Equation Chapter 1 Section 1Новосибирский государственный технический университет

МЕТОДЫ РЕШЕНИЯ БОЛЬШИХ СИСТЕМ УРАВНЕНИЙ

Методические указания к индивидуальным занятиям по курсу "Методы решения больших систем уравнений" для студентов 5-6 курсов ФПМИ, направление 010500

Составители: ассист. *П.А. Домников*,
д-р техн. наук, проф. *М.Г. Персова*,
д-р техн. наук, проф. *Ю.Г. Соловейчик*

Рецензент: д-р техн. наук, проф. *М.Э. Рояк*

Работа подготовлена на кафедре прикладной математики

Цель работы

Разработать программы решения СЛАУ с комплексными матрицами. Изучить особенности схем предобусловливания с заменой скалярного произведения. Исследовать влияние схем предобусловливания и сглаживания невязки на сходимость итерационных методов на матрицах большой размерности. Приобрести навыки работы с библиотеками для высокопроизводительных вычислений. Выполнить распараллеливание итерационных методов решения СЛАУ с разреженными матрицами в вычислительных системах с общей памятью. Оценить ускорение, получаемое за счет распараллеливания.

Теоретическая часть

КОМПЛЕКСНЫЕ МАТРИЦЫ И ВЕКТОРЫ

Поле комплексных чисел можно рассматривать как поле действительных матриц вида

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}, \quad a,b \in \mathbb{R}, \tag{1}$$

в котором роль действительных чисел играют матрицы вида $\begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}$, роль мнимой единицы – матрица $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ [3, с.132]. Таким образом, комплексные матрицы можно рассматривать как вещественные, состоящие из блок-элементов вида (1).

Комплексные векторы также можно рассматривать как вещественные: вектору $x \in \mathbb{C}^n$ будет соответствовать вектор $x' \in \mathbb{R}^{2n}$, причем $x' = (x_1^{\mathrm{Re}}, x_1^{\mathrm{Im}}, x_2^{\mathrm{Re}}, x_2^{\mathrm{Im}}, ..., x_n^{\mathrm{Re}}, x_n^{\mathrm{Im}})$.

Пусть $x, y \in \mathbb{C}^n$. Скалярное произведение двух комплексных векторов задается соотношением:

$$(x,y)=\sum_{k=1}^{n}\overline{x}_k y_k=\sum_{k=1}^{n}(x_k^{\mathrm{Re}}-ix_k^{\mathrm{Im}})(y_k^{\mathrm{Re}}+iy_k^{\mathrm{Im}})= \\ =\sum_{k=1}^{n}\Big((x_k^{\mathrm{Re}}y_k^{\mathrm{Re}}+x_k^{\mathrm{Im}}y_k^{\mathrm{Im}})+i(x_k^{\mathrm{Re}}y_k^{\mathrm{Im}}-x_k^{\mathrm{Im}}y_k^{\mathrm{Re}})\Big)\in\mathbb{C}. \tag{2}$$

Горизонтальная черта означает комплексное сопряжение. Выражение вида $(\overline{x},y)$ вычисляется по формуле:

$$(\overline{x},y)=\sum_{k=1}^{n}\overline{\overline{x}}_k y_k=\sum_{k=1}^{n}x_k y_k=\sum_{k=1}^{n}(x_k^{\mathrm{Re}}+ix_k^{\mathrm{Im}})(y_k^{\mathrm{Re}}+iy_k^{\mathrm{Im}})= \\ =\sum_{k=1}^{n}\Big((x_k^{\mathrm{Re}}y_k^{\mathrm{Re}}-x_k^{\mathrm{Im}}y_k^{\mathrm{Im}})+i(x_k^{\mathrm{Re}}y_k^{\mathrm{Im}}+x_k^{\mathrm{Im}}y_k^{\mathrm{Re}})\Big)\in\mathbb{C}. \tag{3}$$

Билинейная форма $[x,y]=(\overline{x},y)$, определяемая соотношением (3), не является скалярным произведением.

