Системы линейных алгебраических уравнений

Проверка свойств матриц

Вырожденность. Матрица A размера $n \times n$ вырождена, если ее определитель равен нулю ($\det(A) = 0$), или число обусловленности велико ($\text{cond}(A) > 10^{12}$), или ранг меньше размера матрицы ($\text{rank}(A) < n$).

Листинг 1. Пример. Проверка матрицы на вырожденность

```
% Число обусловленности
if cond(A) > 1e12 % эмпирический порог
    disp('Матрица близка к вырожденной');
end

% Определитель
if det(A) < 1e12
    disp('Матрица вырождена');
end

% Проверка ранга
if rank(A) < size(A,1)
    disp('Матрица вырождена');
end
```

Ортогональность. Матрица Q является ортогональной (или унитарной), если $Q^{\mathsf{T}}Q$ достаточно близка к единичной матрице.

Листинг 2. Пример. Проверка матрицы на ортогональность

```
Q = orth(rand(4));  % случайная ортогональная матрица

% корень из суммы квадратов всех элементов матрицы – норма Фробениуса
if norm(Q'*Q - eye(size(Q)), 'fro') < 1e-10
    disp('Матрица ортогональна');
else
    disp('Матрица не ортогональна');
end
```

Симметричность. Матрица A симметрична, если $A = A^{\mathsf{T}}$.

Листинг 3. Пример. Проверка матрицы на симметричность

```
% Проверка симметричности
A = [4, -2, 2; -2, 2, -4; 2, -4, 11];

% Способ 1: сравнение с транспонированной
if isequal(A, A')
    disp('Матрица симметрична');
else
    disp('Матрица не симметрична');
end
% Способ 2: с учётом погрешности
tol = 1e-10;
if norm(A - A', 'fro') < tol
    disp('Матрица симметрична в пределах погрешности');
else
    disp('Матрица не симметрична');
end
```

Положительная определенность и полуопределенность. Симметричная матрица A размера $n \times n$ является:

• положительно определенной, если все её собственные значения положительны:

$$\lambda_i > 0 \quad \text{для всех } i = 1, \ldots, n.$$

• положительно полуопределенной, если все её собственные значения неотрицательны:

$$\lambda_i \geq 0 \quad \text{для всех } i = 1, \ldots, n.$$

Листинг 4. Пример. Проверка матрицы на положительную определенность

```
if all(eig(A) > 0)
    disp('Матрица положительно определена');
else
    disp('Матрица не положительно определена');
end
```

Листинг 5. Пример. Проверка матрицы на положительную полуопределенность

```
tol = 1e-10;
if all(eig(A) >= -tol)
    disp('Матрица положительно полуопределена');
else
    disp('Матрица не является положительно полуопределенной');
end
```

Перед решением СЛАУ целесообразно проверить свойства матрицы для выбора оптимального способа решения.

Листинг 6. Пример. Проверка нескольких свойств перед решением

```
A = [4 -2 2; -2 2 -4; 2 -4 11];
b = [6; -10; 27];

opts = struct('SYM', false, 'POSDEF', false);

% Проверка симметричности
if norm(A - A', 'fro') < 1e-10
    opts.SYM = true;
    % Проверка положительной определенности
    [~, p] = chol(A);
    if p == 0
        opts.POSDEF = true;
    end
end

% Решение с учетом свойств
x = linsolve(A, b, opts);
```

В листинге 6 строка `[~, p] = chol(A)` обозначает попытку разложения Холецкого, если разложение успешно, то `p==0`, иначе `p` указывает на строку, где процесс прервался из-за неположительной определенности.

Установка `SYM = true` и `POSDEF = true` позволяет `linsolve` использовать разложение Холецкого, что значительно быстрее и устойчивее для симметричных положительно определенных систем.

Методы решения СЛАУ в MATLAB

В MATLAB существует множество способов решения систем линейных алгебраических уравнений (СЛАУ) вида $Ax = b$, где A — матрица коэффициентов, b — вектор правой части, x — искомый вектор. Выбор метода зависит от свойств матрицы A и требуемой точности.

1 Оператор обратной косой черты \

Самый универсальный и рекомендуемый способ. MATLAB автоматически выбирает наиболее эффективный алгоритм на основе типа матрицы — плотная, разреженная, треугольная, симметричная и т.д. Этот оператор обрабатывает также переопределенные и недоопределенные системы, выдавая решение в смысле наименьших квадратов.

1.1 Пример. Решение СЛАУ с плотной матрицей

Для плотной матрицы MATLAB использует LU-разложение с частичным выбором ведущего элемента. Плотная матрица — большинство элементов ненулевые.

