Итерационные методы решения СЛАУ

Методы установления, Якоби, Зейделя, SOR.

Каноническая форма двухслойного итерационного метода

Цыбулин Иван (tsybulin@crec.mipt.ru)

Метод простой итерации

$$\mathbf{x}_{k+1} = \mathbf{B}\mathbf{x}_k + \mathbf{F}$$

- Сходится, если для какой-то согласованной матричной нормы $\|\mathbf{B}\| = q < 1.$
- Сходится тогда и только тогда, когда $\rho(\mathbf{B}) = \max|\lambda(\mathbf{B})| < 1$
- Сходится к своей неподвижной точке $\mathbf{x}^*$, $\quad \mathbf{x}^* = \mathbf{B}\mathbf{x}^* + \mathbf{F}$

Метод с параметром τ

- Для системы $\mathbf{A}\mathbf{x} = \mathbf{f}$
- На каждой итерации к приближению $\mathbf{x}_k$ прибавляется его невязка $\mathbf{r}_k = \mathbf{f} - \mathbf{A}\mathbf{x}_k$, умноженная на некоторое число $\tau \neq 0$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \tau(\mathbf{f} - \mathbf{A}\mathbf{x}_k)$$

$$\mathbf{x}_{k+1} = \underbrace{(\mathbf{E} - \tau\mathbf{A})}_{\mathbf{B}}\mathbf{x}_k + \underbrace{\tau\mathbf{f}}_{\mathbf{F}}$$

Будем рассматривать случай $\mathbf{A} = \mathbf{A}^\top > 0$. Если это не так, систему можно *симметризовать*

$$\mathbf{A}\mathbf{x} = \mathbf{f} \quad \Longrightarrow \quad \mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{f}$$

На практике стараются избегать симметризовать матрицу системы таким образом, поскольку при этом число обусловленности сильно возрастает

$$\mu(\mathbf{A}^\top \mathbf{A}) \sim \mu(\mathbf{A})^2$$

Сходимость метода с параметром

- Для симметричной $\mathbf{A} = \mathbf{A}^\top > 0$ матрица $\mathbf{B} = \mathbf{E} - \tau\mathbf{A} = \mathbf{B}^\top$.
- $\max|\lambda(B)| = \max(|\lambda_{\min}(\mathbf{B})|, |\lambda_{\max}(\mathbf{B})|)$
- $\lambda(\mathbf{B}) = \lambda(\mathbf{E} - \tau\mathbf{A}) = 1 - \tau\lambda(\mathbf{A})$

Пусть $\tau > 0$

$$\lambda_{\min}(\mathbf{B}) = 1 - \tau\lambda_{\max}(\mathbf{A}) \qquad \lambda_{\max}(\mathbf{B}) = 1 - \tau\lambda_{\min}(\mathbf{A})$$