Если рассматривать векторы $x\in\mathbb{C}^n$ и $y\in\mathbb{C}^n$ как вещественные, то можно определить операцию скалярного произведения над соответствующими векторами в вещественном евклидовом пространстве. Пусть $x'=(x_1^{\mathrm{Re}},x_1^{\mathrm{Im}},x_2^{\mathrm{Re}},x_2^{\mathrm{Im}},...,x_n^{\mathrm{Re}},x_n^{\mathrm{Im}})\in\mathbb{R}^{2n}$ и $y'=(y_1^{\mathrm{Re}},y_1^{\mathrm{Im}},y_2^{\mathrm{Re}},y_2^{\mathrm{Im}},...,y_n^{\mathrm{Re}},y_n^{\mathrm{Im}})\in\mathbb{R}^{2n}$, тогда

$$(x',y')=\sum_{k=1}^{n}(x_k^{\mathrm{Re}}y_k^{\mathrm{Re}}+x_k^{\mathrm{Im}}y_k^{\mathrm{Im}})\in\mathbb{R}. \tag{4}$$

Для решения разреженных СЛАУ большой размерности с блочной структурой (1) можно применить итерационные методы, предназначенные для решения СЛАУ с вещественными несимметричными матрицами, такие как GMRES [1, с.885; 2, с.164], BCG [1, с.885; 2, с.222], локально-оптимальная схема [1, с.880]. Однако, существуют специальные итерационные методы, предназначенные для решения разреженных СЛАУ с комплексно-симметричными и комплексно-несимметричными матрицами.

CONJUGATE ORTHOGONAL CONJUGATE GRADIENT METHOD (COCG)

Данный метод может быть рассмотрен как обобщение метода сопряженных градиентов для системы вида

$$Ax = b, \tag{5}$$

где комплексная матрица A размера $n \times n$ симметричная $(A = A^T)$, но, в общем случае, может не быть эрмитовой $(A \neq \overline{A}^T)$ [6]. Алгоритм метода может быть записан следующим образом.

$$\text{Выбрать } x^0. \text{ Вычислить } r^0 = b - Ax^0, p^0 = r^0. \tag{6}$$

$$\text{Для } j = 0,1,2,...$$

$$\alpha_j = \frac{(\overline{r}^j, r^j)}{(\overline{Ap}^j, p^j)}, \tag{7}$$

$$x^{j+1} = x^j + \alpha_j p^j, \tag{8}$$

$$r^{j+1} = r^j - \alpha_j A p^j, \tag{9}$$

$$\beta_j = \frac{(\overline{r}^{j+1}, r^{j+1})}{(\overline{r}^j, r^j)}, \tag{10}$$

$$p^{j+1} = r^{j+1} + \beta_j p^j, \tag{11}$$

где x^0 – вектор начального приближения; x^j – вектор решения на *j*-й (текущей) итерации; r^j – вектор невязки на *j*-й (текущей) итерации; p^j – вектор спуска на *j*-й (текущей) итерации; $\alpha_j, \beta_j \in \mathbb{C}$ – коэффициенты. Выражения вида $(\overline{x}, y)$ в (7) и (10) вычисляются по формуле (3).

Рассмотрим схему метода COCG применительно к предобусловленной системе $M^{-1}Ax = M^{-1}b$. Нетрудно показать, что при замене естественного скалярного произведения на $(x, y)_M = (Mx, y)$ [2, с.263; 3, с.451] итерационный процесс (6)-(11) можно представить в виде:

$$\text{Выбрать } x^0. \text{ Вычислить } r^0 = b - Ax^0,\ z^0 = M^{-1}r^0,\ p^0 = z^0. \tag{12}$$

Для $j = 0,1,2,...$

$$\alpha_j = \frac{(\overline{r}^j, z^j)}{(\overline{Ap}^j, p^j)}, \tag{13}$$

$$x^{j+1} = x^j + \alpha_j p^j, \tag{14}$$

$$r^{j+1} = r^j - \alpha_j A p^j, \tag{15}$$

$$z^{j+1} = M^{-1} r^{j+1}, \tag{16}$$

$$\beta_j = \frac{(\overline{r}^{j+1}, z^{j+1})}{(\overline{r}^j, z^j)}, \tag{17}$$

$$p^{j+1} = z^{j+1} + \beta_j p^j, \tag{18}$$

где вектор z^j - вектор невязки предобусловленной системы на *j*-й (текущей) итерации. В алгоритме (12)-(18) на каждой итерации требуется выполнять только одно матрично-векторное умножение и одно решение СЛАУ с матрицей предобусловливания M. В векторах x^j и r^j хранится соответственно решение и невязка исходной (не предобусловленной) СЛАУ на *j*-й (текущей) итерации.

