Лабораторная работа №1. (часть2)
Прямые методы решения больших разреженных СЛАУ MatLab

БАЗА (0) Применим метод сопряженных градиентов **pcg** к полностью заполненной симметричной положительно-определенной (ПО) матрице (**A**). Поменяем точность, число итераций, начальное приближение. Проследим за значением выходных параметров

1. Задайте точку отсчета случайных чисел	`rng default`

2. Создайте СЛАУ: случайную полностью заполненную симметричную ПО матрицу размерности 10x10, вектор решений (можно единичный), вычислите правую часть	`A = rand...`
	`A=A*A'`
	`X_t=ones...`
	`C = ...`

3. Вычислите решение СЛАУ
Метод не сошелся за 10 итераций с точностью 1e-6, и остановился на точности 2.3e-5
Число итераций – минимальное число из n и 20
	`X0 = pcg(A,C)`
	`pcg stopped at iteration 10 without converging to the desired tolerance 1e-06 because the maximum number of iterations was reached.`
	`The iterate returned (number 10) has relative residual 2.3e-05`

4. Увеличьте максимальное количество итераций до 15 и погрешность до 1e-7	`X1 = pcg(A,C,1e-7,15)` `pcg converged at iteration 11 to a solution with relative residual 4.6e-11.`
5. Проверьте точность по формуле `norm(b-A*x)/norm(b)`	`er = norm...` `er1 = norm...`
6. Задайте больше выходных параметров **flag** – описание сходимости метода **0** – метод сошелся с заданными входными параметрами **1** – число итераций = **maxit**, точность не достигнута **rr2** – текущая достигнутая точность по формуле **it2** – номер итерации для вычисленного **x** **rv2** – вектор норм невязок на каждой итерации	`tol = 1e-7;  maxit = 15;` `[x2,flag,rr2,it2,rv2] = pcg(A,b,tol,maxit)`
7. Задайте начальное приближение **x0**	`[x3] = pcg(A,C,1e-6,[],[],[],x0)`

МИНИМУМ (+1) Применение метода pcg для разреженной матрицы и матрицы, нарушающей условия сходимости

1. Проделать все действия для случайной разреженной (плотность 0.5) симметричной ПО матрицы размерностью 10x10 – матрица **B**

2. Применить метод для случайной матрицы не являющейся симметричной ПО (случай матрицы **A** и матрицы **B**). Вывести параметр **flag** и нормы невязок

flag = 4 - одна из величин в процессе решения вышла за пределы допустимых величин чисел

ДОСТАТОЧНО (+1) Исследовать зависимость количества итераций от числа обусловленности матрицы при одной и той же заданной точности

1. Создайте матрицы A и B (плотность 0,3) размерностью 1000x1000 с числом обусловленности 1000, единичный вектор решений xt и вычислите правую часть Ca и Cb

`N = 1000`
`[Q, R] = qr(rand(N));`
`D = diag(1:N)`
`A = Q'*D*Q;`
	`B = sprandsym(100,0.3,1e-3,1);`

`xt = ones(N,1);` `Ca = A*xt;`	`Cb = B*xt`

2. Примените метод **pcg** с точностью 1e-7 и выходными параметрами **flag** и **iter**.

```
[xA,flagA,relresA,iterA] = pcg(A, Ca, 1e-7)
[xB,flagB,relresB,iterB] = pcg(B, Cb, 1e-7)
```

3. Увеличьте число итераций до 200. Убедитесь, что метод сошелся для обеих матриц. Если нет – увеличьте максимальное число итераций

```
[xA,flagA,relresA,iterA] = pcg(A, Ca, 1e-7, 200)
[xB,flagB,relresB,iterB] = pcg(в, Cb, 1e-7, 200)
```

4. Создайте цикл для изменения числа обусловленности матриц от 1 до 1e10 и постройте график зависимости числа итераций от числа обусловленности для обеих матриц

МАКСИМУМ (+1) Применение метода сопряженных градиентов с предобуславливателем

1. Создать СЛАУ с пятидиагональной матрицей, которая получается при решении эллиптического уравнения методом конечных разностей.