Листинг 7. Пример. Решение СЛАУ с плотной матрицей

```
% Плотная матрица 5x5
A = [2 1 -1 0 0;
     1 3 1 0 0;
    -1 1 4 1 2;
     0 0 1 5 -1;
     0 0 2 -1 6];
b = [1; 2; 3; 4; 5];
% Используется LU разложение с частичным выбором ведущего элемента
x = A \ b; % решение
disp(x);
res = norm(A*x - b); % проверка невязки
fprintf('Невязка: %e\n', res);
```

1.2 Пример. Решение СЛАУ с симметричной положительно определенной матрицей

Для симметричных положительно определенных матриц используется разложение Холецкого. У положительно определенной матрицы все собственные значения строго положительны. Симметричная матрица — симметрична относительно главной диагонали, то есть совпадает с транспонированной.

Листинг 8. Пример. Решение СЛАУ с симметричной положительно определенной матрицей

```
C = randn(n); % квадратная невырожденная
A = C' * C; % симметричная положительно определенная матрица
eig(A) % собственные числа > 0
b = randn(n, 1) % столбец свободных членов
x = A \ b;
disp(x);
fprintf('Симметричная положительно определенная матрица:
        невязка = %e\n', norm(A*x-b));
```

1.3 Пример. Решение СЛАУ с треугольной матрицей

Если матрица является верхней или нижней треугольной, оператор \ выполняет прямую или обратную подстановку без разложения. Треугольная матрица – это квадратная матрица, у которой все элементы по одну сторону от главной диагонали равны нулю.

Листинг 9. Пример. Решение СЛАУ с треугольной матрицей

```
% Верхняя треугольная матрица
U = triu(randn(5,5));
b = randn(5,1);
% Решение для верхней треугольной матрицы
x = U \ b;
disp(x);
% Проверка
fprintf('Невязка: %e\n', norm(U*x - b));
```

1.4 Пример. Решение СЛАУ для разреженной матрицы

Для разреженных матриц MATLAB выбирает соответствующие разреженные алгоритмы, например, LU-разложение для разреженных матриц. Разреженная матрица — это матрица, в которой большинство элементов равны нулю, например, доля ненулевых элементов — 0.01 от общего количества или даже меньше.

Листинг 10. Пример. Решение СЛАУ для разреженной матрицы

```
% добавляем единичную, чтобы не была вырожденной
A = sprand(n, n, 0.1) + speye(n);
b = randn(n, 1);
x = A \ b;
fprintf('Разреженная матрица: невязка = %e\n', norm(A*x-b));
```

1.5 Пример. Решение переопределенной СЛАУ

Для переопределенных систем оператор \ выдает решение в смысле наименьших квадратов (минимизация $||Ax-b||_2$). Переопределенная система — это система уравнений, в которой количество уравнений больше количества неизвестных.

Листинг 11. Пример. Решение переопределенной СЛАУ

```
% Переопределенная система: 10 уравнений, 5 неизвестных
A = randn(10,5);
b = randn(10,1);
x = A \ b;
fprintf('Переопределенная система:
         норма невязки = %e\n', norm(A*x-b));
% Сравнение с решением через псевдообратную матрицу
x_pinv = pinv(A) * b;
fprintf('Отличие от псевдообратного решения: %e\n', norm(x - x_pinv));
```

1.6 Пример. Решение недоопределенной СЛАУ

Для недоопределенных систем оператор \ находит решение с минимальной евклидовой нормой. Недоопределенная система — это система уравнений, в которой количество неизвестных больше количества уравнений.

Листинг 12. Пример. Решение недоопределенной СЛАУ

```
% Недоопределенная система: 5 уравнений, 10 неизвестных
A = randn(5,10);
b = randn(5,1);
x = A \ b;
fprintf('Недоопределенная система: норма решения = %e\n', norm(x));
% Проверка, что решение удовлетворяет уравнениям
fprintf('Невязка: %e\n', norm(A*x-b));
```

2 Обратная матрица inv

Метод $x = A^{-1}b$ через функцию `inv` не рекомендуется из-за вычислительной нестабильности и избыточных затрат (вычисление полной обратной матрицы). Используется крайне редко, только для очень маленьких систем.

```
x = inv(A) * b;
```

3 LU-разложение

LU-разложение представляет матрицу A в виде произведения нижней треугольной L и верхней треугольной U:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{21} & 1 & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

Представление матрицы A в виде произведения нижней треугольной L и верхней треугольной U с перестановками:

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}}_{P} \cdot \underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \\ \ell_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \cdots & 1 \end{pmatrix}}_{L} \cdot \underbrace{\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}}_{U}$$

Решение СЛАУ сводится к двум треугольным системам:

1. Решить $Ly = Pb$ (прямая подстановка)
2. Решить $Ux = y$ (обратная подстановка)

Без явного указания матрицы перестановок можно использовать `lu(A)` с двумя выходными аргументами, но тогда L будет произведением нижней треугольной и перестановок, что может быть неудобно.