CONJUGATE A-ORTHOGONAL CONJUGATE RESIDUAL METHOD (COCR)

Кроме адаптации метода сопряженных градиентов к решению СЛАУ с комплексно-симметричными матрицами существуют и аналогичные адаптации других итерационных методов. Приведем алгоритм метода COCR [5], являющегося обобщением метода сопряженных невязок для систем с комплексно-симметричными матрицами.

Выбрать x^0. Вычислить $r^0 = b - Ax^0$, $p^0 = r^0$, $z^0 = Ap^0$, (19)

Для $j = 0,1,2,...$

$$\alpha_j = \frac{(\overline{r}^j, Ar^j)}{(\overline{z}^j, z^j)}, \tag{20}$$

$$x^{j+1} = x^j + \alpha_j p^j, \tag{21}$$

$$r^{j+1} = r^{j+1} - \alpha_j z^j, \tag{22}$$

$$\beta_j = \frac{(\overline{r}^{j+1}, Ar^{j+1})}{(\overline{r}^j, Ar^j)}, \tag{23}$$

$$p^{j+1} = r^{j+1} + \beta_j p^j, \tag{24}$$

$$z^{j+1} = Ar^{j+1} + \beta_j z^j, \tag{25}$$

где $z^j = Ap^j$ – вспомогательный вектор, который вычисляется не умножением матрицы на вектор, а пересчитывается рекуррентно по формуле (25).

Схема метода COCR с матрицей предобусловливания M (при применении ее к СЛАУ с матрицей $M^{-1}A$ и при замене естественных скалярных произведений на $(x,y)_M = (Mx,y)$) выглядит следующим образом.

Выбрать начальное приближение x^0 и положить

$$r^0 = b - Ax^0,\ p^0 = s^0 = M^{-1}r^0,\ a^0 = z^0 = As^0,\ w^0 = M^{-1}z^0. \tag{26}$$

Для $j = 0,1,2,...$

$$\alpha_j = \frac{(\overline{a}^j, s^j)}{(\overline{z}^j, w^j)}, \tag{27}$$

$$x^{j+1} = x^j + \alpha_j p^j, \tag{28}$$

$$r^{j+1} = r^j - \alpha_j z^j, \tag{29}$$

$$s^{j+1} = s^j - \alpha_j w^j, \tag{30}$$

$$a^{j+1} = As^{j+1}, \tag{31}$$

$$\beta_j = \frac{(\overline{a}_{j+1}, s_{j+1})}{(\overline{a}_j, s_j)}, \tag{32}$$

$$p^{j+1} = s^{j+1} + \beta^j p^j, \tag{33}$$

$$z^{j+1} = a^{j+1} + \beta_j z^j, \tag{34}$$

$$w^{j+1} = M^{-1} z^{j+1}, \tag{35}$$

где вектор s^j – вектор невязки предобусловленной системы на *j*-й (текущей) итерации; $a^j = As^j$, $z^j = Ap^j$, $w^j = M^{-1}z^j$ – вспомогательные векторы, причем вектор $z^j = Ap^j$, как и ранее, вычисляется не умножением матрицы на вектор, а пересчитывается рекуррентно по формуле (34).

КОМЛЕКСНАЯ ЛОКАЛЬНО ОПТИМАЛЬНАЯ СХЕМА

Рассмотрим модификацию локально-оптимальной схемы (ЛОС) для решения СЛАУ (5) с комплексно-симметричной матрицей A.

Выбрать x^0. Вычислить $r^0 = b - Ax^0$, положить $p^0 = r^0$, $z^0 = Ap^0$. (36)

Для $j = 0,1,2,...$

$$\alpha_j = \frac{(\overline{z}^j, r^j)}{(\overline{z}^j, z^j)}, \tag{37}$$

$$x^{j+1} = x^j + \alpha_j p^j, \tag{38}$$

$$r^{j+1} = r^j - \alpha_j z^j, \tag{39}$$

$$\beta_j = -\frac{(\overline{z}^j, Ar^{j+1})}{(\overline{z}^j, z^j)}, \tag{40}$$

$$p^{j+1} = r^{j+1} + \beta_j p^j, \tag{41}$$

$$z^{j+1} = Ar^{j+1} + \beta_j z^j, \tag{42}$$

где $z^j = Ap^j$ – вспомогательный вектор, который вычисляется не умножением матрицы на вектор, а пересчитывается рекуррентно по формуле (42).

