Введение в численные методы. Итерационные методы решения СЛАУ

Баев А.Ж.

Казахстанский филиал МГУ

18 февраля 2020

План на семестр

1. СЛАУ (точные методы)
2. **СЛАУ (итерационные методы)**
3. решение нелинейных уравнений
4. интерполяция
5. аппроксимация
6. интегрирование
7. дифференцирование

Линейная алгебра

Решение системы линейных алгебраических уравнений:

$$Ax = f$$

где $A \in \mathbb{R}^{n \times n}$, $x \in \mathbb{R}^n$, $f \in \mathbb{R}^n$.
Дано A, f. Найти x.

Каноническая форма

x_0 — задается произвольным образом.

$$B\frac{x_{k+1} - x_k}{\tau} + Ax_k = f$$

где $A > 0$ — исходная матрица, B — невырожденная матрица, зависящая от метода, τ — итерационный параметр.

x_k сходится к решению.

Каноническая форма

$$Bx_{k+1} = Bx_k - \tau A x_k + f\tau$$

где A — исходная матрица, B — невырожденная матрица, зависящая от метода, τ — итерационный параметр.

Необходимо выбирать B такую, что её можно обратить быстрее, чем матрицу A.

Метод простой итерации

$$B = E$$

$$x_{k+1} = x_k - \tau A x_k + \tau f$$

Определим τ.

Достаточное условие сходимости (теорема Самарского):

$$E - \frac{\tau}{2} A > 0 \Leftrightarrow 1 - \frac{\tau}{2}\lambda_i > 0 \Leftrightarrow \tau < \frac{2}{\lambda_i}$$

Необходимое и достаточное условие сходимости:

$$z_{k+1} = z_k - \tau A z_k = (E - \tau A) z_k \Leftrightarrow |1 - \tau\lambda_i| < 1 \Leftrightarrow \tau < \frac{2}{\lambda_i}$$

Метод простой итерации (скорость сходимости)

Обозначим $S = E - \tau A$.

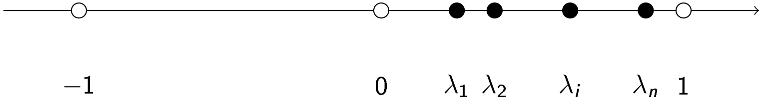
$$z_{k+1} = Sz_k$$

$$||z_{k+1}|| \leqslant ||S|| \cdot ||z_k||$$

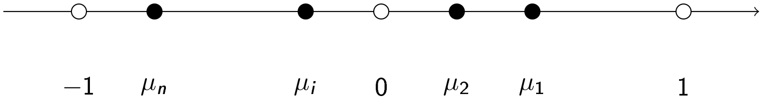
Чем меньше $||S||$, тем быстрее сходится метод.

Метод простой итерации (скорость сходимости)

Рассмотрим спектр A ($\lambda_i > 0$):



Рассмотрим спектр S ($\mu_i = 1 - \tau\lambda_i$):



Максимальная скорость сходимости:

$$\mu_1 + \mu_n = 0$$

$$1 - \tau_o\lambda_1 + 1 - \tau_o\lambda_n = 0$$

$$\tau_o = \frac{2}{\lambda_1 + \lambda_n}$$

Метод простой итерации

Коэффициент сходимости

$$\rho = 1 - \tau_o \lambda_n = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = \frac{1 - \xi}{1 + \xi}$$

где $\xi = \frac{\lambda_n}{\lambda_1} = M_A$ — число обусловленности матрицы A.

Метод верхней и нижней релаксации

Пусть $A = L + D + U$, где
L — строго нижнетреугольная,
D — диагональная,
U — строго верхнетреугольная.

Так как $A = A^*$, то $L^* = U \Leftrightarrow (Lx, x) = (x, Ux) = (Ux, x)$.