Листинг 13. Пример. Решение СЛАУ с помощью LU разложения

```
% Исходная матрица и правая часть
A = [2 -1 -2; -4 6 3; -4 -2 8];
b = [2; 4; 8];

% LU разложение с перестановками (PA = LU)
[L, U, P] = lu(A);
% Проверка: P*A = L*U
disp('P*A ='); disp(P*A);
disp('L*U ='); disp(L*U);

% Решение системы:
% 1) Решаем L*y = P*b - прямая подстановка
y = L \ (P*b);
% 2) Решаем U*x = y - обратная подстановка
x = U \ y;

disp('Решение x ='); disp(x); % вывод решения
% Проверка с помощью стандартного оператора \
x_check = A \ b;
disp('Решение через A\b ='); disp(x_check);
```

4 QR-разложение

Для прямоугольных или плохо обусловленных систем часто применяют QR-разложение, $A = QR$, где Q — ортогональная, R — верхняя треугольная:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}}_{A\ (m\times n)} = \underbrace{\begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{pmatrix}}_{Q\ (m\times n)} \cdot \underbrace{\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{pmatrix}}_{R\ (n\times n)},$$

$Q^{\mathsf{T}}Q = I_n,$ столбцы ортонормированы.

Решение находится как $x = R^{-1}Q^T b$:

```
[Q, R] = qr(A);
x = R \ (Q' * b);
```

MATLAB автоматически использует этот метод в операторе \ для переопределенных систем.

Листинг 14. Пример. Решение СЛАУ с помощью QR разложения

```
% Исходная матрица и правая часть
A = [2 -1 -2; -4 6 3; -4 -2 8];
b = [2; 4; 8];

% QR  разложение: A = Q*R
[Q, R] = qr(A);   % Q - ортогональная, R - верхняя треугольная

% Решение системы:
% 1) Умножаем обе части на Q^T:
%     Q^T * A * x = Q^T * b  =>  R * x = Q^T * b
% 2) Решаем R*x = c обратной подстановкой,  R - верхняя треугольная
c = Q' * b;
x = R \ c;  % оператор \ автоматически использует обратную подстановку

disp('Решение x ='); disp(x); % вывод решения
x_check = A \ b; % проверка с помощью стандартного оператора \
disp('Решение через A\b ='); disp(x_check);
```

5 Разложение Холецкого

Если матрица A симметрична и положительно определена, эффективно применить разложение Холецкого $A = R^T R$, где R — верхняя треугольная матрица:

$$A = R^{\mathsf{T}} R = \begin{pmatrix} r_{11} & 0 & \cdots & 0 \\ r_{12} & r_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r_{1n} & r_{2n} & \cdots & r_{nn} \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{pmatrix}$$

Решение сводится к двум треугольным системам:

```
R = chol(A);
x = R \ (R' \ b);
```

Для разреженных симметричных положительно определенных матриц используют `chol` с параметром `'lower'` или `'upper'`.

Листинг 15. Пример. Решение СЛАУ с помощью разложения Холецкого

```
% Исходная симметричная положительно определённая матрица
A = [4, -2, 2; -2, 2, -4; 2, -4, 11];
b = [6; -10; 27];

% 1. Разложение Холецкого: A = R' * R
R = chol(A); % возвращает верхнюю треугольную матрицу R
% 2. Решение системы через два этапа:
%    a) Решаем R' * y = b, прямая подстановка
y = R' \ b;
%    b) Решаем R * x = y, обратная подстановка
x = R \ y;

disp('Решение системы:');
disp(x); % Вывод результата
residual = norm(A*x - b); % вычисляем невязку
disp(['Невязка ||Ax - b|| = ', num2str(residual)]);
% Сравнение со стандартным оператором \
x_check = A \ b;
disp('Решение через A\b:');
disp(x_check);
```

6 Функция linsolve

Функция `linsolve` дает больше контроля над решением, позволяя указать свойства матрицы — симметричность, положительная определенность, треугольная форма и т.д., это может ускорить вычисления. Если свойства матрицы указаны неверно, результат может быть некорректным.