При использовании матрицы предобусловливания M и замене естественного скалярного произведения на $(x,y)_M = (Mx,y)$ локально-оптимальная схема (36)-(42) (примененная к СЛАУ с матрицей $M^{-1}A$) преобразуется к следующему виду.

Выбрать начальное приближение x^0 и положить

$$r^0 = (b - Ax^0),\ p^0 = s^0 = M^{-1}r^0,\ z^0 = a^0 = Ap^0,\ w^0 = M^{-1}z^0, \tag{43}$$

Для $j = 0,1,2,...$

$$\alpha_j = \frac{(\overline{w}^j, r^j)}{(\overline{w}^j, z^j)}, \tag{44}$$

$$x^{j+1} = x^j + \alpha_j p^j, \tag{45}$$

$$r^{j+1} = r^j - \alpha_j z^j, \tag{46}$$

$$s^{j+1} = s^j - \alpha_j w^j, \tag{47}$$

$$a^{j+1} = As^{j+1}, \tag{48}$$

$$\beta_j = -\frac{(\overline{w}^j, a^{j+1})}{(\overline{w}^j, z^j)}, \tag{49}$$

$$p^{j+1} = s^{j+1} + \beta_j p^j, \tag{50}$$

$$z^{j+1} = a^{j+1} + \beta_j z^j, \tag{51}$$

$$w^{j+1} = M^{-1}z^{j+1}, \tag{52}$$

где вектор s^j – вектор невязки предобусловленной системы на *j*-й (текущей) итерации; $a^j = As^j$, $z^j = Ap^j$, $w^j = M^{-1}z^j$ – вспомогательные векторы, причем вектор $z^j = Ap^j$, как и ранее, вычисляется не умножением матрицы на вектор, а пересчитывается рекуррентно по формуле (51).

Также существует вариант локально-оптимальной схемы, который можно применить и для решения СЛАУ с комплексно-несимметричными матрицами. В отличие от варианта, адаптированного для решения СЛАУ с комплексно-симметричными матрицами, в данном случае в схеме метода вместо билинейной формы (3) используются комплексные скалярные произведения, определяемые формулой (2). Таким образом, коэффициенты α_j и β_j в схеме (36)-(42) будут вычисляться соответственно по формулам

$$\alpha_j = \frac{(w^j, r^j)}{(w^j, z^j)}, \tag{53}$$

$$\beta_j = -\frac{(z^j, Ar^{j+1})}{(z^j, z^j)}, \tag{54}$$

а в схеме (43)-(52) с предобусловливанием – по формулам

$$\alpha_j = \frac{(w^j, r^j)}{(w^j, z^j)}, \tag{55}$$

$$\beta_j = -\frac{(w^j, a^{j+1})}{(w^j, z^j)}. \tag{56}$$

GMRES ДЛЯ КОМПЛЕКСНЫХ СЛАУ

Адаптация метода GMRES для комплексных СЛАУ может быть получена из алгоритма метода GMRES для вещественных СЛАУ [1, с.885; 2, с.164] путем замены скалярных произведений на комплексные скалярные произведения, определяемые формулой (2). Еще одно важное отличие состоит в том, что в комплексном случае матрица $\mathbf{H}$ приводится к верхнетреугольному виду с использованием следующей матрицы вращения [2, с.184]:

$$\mathbf{R}^{(i)} = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & \overline{c}_i & \overline{s}_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix}, \tag{57}$$

где

$$s_i = \frac{H_{i+1,i}}{\sqrt{\left|H_{ii}^{(i-1)}\right|^2 + \left(H_{i+1,i}\right)^2}}, \quad c_i = \frac{H_{ii}^{(i-1)}}{\sqrt{\left|H_{ii}^{(i-1)}\right|^2 + \left(H_{i+1,i}\right)^2}}. \tag{58}$$

Следует учесть, что поддиагональный элемент $H_{i+1,i} = \left\| \breve{\mathbf{v}}^{i+1} \right\|$ матрицы $\mathbf{H}$ является нормой вектора [1, с.883], и поэтому является вещественным неотрицательным числом. Следовательно, величина s_i в (58) также является вещественным неотрицательным числом. Величина c_i в общем случае является комплексной.