2. Вычислить предобуславливатель с помощью неполного разложения Холецкого.

3. Применить предообуславливатель в методе сопряженных градиентов.

4. Построить график зависимости количества итераций от порога неполного разложения

1. Создание СЛАУ. Матрица в самом простом случае выглядит так

```
 4 -1  0 -1  0  0  0  0  0
-1  4 -1  0 -1  0  0  0  0
 0 -1  4  0  0 -1  0  0  0
-1  0  0  4 -1  0 -1  0  0
 0 -1  0 -1  4 -1  0 -1  0
 0  0 -1  0 -1  4  0  0 -1
 0  0  0 -1  0  0  4 -1  0
 0  0  0  0 -1  0 -1  4 -1
 0  0  0  0  0 -1  0 -1  4
```

Блочно трехдиагональная. Блоки на диагонали: трехдиагональная матрица с 4 на диагонали и -1 на побочных диагоналях. Блоки около диагонали: единичные матрицы с обратным знаком
Создать матрицу 4x4 блока. Визуализировать и проверить правильность.
Создать вектор точных решений (длиной 4*4=16!)
Вычислить правую часть СЛАУ

2. При неполном разложении Холецкого (**`ichol`**) вычисляется нижняя(!) треугольная матрица **`L`** такая, что произведение **`LL'`** со спектральными свойствами близкими к исходной матрице. В неполном разложении Холецкого можно ввести порог, который ограничивает количество новых ненулевых элементов треугольной матрицы разложения, появляющихся в процессе разложения. Именно, слишком маленькие элементы заменяются нулями. Получается, что значение порога также управляет разреженностью множителей.
Сравнить с помощью spy матрицы
– исходную (**`A`**)
– разложения Холецкого (**`chol(A)`**)
– неполного разложения Холецкого без порога (**`ichol(A) = ichol(A, struct('type', 'nofill') )`**
– с нулевым порогом **`ichol(A, struct('type','ict','droptol',0) )`**
– с ненулевым порогом **`ichol(A, struct('type','ict','droptol',0.1) )`**
– с порогом =1 **`ichol(A, struct('type','ict','droptol',1) )`**
Какие из полученных матриц оказались одинаковыми?

3.

4. Создать матрицу 400 x 400
Создать вектор точных решений (длиной 400*400)
Вычислить правую часть СЛАУ

pcg(A,b,tol,maxit,M) и pcg(A,b,tol,maxit,M1,M2) – при решении используется матрица предусловий M или M=M1*M2, так что производится решение системы inv(M)*A*x=inv(M)*b относительно x. Если M1 или M2 – пустые матрицы, то они рассматриваются как единичные матрицы, что эквивалентно отсутствию входных условий вообще.
pcg(A,B,tol,maxit,M1,M2,X0) – точно задается начальное приближение X0. Если X0 – пустая матрица, то по умолчанию используется вектор, состоящий из нулей.
X = pcg(A,B,tol,maxit,M1,M2,X0) – при наличии единственного выходного параметра возвращает решение X. Если метод pcg сходится, выводится соответствующее сообщение. Если метод не сходится после максимального числа итераций или по другой причине, на экран выдаются относительный остаток norm(B–A*X)/norm(B) и номер итерации, на которой метод остановлен.
[X,flag] = pcg (A,X,tol,maxit,M1,M2,X0) – возвращает решение X и флаг flag, описывающий сходимость метода. Значения flag предоставляют следующие данные о сходимости решения:
• 0 – решение сходится при заданной точности tol и числе итераций не более заданного maxit;
• 1 – число итераций равно заданному maxit, но сходимость не достигнута;
• 2 – матрица предусловий M плохо обусловлена;
• 3 – процедура решения остановлена, поскольку две последовательные оценки решения оказались одинаковыми;
• 4 – одна из величин в процессе решения вышла за пределы допустимых величин чисел (разрядной сетки компьютера).