Листинг 16. Пример. Решение СЛАУ с помощью функции linsolve

```
% Исходная симметричная положительно определённая матрица
A = [4, -2, 2;
     -2, 2, -4;
     2, -4, 11];
b = [6; -10; 27];

% Настройка структуры опций
opts.SYM = true;    % матрица симметричная
opts.POSDEF = true; % матрица положительно определённая

% Решение системы с учётом свойств
x = linsolve(A, b, opts);

% Вывод результата
disp('Решение системы:');
disp(x);

% Проверка невязки
residual = norm(A*x - b);
disp(['Невязка ||Ax - b|| = ', num2str(residual)]);
```

Обычный оператор $A \setminus b$ для симметричных положительно определенных матриц также использует разложение Холецкого, но `linsolve` дает больший контроль и может быть чуть эффективнее при многократном решении с разными правыми частями (если использовать `opts.UT = true` и т.п.). Если свойства матрицы известны заранее, их указание исключает лишние проверки и выбор метода.

Контрольные вопросы

1. Каков основной и наиболее универсальный способ решения СЛАУ в MATLAB?

2. Какие алгоритмы автоматически выбирает оператор \ в зависимости от свойств матрицы?

3. Почему вычисление решения через `inv(A)*b` не рекомендуется? В каких случаях его все же можно применять?

4. Опишите принцип LU-разложения. Как с его помощью решить систему $Ax = b$?

5. Для каких систем обычно применяют QR-разложение? Как связан оператор \ с QR-разложением при решении переопределенных систем?

6. Какие требования предъявляются к матрице для использования разложения Холецкого? Запишите последовательность команд для решения СЛАУ с его помощью.

7. В чем преимущество функции `linsolve` перед оператором \? Какие структуры опций можно задать?

8. Как оператор \ обрабатывает переопределенные и недоопределенные системы? Какое решение получается в каждом случае?

9. Какие методы MATLAB использует для треугольных и разреженных матриц при использовании оператора \?

10. Предположим, необходимо решить несколько систем с одной и той же матрицей A, но разными правыми частями b. Какой подход позволит сделать это наиболее эффективно? Приведите пример с использованием LU-разложения.

Пример. Сравнение скорости получения решений

```
clc;clear;
% Генерация симметричной положительно определённой матрицы
n = 2000; % размерность
A = randn(n); % случайная матрица
% сдвиг, гарантирующий положительную определённость
A = A' * A + 10*eye(n);
b = randn(n,1); % правая часть
% 1. Стандартный оператор \
tic;
x1 = A \ b;
t1 = toc;
fprintf('||A*b-x||: %e\n', norm(A*b-x1));

% 2. Явное разложение Холецкого
tic;
R = chol(A); % верхняя треугольная матрица
y = R' \ b; % прямая подстановка: R' * y = b
x2 = R \ y; % обратная подстановка: R * x = y
t2 = toc;
fprintf('||x1-x2||: %e\n', norm(x1-x2));

% 3. linsolve с указанием свойств матрицы
opts.SYM = true;
opts.POSDEF = true;
tic;
x3 = linsolve(A, b, opts);
t3 = toc;
fprintf('||x1-x3||: %e\n', norm(x1-x3));

% Сравнение времени
fprintf('1. A\\b : %f сек\n', t1);
fprintf('2. chol : %f сек\n', t2);
fprintf('3. linsolve : %f сек\n', t3);
```

Задание 1. Решение СЛАУ методом Холецкого

Решить методом Холецкого СЛАУ $Ax = b$ с симметричной положительно определенной матрицей A для десяти различных правых частей b. Вычислить невязку для каждого полученного решения. Измерить общее время решения всех десяти систем.

При решении обратите внимание на то, что разложение Холецкого нужно будет выполнить один раз для всех правых частей:

```
R = chol(A);
x = R \ (R' \ b);
```

Ниже в таблице 1 приведены варианты параметров: для каждого варианта заданы диапазон $[m, k]$ генерации целых случайных чисел, используемых для формирования элементов матрицы B и компонент векторов правых частей, а также размер n квадратной матрицы A.

Используйте формулу $A = B^T B + \delta I$ для формирования симметричной положительно определенной матрицы. Здесь B — случайная квадратная матрица целых чисел размера $n \times n$, δ — вещественное число, вычисленное в MATLAB по формуле: `delta=mean(diag(B'*B))`, I — единичная матрица.

Решите все СЛАУ с помощью `\` обратной косой черты. Сравните время решений, полученных первым и вторым методом. Сделайте выводы.