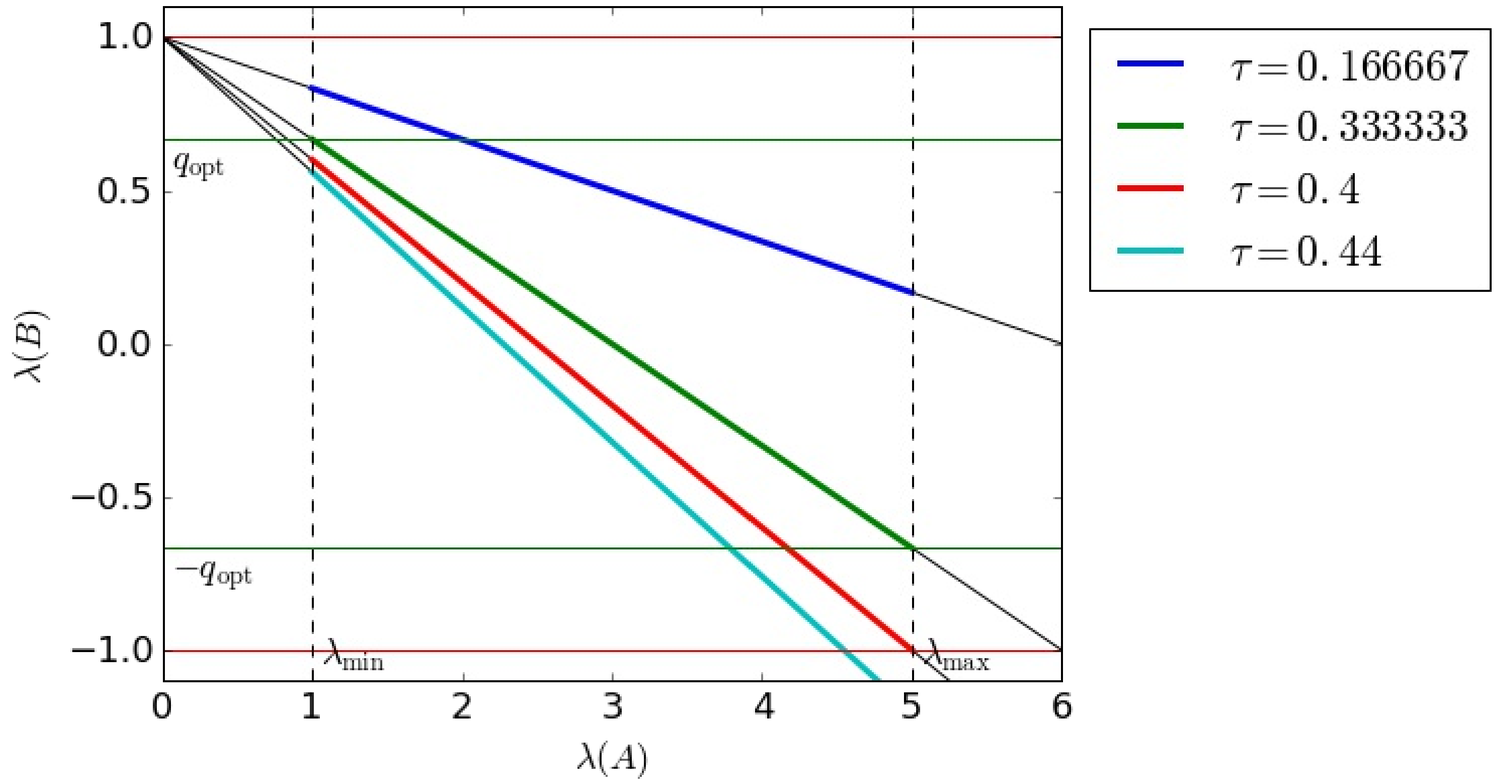
Скорость сходимости в Евклидовой норме

$$q = \|\mathbf{B}\|_E = \max|\lambda(\mathbf{B})| = \max(|1 - \tau\lambda_{\max}(\mathbf{A})|, |1 - \tau\lambda_{\min}(\mathbf{A})|)$$

определяется параметром τ и границами спектра матрицы $\mathbf{A}$. Наивысшая скорость сходимости

$$q_{\text{opt}} = \frac{\lambda_{\max}(\mathbf{A}) - \lambda_{\min}(\mathbf{A})}{\lambda_{\max}(\mathbf{A}) + \lambda_{\min}(\mathbf{A})} = \frac{1 - \mu_E(\mathbf{A})}{1 + \mu_E(\mathbf{A})}, \qquad \tau_{\text{opt}} = \frac{2}{\lambda_{\max}(\mathbf{A}) + \lambda_{\min}(\mathbf{A})}$$

Show code


1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6
$\lambda(B)$
$\lambda(A)$
q_{opt}
$-q_{\text{opt}}$
λ_{min}
λ_{max}
$\tau = 0.166667$
$\tau = 0.333333$
$\tau = 0.4$
$\tau = 0.44$

Оценки сходимости

Пусть в некоторой норме $\|\mathbf{B}\| = q < 1$. Тогда в этой норме

$$\|\mathbf{e}_n\| \leqslant q^n \|\mathbf{e}_0\|$$

Эта оценка показывает, во сколько раз уменьшается норма погрешности за n итераций.

$$\|\mathbf{e}_n\| \leqslant \frac{q^n}{1-q} \|\mathbf{x}_1 - \mathbf{x}_0\|$$

Эта оценка позволяет оценить саму норму ошибки через n итераций.

Пусть нас интересует число итераций, после которых норма ошибки уменьшится в 10^6 раз. Тогда из первой формулы

$$\|\mathbf{e}_n\| \leqslant q^n \|\mathbf{e}_0\|$$

достаточно взять

$$n > \log_q \frac{\|\mathbf{e}_n\|}{\|\mathbf{e}_0\|} = \frac{\ln \|\mathbf{e}_n\| - \ln \|\mathbf{e}_0\|}{\ln q}.$$

Если q достаточно близко к единице, можно использовать $\ln q \approx q - 1$

Пусть нас интересует число итераций, после которых норма ошибки станет 10^{-6}. Тогда из второй формулы

$$\|\mathbf{e}_n\| \leqslant \frac{q^n}{1-q}\|\mathbf{x}_1 - \mathbf{x}_0\|$$

достаточно взять

$$n > \log_q \frac{(1-q)\|\mathbf{e}_n\|}{\|\mathbf{x}_1 - \mathbf{x}_0\|} = \frac{\ln(1-q) + \ln\|e_n\| - \ln\|\mathbf{x}_1 - \mathbf{x}_0\|}{\ln q}.$$

Для этой оценки необходимо сначала сделать одну итерацию метода, чтобы получить $\mathbf{x}_1$.