СГЛАЖИВАНИЕ НЕВЯЗКИ

Невязки методов COCG и COCR и некоторых других методов могут быть подвержены сильным осцилляциям, что затрудняет контроль сходимости методов и завершение процесса по исчерпанию числа итераций. Поэтому для данных методов целесообразно использовать процедуру *сглаживания невязки* [2, с.181; 7].

Суть сглаживания невязки заключается в следующем. Пусть некоторый итерационный метод генерирует на *j*-й итерации приближенное решение x_j и соответствующий вектор невязки r_j. Вводятся два вспомогательных вектора y_j и s_j следующим образом:

$$y_0 = x_0,\ s_0 = r_0, \tag{59}$$

$$y_j = (1-\eta_j)y_{j-1} + \eta_j x_j, \quad s_j = (1-\eta_j)s_{j-1} + \eta_j r_j \ \text{ для } \ j = 1,2,3,\ldots \tag{60}$$

Коэффициент $\eta_j \in \mathbb{R}$ выбирается так, чтобы величина $\left\| f - Ay_j \right\|^2$ минимизировалась на каждом шаге:

$$\eta_j = -\frac{(s_{j-1},\ r_j - s_{j-1})}{(r_j - s_{j-1},\ r_j - s_{j-1})}. \tag{61}$$

Скалярное произведение в (61) вычисляется по формуле (4).

Для достижения строго монотонной сходимости необходимо обеспечить выполнение условия $\eta_j \in [0,\ 1]$. То есть если $\eta_j < 0$, то следует положить $\eta_j = 0$, если $\eta_j > 1$, то следует положить $\eta_j = 1$.

Решением системы на *j*-й итерации считается вектор y_j с соответствующим вектором невязки s_j (при этом выполняется неравенство $\|s_j\| \le \|r_j\|$). При этом векторы x_j и r_j вычисляются по исходным формулам итерационного метода, в который встроена процедура сглаживания невязки.

РАСПАРАЛЛЕЛИВАНИЕ ИТЕРАЦИОННЫХ МЕТОДОВ РЕШЕНИЯ СЛАУ В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ

В рассмотренных итерационных методах основными являются следующие операции, которые могут быть распараллелены: скалярное произведение и линейное комбинирование векторов, диагональное предобусловливание и умножение разреженной матрицы на вектор. Далее будут рассмотрены подходы к распараллеливанию данных операций с использованием системы параллельного программирования OpenMP [8].

Рассмотрим распараллеливание скалярных произведений. В программе применяется директива распараллеливания цикла и редуцированная операция суммирования по всем параллельным нитям. Стоит обратить внимание на то, что объявление параллельной секции занимает определенное количество процессорного времени, поэтому при относительно малой размерности векторов следует отказаться от распараллеливания.

```
double DotProduct(double *x, double *y, int n)
{
	double sum = 0.0;
	int i;

	if (n > N_MIN)
	{
		#pragma omp parallel for shared(x, y) private(i) reduction(+:sum)
		for (i=0; i<n; i++)
		{
			sum += x[i]*y[i];
```

```
        }
    }
    else
    {
        for (i=0; i<n; i++)
            sum += x[i]*y[i];
    }
    return sum;
}
```

Распараллеливание операции линейного комбинирования векторов и диагонального предобусловливания выполняется аналогично.

Рассмотрим следующие два формата хранения разреженных матриц в памяти компьютера [3]. В *разреженном строчном формате* (часто используется аббревиатура CSR – compressed sparse row) все ненулевые элементы матрицы (включая элементы из верхнего и нижнего треугольника матрицы) хранятся последовательно в строчном порядке в одном массиве. Пример подпрограммы параллельного матрично-векторного умножения для матрицы, хранящейся в разреженном строчном формате приведен ниже.

```
void MultMVCSR(int n, int *iptr, int *jptr, double *aelem, double *x, double *y)
{
    int i, j;

#pragma omp parallel shared(iptr, jptr, aelem, x, y) private(i, j)
num_threads(threadNum)
    {
        #pragma omp for
        for (i=0; i<n; i++)
        {
            y[i] = 0;
            for (j=iptr[i]; j<=iptr[i+1]-1; j++)
                y[i] += aelem[j]*x[jptr[j]];
        }
    }
}
```