Верхняя релаксация:

$$B = \tau L + D$$

Нижняя релаксация:

$$B = D + \tau U$$

Метод верхней релаксации

Проверим условие из теоремы Самарского

$$\tau L + D - \frac{\tau}{2}(L + D + U) > 0$$

$$L + \frac{2-\tau}{2\tau} D - U > 0$$

$$(Lx, x) + \frac{2-\tau}{2\tau}(Dx, x) - (Ux, x) > 0$$

С учётом, что $(Lx, x) = (Ux, x)$.

$$\frac{2-\tau}{2\tau}(Dx, x) > 0$$

Так как $A > 0$, то $D > 0$ (для этого подставим $(Ax, x) > 0$ поочерёдно все базовые вектора).

Получаем, что $0 < \tau < 2$.

Метод Зейделя

$$B\frac{x^{k+1}-x^k}{\tau}+Ax^k=f$$

При $\tau=1$ — метод Зейделя.

$$(L+D)(x^{k+1}-x^k)+(L+D+U)x^k=f$$

$$(L+D)x^{k+1}+Ux^k=f$$

Выпишем поэлементно:

$$\sum_{j=1}^{i-1}a_{i,j}x_j^{k+1}+a_{i,i}x_i^{k+1}+\sum_{j=i+1}^{n}a_{i,j}x_j^k=f_i$$

Выразим x^{k+1}:

$$x_i^{k+1}=\frac{1}{a_{i,i}}\left(f_i-\sum_{j=1}^{i-1}a_{i,j}x_j^{k+1}-\sum_{j=i+1}^{n}a_{i,j}x_j^k\right)$$

Заметим, что при реализации нет необходимо хранить все слои, кроме последнего.

Метод Зейделя

```
seidel(A, f, x)
    xnew[1:n]
    for i := 1 .. n
        s := 0
        for j := 1 .. i-1
            s := s + A[i][j] * xnew[j]
        for j := i+1 .. n
            s := s + A[i][j] * x[j]
        xnew[i] := (f[i] - s) / A[i][i]
    return xnew

solve(A, f)
    xnew[1:n]
    do
        x = xnew
        xnew = seidel(A, f, x)
    while diff(x, xnew) > eps
```

Метод Зейделя

$$\begin{pmatrix} 2 & -1 & 0 & -1 \\ 0 & 2 & -1 & 0 \\ -1 & 1 & 3 & -0 \\ 1 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Метод Зейделя

Представим A как сумму диагональной и треугольных матриц:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & 0 & -2 & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{pmatrix} + \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{pmatrix} +$$

$$+ \begin{pmatrix} 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Выразим явно:

$$\tilde{x}_1 = \frac{4 - (-1) * x_2 - 0 * x_3 - (-1) * x_4}{2}$$

$$\tilde{x}_2 = \frac{3 - 0 * \tilde{x}_1 - (-1) * x_3 - 0 * x_4}{2}$$

$$\tilde{x}_3 = \frac{2 - (-1) * \tilde{x}_1 - 1 * \tilde{x}_2 - 0 * x_4}{3}$$

$$\tilde{x}_4 = \frac{1 - 1 * \tilde{x}_1 - 0 * x_2 - (-2) * x_3}{4}$$

Метод Зейделя

В качестве начального приближения выберем нулевой вектор, а допустимую точность выберем $\varepsilon = 0.2$ Выпишем итерации:

$$(0,0,0,0) \to \left(2, \frac{3}{2}, \frac{5}{6}, \frac{1}{6}\right) \to \left(\frac{17}{6}, \frac{23}{12}, \frac{35}{36}, \frac{1}{36}\right)$$

Разница между последними двумя векторами $||x - \tilde{x}||^2 < \varepsilon^2$.
Ответ:

$$x = (2.83333, 1.91667, 0.972222, 0.027778)^T$$

(точное решение $(3, 2, 1, 0)$).