Метод Якоби

Представим матрицу системы в виде суммы ее диагональной части и остатка:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \underbrace{\begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}}_{\mathbf{D}} + \underbrace{\begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ a_{21} & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & 0 \end{pmatrix}}_{\mathbf{A}-\mathbf{D}}$$

$$\mathbf{A}\mathbf{x} = \mathbf{f} \Longrightarrow \mathbf{D}\mathbf{x} + (\mathbf{A} - \mathbf{D})\mathbf{x} = \mathbf{f}$$

Рассмотрим уравнение

$$\mathbf{D}\mathbf{x} = \mathbf{f} + (\mathbf{D} - \mathbf{A})\mathbf{x}$$

как неподвижную точку итерационного процесса

$$\mathbf{D}\mathbf{x}_{k+1} = \mathbf{f} + (\mathbf{D} - \mathbf{A})\mathbf{x}_k.$$

Этот же процесс можно записать в виде

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{D}^{-1}\mathbf{r}_k \equiv \mathbf{x}_k + \mathbf{D}^{-1}(\mathbf{f} - \mathbf{A}\mathbf{x}_k).$$

Метод Якоби гарантированно сходится для матриц со строгим диалгональным преобладанием.

$$\mathbf{B} = \mathbf{E} - \mathbf{D}^{-1}\mathbf{A} = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \dots & 0 \end{pmatrix}$$

$$\|\mathbf{B}\|_\infty = \max_i \sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} = \max_i \left(\frac{1}{|a_{ii}|} \sum_{j \neq i} |a_{ij}| \right) < 1$$

Этого *достаточно* для сходимости метода Якоби.

Необходимым и достаточным условием сходимости будет условие $\rho(\mathbf{B}) = \max|\lambda(\mathbf{B})| < 1$.

Заметим, что

$$\det(\mathbf{B} - \lambda\mathbf{E}) = \det(\mathbf{E} - \mathbf{D}^{-1}\mathbf{A} - \lambda E) = -\det\mathbf{D}^{-1}\det(\mathbf{A} - \mathbf{D} + \lambda\mathbf{D}).$$

Таким образом, собственные значения матрицы $\mathbf{B}$ удовлетворяют уравнению

$$\begin{vmatrix} a_{11}\lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22}\lambda & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn}\lambda \end{vmatrix} = 0$$

```
def jacobi(A, f, x0, eps=1e-6, maxiter=1000):
    x = x0.copy()
    dx = np.zeros_like(f)
    it = 0
    n = len(x)
    while it < maxiter:
        it += 1
        r = f - A.dot(x)
        for i in range(n):
            dx[i] = r[i] / A[i, i]
        x = x + dx
        if np.linalg.norm(r) < eps:
            return x, it
    raise RuntimeError('Maximum number of iterations exceeded')
```

```
n = 10
A = np.random.rand(n, n) # берем случаную матрицу
A = A + np.diag(A.sum(axis=1)) # и делаем ей диагональное преобладание
x = np.ones(n) # точное решение
f = A.dot(x)
x0 = np.zeros_like(x)
xres, it = jacobi(A, f, x0)
print('Done in %d iterations' % it)
print('||x - x_res|| =', np.linalg.norm(xres - x))
```

```
Done in 137 iterations
||x - x_res|| = 8.23193943958e-08
```

Метод Зейделя

В методе Зейделя матрица разбивается в сумму трех

$$\mathbf{Ax} = \mathbf{f} \Longrightarrow (\mathbf{L} + \mathbf{D})\mathbf{x} + \mathbf{Ux} = \mathbf{f}.$$

Здесь $\mathbf{L}, \mathbf{U}$ — треугольные матрицы из поддиагональных и наддиагональных элементов соответственно. Обратите внимание, эти матрицы не имеют отношения к LU разложению матрицы системы.