Таблица 1. Варианты для заданий 1-4

№	m, k	n	№	m, k	n	№	m, k	n	№	m, k	n
1	$-5, 5$	100	**14**	$0, 10$	250	**27**	$-4, 4$	400	**40**	$-7, 7$	100
2	$-10, 10$	150	**15**	$-20, 20$	300	**28**	$1, 6$	450	**41**	$-4, 4$	150
3	$0, 5$	200	**16**	$1, 100$	350	**29**	$-5, 5$	500	**42**	$1, 6$	200
4	$-3, 3$	250	**17**	$-50, 50$	400	**30**	$-10, 10$	100	**43**	$-5, 5$	250
5	$-2, 2$	300	**18**	$0, 5$	450	**31**	$0, 10$	150	**44**	$-10, 10$	300
6	$-1, 1$	350	**19**	$-3, 3$	500	**32**	$-20, 20$	200	**45**	$0, 10$	350
7	$0, 1$	400	**20**	$-2, 2$	100	**33**	$1, 100$	250	**46**	$-20, 20$	400
8	$-7, 7$	450	**21**	$-1, 1$	150	**34**	$-50, 50$	300	**47**	$1, 100$	450
9	$-4, 4$	500	**22**	$0, 1$	200	**35**	$0, 5$	350	**48**	$-50, 50$	500
10	$1, 6$	100	**23**	$-7, 7$	250	**36**	$-3, 3$	400	**49**	$0, 5$	100
11	$-5, 5$	150	**24**	$-4, 4$	300	**37**	$-2, 2$	450	**50**	$-3, 3$	150
12	$-10, 10$	200	**25**	$1, 6$	350	**38**	$-1, 1$	500	**51**	$-2, 2$	200
13	$0, 5$	250	**26**	$-5, 5$	400	**39**	$0, 1$	100	**52**	$-1, 1$	250

Задание 2. Решение СЛАУ методом QR разложения

Решить методом QR разложения СЛАУ $Ax = b$ с плохо обусловленной матрицей A для десяти различных правых частей b. Вычислить невязку для каждого полученного решения. Измерить общее время решения всех десяти систем.

При решении обратите внимание на то, что QR разложение нужно будет выполнить один раз для всех правых частей:

```
[Q, R] = qr(A);
x = R \ (Q' * b);
```

В таблице 1 приведены варианты параметров: для каждого варианта заданы диапазон $[m, k]$ генерации целых случайных чисел, используемых для формирования элементов матрицы A и компонент векторов правых частей b, а также размер n квадратной матрицы A. Используйте команду MATLAB создания матрицы Гильберта `hilb(n)` для формирования плохо обусловленной матрицы, здесь `n` — размер матрицы.

Решите все СЛАУ с помощью `\` обратной косой черты. Сравните время решений, полученных первым и вторым методом. Сделайте выводы.

Задание 3 Решение прямоугольных СЛАУ

Решить методом QR разложения переопределенную СЛАУ для десяти различных правых частей. Вычислить невязку для каждого полученного решения. Измерить общее время решения всех десяти систем.

Ниже в таблице 1 приведены варианты параметров: для каждого варианта заданы диапазон $[m, k]$ генерации целых случайных чисел, используемых для формирования элементов матрицы и компонент векторов правых частей b, а также целое число n для задания размера матрицы: $n + 2$ — число строк матрицы, n — число столбцов матрицы.

Выполните решение недоопределенных СЛАУ, когда n — число строк матрицы, $n + 2$ — число столбцов матрицы. Вычислите невязку для каждого полученного решения. Измерьте общее время решения всех десяти систем.

Задание 4. Решение СЛАУ методом LU разложения

Решить методом LU разложения СЛАУ $Ax = b$ для десяти различных правых частей b. Вычислить невязку для каждого полученного решения. Измерить общее время решения всех десяти систем.

При решении обратите внимание на то, что LU разложение нужно будет выполнить один раз для всех правых частей:

```
% LU разложение с перестановками (PA = LU)
[L, U, P] = lu(A);
%  прямая подстановка
y = L \ (P*b);
% обратная подстановка
x = U \ y;
```

В таблице 1 приведены варианты параметров: для каждого варианта заданы диапазон $[m, k]$ генерации целых случайных чисел, используемых для формирования элементов матрицы A и компонент векторов правых частей b, а также размер n квадратной матрицы A.

Решите все СЛАУ с помощью \ обратной косой черты. Сравните время решений, полученных первым и вторым методом. Сделайте выводы.