2.6 Привести пример, когда количества итераций по умолчанию не хватает для достижения заданной точности, например, такой:

```
N = 1000; % задаем размер матрицы
D = diag(1:N); % генерируем матрицу с числом обусловленности 1000
[Q, R] = qr(rand(N)); % получаем ортогональную матрицу Q для построения матрицы с заданным числом обусловленности
A = Q'*D*Q; % строим заполненную матрицу с числом обусловленности 1000
xt = ones(N,1); % задаем точное решение
b = A*xt; % вычисляем соответствующую xt правую часть СЛАУ
[x,flag,relres,iter] = pcg(A, b) % запускаем метод сопряженных градиентов
```

Увеличить количество итераций для получения flag = 0:

```
[x,flag,relres,iter] = pcg(A, b, [], 150)
```

2.7 Исследовать зависимость количества итераций в pcg от числа обусловленности матрицы, например, так:

```
N = 1000; % задаем размер матрицы
CN = linspace(1, 1e10, 10); % генерируем массив чисел обусловленности тестовых матриц
[Q, R] = qr(rand(N)); % получаем ортогональную матрицу Q для построения матрицы с заданным числом обусловленности
xt = ones(N, 1); % заполняем вектор точного решения СЛАУ
ITER = zeros(1, 10); % готовим массив для записи количества итераций pcg
% цикл по матрицам с увеличивающимся числом обусловленности
for k = 1:length(CN)
    D = diag(linspace(1, CN(k), N)); % заполняем диагональную матрицу с числом обусловленности CN(k)
    A = Q'*D*Q; % генерируем матрицу с числом обусловленности CN(k)
    b = A*xt; % вычисляем правую часть СЛАУ, соответствующую точному решению xt
    [x,flag,relres,iter] = pcg(A, b, 1e-10, 1000); % вызываем метод сопряженных градиентов
    flag
    ITER(k) = iter; % записываем количество итераций в массив
end
% строим график зависимости количества итераций от числа обусловленности
figure
semilogx(CN, ITER)
```

[X,flag,relres] = pcg (A,X,tol,maxit,M1,M2,X0) – также возвращает относительную вторую норму вектора остатков relres=norm(B–A*X)/norm(B). Если флаг flag равен 0, то relres tol.
[X,flag,relres,iter] = pcg(A,B,tol,maxit,M1,M2,X0) – также возвращает номер итерации, на которой был вычислен X. Значение iter всегда удовлетворяет условию 0 < iter < maxit.
[X,flag,relres,iter,resvec] = pcg A,B,tol,maxit,M1,M2, X0) – также возвращает вектор вторых норм остатков resvec для каждой итерации, начиная с resvec(1)=norm(B–A*X0). Если флаг flag равен 0, то resvec имеет длину iter+1 и resvec(end) tol*norm(B). Возможны значения flag, равные 0, 1, 2, 3 и 4.

6. Повторить пример из лекции использования предобусловливателя, полученного при помощи неполного разложения Холецкого, для решения СЛАУ с пятидиагональной матрицей, получающейся при решении эллиптического уравнения методом конечных разностей.

```
% конструирование матрицы для 5-ти точечного разностного оператора Лапласа
n = 100; I = ones(n,1); D = spdiags(-I, 0, n, n); T = spdiags([I -2*I I], [-1 0 1], n, n);
```

```
AN = kron(D, T) + kron(T, D);
% создание правой части системы для точного решения [1, 2, 3, ...]
x_ex = (1:n*n)'; b = AN*x_ex;
% в цикле изменяем порог droptol в неполном разложении Холецкого R'R = AN
% и используем факторизованный предобусловливатель R'R
DROPTOL = logspace(-6, 0, 20); ITER = [];
for droptol = DROPTOL
R = ichol(AN, droptol);  % Incomplete Cholesky factorization
[x, flag, relres, iter] = pcg(AN, b, 1e-10, 1000, R', R);
ITER = [ITER iter];
end
figure('Color', 'w'), loglog(DROPTOL, ITER)
xlabel(' droptol '), ylabel('Количество итераций')
% сопряженные градиенты без предобусловливания
[x, flag, relres, iter] = pcg(AN, b, 1e-10, 1000);
hold all
loglog([DROPTOL(1) DROPTOL(end)], [iter iter])
grid on
        legend('с предобусловливателем', 'без предобусловливателя')
```