Итерационный процесс Зейделя имеет вид

$$(\mathbf{L}+\mathbf{D})\mathbf{x}_{k+1} = \mathbf{f} - \mathbf{U}\mathbf{x}_k.$$

Для определения вектора $\mathbf{x}_{k+1}$ необходимо *на каждой итерации* решать систему с треугольной матрицей $\mathbf{L}+\mathbf{D}$, что делается прямой подстановкой. Отметим, что метод можно записать в форме

$$\mathbf{x}_{k+1} = \mathbf{x}_k + (\mathbf{L}+\mathbf{D})^{-1}\mathbf{r}_k$$

Метод Зейделя гарантированно сходится для положительно определенных симметричных матриц.

Для метода Зейделя собственные числа матрицы $\mathbf{B}$ можно определить из похожего уравнения

$$\begin{vmatrix} a_{11}\lambda & a_{12} & \dots & a_{1n} \\ a_{21}\lambda & a_{22}\lambda & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}\lambda & a_{n2}\lambda & \dots & a_{nn}\lambda \end{vmatrix} = 0$$

Здесь все элементы матрицы $\mathbf{A}$ на диагонали и под ней умножены на λ.

Запишем метод Зейделя в компонентной форме. Будем обозначать верхним индексом номер итерации

$$
\begin{aligned}
a_{11}x_1^{(k+1)} + a_{12}x_2^{(k)} + a_{13}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} &= f_1 \\
a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} + a_{23}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} &= f_2 \\
a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_2^{(k+1)} + \cdots + a_{1n}x_n^{(k)} &= f_3 \\
&\vdots \\
a_{n1}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_2^{(k+1)} + \cdots + a_{1n}x_n^{(k+1)} &= f_n
\end{aligned}
$$

Если решать уравнения сверху вниз, в каждом уравнении единственной неизвестной величиной является диагональная неизвестная.

```python
def seidel(A, f, x0, eps=1e-6, maxiter=1000):
    x = x0.copy()
    n = len(x)
    it = 0
    while it < maxiter:
        it += 1
        for i in range(n):
            Lx = 0 # суммируем начало строки до i-го элемента
            for j in range(i):
                Lx += A[i, j] * x[j] # Здесь x - x_{k+1}
            Ux = 0 # суммируем конец строки с i+1-го элемента
            for j in range(i+1, n):
                Ux += A[i, j] * x[j] # Здесь x - x_k
            # Заменяем диагональный элемент, старый нам уже не понадобится
            x[i] = (f[i] - Lx - Ux) / A[i, i]
        r = f - A.dot(x)
        if np.linalg.norm(r) < eps:
            return x, it
    raise RuntimeError('Maximum number of iterations exceeded')
```

```
n = 10
A = np.random.rand(n, n)
# Делаем случайную симметричную матрицу
A = A.T.dot(A)
# Добавим немного единичной матрицы для лучшей обусловленности A
A = 0.1 * np.eye(n) + A

x = np.ones(n)
f = A.dot(x)
x0 = np.zeros_like(x)
xres, it = seidel(A, f, x0)
print('Done in %d iterations' % it)
print('||x - x_res|| =', np.linalg.norm(xres - x))
```

```
Done in 232 iterations
||x - x_res|| = 2.6383794071e-06
```

Метод SOR

Метод SOR (метод релаксации) является модификацией метода Зейделя. В нем имеется параметр релаксации $\omega \in (0, 2)$:

$$\left(\mathbf{L} + \frac{1}{\omega}\mathbf{D}\right)\mathbf{x}_{k+1} = \mathbf{f} - \left(\mathbf{U} + \frac{\omega - 1}{\omega}\mathbf{D}\right)\mathbf{x}_k.$$

При $\omega = 1$ метод совпадает с методом Зейделя. Метод допускает представление в виде

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \left(\mathbf{L} + \frac{1}{\omega}\mathbf{D}\right)^{-1}\mathbf{r}_k$$

Метод SOR сходится для систем с положительно определенной симметричной матрицей (как и метод Зейделя). Определение оптимального параметра ω является нетривиальной задачей. Для трехдиагональных матриц оптимальное значение ω дается выражением