Метод Якоби

$$B\frac{x^{k+1}-x^k}{\tau}+Ax^k=f$$

При $B=D$ — метод Якоби.

$$D(x^{k+1}-x^k)+(L+D+U)x^k=f$$

$$Dx^{k+1}+(L+U)x^k=f$$

Выпишем поэлементно:

$$\sum_{j=1}^{i-1}a_{i,j}x_j^k+a_{i,i}x_i^{k+1}+\sum_{j=i+1}^{n}a_{i,j}x_j^k=f_i$$

Выразим x^{k+1}:

$$x_i^{k+1}=\frac{1}{a_{i,i}}\left(f_i-\sum_{j=1}^{i-1}a_{i,j}x_j^k-\sum_{j=i+1}^{n}a_{i,j}x_j^k\right)$$

Заметим, что при реализации нет необходимо хранить все слои, кроме последнего.

Метод Якоби

Проверим условие из теоремы Самарского

$$D - \frac{\tau}{2}(L + D + U) > 0$$

$$\frac{2-\tau}{2\tau} D > L + U$$

Скорость сходимости:

$$\rho = ||E - D^{-1}A|| = ||D^{-1}(D - A)||$$

Метод Якоби

```
jacobi(A, f, x)
    xnew[1:n]
    for i := 1 .. n
        s := 0
        for j := 1 .. i-1
            s := s + A[i][j] * xnew[j]
        for j := i+1 .. n
            s := s + A[i][j] * xnew[j]
        xnew[i] := (f[i] - s) / A[i][i]
    return xnew

solve(A, f)
    xnew[1:n]
    do
        x = xnew
        xnew = jacobi(A, f, x)
    while diff(x, xnew) > eps
```

Метод Якоби

$$\begin{pmatrix} 2 & -1 & 0 & -1 \\ 0 & 2 & -1 & 0 \\ -1 & 1 & 3 & -0 \\ 1 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Метод Якоби

Представим A как сумму диагональной и треугольных матриц:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & 0 & -2 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{pmatrix} +$$

$$+ \begin{pmatrix} 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Выразим явно:

$$\tilde{x}_1 = \frac{4 - (-1) * x_2 - 0 * x_3 - (-1) * x_4}{2}$$

$$\tilde{x}_2 = \frac{3 - 0 * x_1 - (-1) * x_3 - 0 * x_4}{2}$$

$$\tilde{x}_3 = \frac{2 - (-1) * x_1 - 1 * x_2 - 0 * x_4}{3}$$

$$\tilde{x}_4 = \frac{1 - 1 * x_1 - 0 * x_2 - (-2) * x_3}{4}$$

Метод Якоби

В качестве начального приближения выберем нулевой вектор, а допустимую точность выберем $\varepsilon = 0.2$ Выпишем итерации:

$$(0,0,0,0) \to \left(2, \frac{3}{2}, \frac{2}{3}, \frac{1}{4}\right) \to \left(\frac{23}{8}, \frac{11}{6}, \frac{5}{6}, \frac{1}{12}\right) \to \left(\frac{71}{24}, \frac{23}{12}, \frac{73}{72}, -\frac{5}{96}\right)$$

Разница между последними двумя векторами $||x - \tilde{x}||^2 < \varepsilon^2$.
Ответ:

$$x = (2.95833, 1.91667, 1.01389, -0.052083)^T$$

(точное решение $(3, 2, 1, 0)$).

Собственные значения

Простая итерация

$$x_{k+1} = Ax_k$$

вытягивает вектор вдоль собственного вектора с максимальным по модулю собственным значением.

$$\lambda \approx \frac{(Ax_k, x_k)}{(x_k, x_k)}$$

Домашнее задание №2

1. Сравнить решение СЛАУ методом Якоби со встроенной функцией *scipy.linalg.solve* на случайной матрице с диагональным преобладанием размером 100×100, 200×200 и т.д. Провести несколько экспериментов, пока время счёта меньше 1 сек. Построить графики зависимостей.
2. Сравнить решение СЛАУ методом Зейделя со встроенной функцией *scipy.linalg.solve* на случайной матрице с диагональным преобладанием размером 100×100, 200×200 и т.д. Провести несколько экспериментов, пока время счёта меньше 1 сек. Построить графики зависимостей.