要提交）

1 ╱ 学习使用Matlab软件

2 ╱ 用熟悉的编程语言直接计算定积分：

$$y_n = \int_0^1 x^n e^{x-1} dx, \qquad n = 0,1,2,\ldots,20$$

也可通过构造递推公式来进行计算，比较两者的差异。