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho^2(\mathbf{B}_{\text{Якоби}})}}$$

```python
def sor(A, f, x0, omega, eps=1e-6, maxiter=1000):
    x = x0.copy()
    n = len(x)
    it = 0
    while it < maxiter:
        it += 1
        for i in range(n):
            Lx = 0 # суммируем начало строки до i-го элемента
            for j in range(i):
                Lx += A[i, j] * x[j] # Здесь x - x_{k+1}
            Ux = 0 # суммируем конец строки с i+1-го элемента
            for j in range(i+1, n):
                Ux += A[i, j] * x[j] # Здесь x - x_k
            # Заменяем диагональный элемент, старый нам уже не понадобится
            x[i] = x[i] + omega * ((f[i] - Lx - Ux) / A[i, i] - x[i])
        r = f - A.dot(x)
        if np.linalg.norm(r) < eps:
            return x, it
    raise RuntimeError('Maximum number of iterations exceeded')
```

```
n = 30
A = np.diag(2.001 * np.ones(n)) - np.diag(np.ones(n-1), k=-1) - np.diag(np.ones(n-1), k=1)
D = np.diag(np.diag(A))
B_J = np.linalg.solve(D, A - D)
rho_B_J = np.abs(np.linalg.eigvals(B_J)).max()
omega = 2 / (1 + np.sqrt(1 - rho_B_J**2))

x = np.ones(n)
f = A.dot(x)
x0 = np.zeros_like(x)
xres, it = seidel(A, f, x0)
xres2, it2 = sor(A, f, x0, omega)
print('Seidel done in %d iterations' % it)
print('SOR done in %d iterations' % it2)
print('Seidel ||x - x_res|| =', np.linalg.norm(xres - x))
print('SOR ||x - x_res|| =', np.linalg.norm(xres2 - x))
```

```
Seidel done in 971 iterations
SOR done in 77 iterations
Seidel ||x - x_res|| = 8.76532826947e-05
SOR ||x - x_res|| = 2.01191621378e-05
```

Каноническая форма

Все перечисленные методы были двухслойными, то есть для вычисления нового приближения $\mathbf{x}_{k+1}$ требовалось знать лишь предыдущее приближение $\mathbf{x}_k$. Более того, итерацию каждого метода можно представить в виде схемы

$$\mathbf{x}_k \xrightarrow{\mathbf{r}=\mathbf{f}-\mathbf{A}\mathbf{x}} \mathbf{r}_k \xrightarrow{\mathbf{P}\Delta\mathbf{x}=\mathbf{r}} \Delta\mathbf{x}_k \xrightarrow{\mathbf{x}_{k+1}=\mathbf{x}_k+\Delta\mathbf{x}_k} \mathbf{x}_{k+1}$$

Запишем эту схему в форме

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{P}^{-1}\mathbf{r}_k \equiv \mathbf{x}_k + \mathbf{P}^{-1}(\mathbf{f} - \mathbf{A}\mathbf{x}_k)$$

или

$$\mathbf{P}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{f} - \mathbf{A}\mathbf{x}_k.$$

Матрица $\mathbf{P}$ называется *предобуславливателем*.

$$\mathbf{P}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{f} - \mathbf{A}\mathbf{x}_k.$$

Матрица $\mathbf{P}$ должна удовлетворять следующим требованиям

- Система относительно $\Delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ с матрицей $\mathbf{P}$ должна решаться сравнительно просто
- Выбор хорошей матрицы $\mathbf{P}$ должен ускорять сходимость итераций

Стандартные предобуславливатели

Для рассмотренных методов предобуславливателями были

- Метод с параметром τ: $\mathbf{P} = \frac{1}{\tau}\mathbf{E}$
- Метод Якоби: $\mathbf{P} = \mathbf{D}$
- Метод Зейделя: $\mathbf{P} = \mathbf{L} + \mathbf{D}$
- Метод SOR: $\mathbf{P} = \mathbf{L} + \frac{1}{\omega}\mathbf{D}$

Матрица итераций $\mathbf{B}$ простым образом связана с предобуславливателем $\mathbf{P}$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{P}^{-1}(\mathbf{f} - \mathbf{A}\mathbf{x}_k) = (\underbrace{\mathbf{E} - \mathbf{P}^{-1}\mathbf{A}}_{\mathbf{B}})\mathbf{x}_k + \mathbf{P}^{-1}\mathbf{f}.$$

Видно, что идеальная сходимость (за одну итерацию) наступает при $\mathbf{P} = \mathbf{A}$. Однако систему с такой матрицей решить просто не получится. Поэтому для итерационных методов стараются выбирать $\mathbf{P} \approx \mathbf{A}$.

Критерий сходимости в канонической форме

Запишем уравнение для $\lambda(\mathbf{B})$ в форме, содержащей лишь матрицы $\mathbf{A}, \mathbf{P}$:

$$\det(\mathbf{B} - \lambda\mathbf{E}) = \det(\mathbf{E} - \mathbf{P}^{-1}\mathbf{A} - \lambda\mathbf{E}) = -\det\mathbf{P}^{-1}\det(\mathbf{A} - \mathbf{P} + \lambda\mathbf{P}).$$

Таким образом, собственные значения $\mathbf{B}$ можно найти из уравнения

$$\det(\mathbf{A} - \mathbf{P} + \lambda\mathbf{P}) = 0.$$