В *разреженном строчно-столбцовом формате* для симметричной матрицы внедиагональные элементы матрицы хранятся только для нижнего треугольника (перечисленные в строчном порядке). При таком способе хранения элементы верхнего треугольника оказываются расположенными в том же массиве, но в столбцовом порядке, поэтому при распараллеливании операций матрично-векторного умножения с матрицей, хранящейся в разреженном строчно-столбцовом формате, могут возникать конфликты при одновременной записи в вектор-результат. Устранить такие конфликты можно следующим образом: каждая нить параллельной программы производит запись в «свой» вектор-результат, а затем производится сложение всех векторов-результатов в один. Пример подпрограммы параллельного матрично-векторного умножения для матрицы, хранящейся в разреженном строчно-столбцовом формате приведен ниже.

```
void MultMv(int *ig, int *jg, double *ggl, double *di, double *x, double *y, int n,
double *y_omp)
{
	int i, j, k;
	int rank;
	int adr;

#pragma omp parallel shared(ig, jg, ggl, di, x, y, y_omp) private(i, j, k, rank, adr)
num_threads(threadNum)
	{
		#pragma omp for
		for (i=0; i<n; i++)
			y[i] = 0.0;

		#pragma omp for
		for (i=0; i<n*(threadNum-1); i++)
			y_omp[i] = 0.0;

		#pragma omp for
		for(i=0; i<n; i++)
		{
			rank = omp_get_thread_num();

			if (rank == 0)
```

```
                {
                        y[i] = di[i]*x[i];
                        for(j=ig[i]; j<=ig[i+1]-1; j++)
                        {
                                k = jg[j];
                                y[i] += ggl[j]*x[k];
                                y[k] += ggl[j]*x[i];
                        }
                }
                else
                {
                        adr = (rank - 1)*n;
                        y_omp[adr + i] = di[i]*x[i];
                        for(j=ig[i]; j<=ig[i+1]-1; j++)
                        {
                                k = jg[j];
                                y_omp[adr + i] += ggl[j]*x[k];
                                y_omp[adr + k] += ggl[j]*x[i];
                        }
                }
        }

        #pragma omp for
        for (i=0; i<n; i++)
        {
                for (j=0; j<threadNum-1; j++)
                        y[i] += y_omp[j*n + i];
        }
}
```

Одной из проблем распараллеливания операции умножения разреженной матрицы на вектор является неравномерная заполненность матриц. При распределении строк разреженной матрицами между нитями параллельной программы процессоры (ядра) могут быть загружены неравномерно, что может вызвать снижение общей производительности. Для устранения возможных простоев строки матрицы распределяются по процессорам (ядрам) таким образом, чтобы каждому из них досталось приблизительно равное количество элементов матрицы.

Практическая часть

Реализовать итерационные методы согласно варианту задания с учетом следующих требований.

- Во всех вариантах заданий реализовать указанные методы с диагональным предобусловливанием и с предобусловливанием неполной факторизацией.
- Неполную факторизацию выполнять один раз перед началом итерационного процесса.
- Матрицу СЛАУ хранить в разреженном блочно-строчном формате [1, с.502].
- Для вычисления элементов матриц неполной факторизации комплексной матрицы использовать формулы, аналогичные формулам неполной факторизации для вещественных матриц [1, с.871-880] с учетом того, что все величины в этих формулах теперь являются комплексными.
- Квадратный корень из комплексного числа $z = a + ib$ искать как решение системы уравнений

$$\begin{cases} x^2 - y^2 = a; \\ 2xy = b. \end{cases} \tag{62}$$

  При этом выбирать значение $x + iy$ с фазой из интервала $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$ (с неотрицательной вещественной частью).
- Запрещается написание программ под .NET.
- В процессе счета записывать в файл информацию о номере итерации и относительную невязку.
- Выход из итерационного процесса осуществлять по условию малости относительной невязки

$$\frac{\left\|r^j\right\|}{\left\|f\right\|} < \varepsilon, \tag{63}$$

либо (аварийно) по превышению максимально допустимого числа итераций. При вычислении нормы векторов в (63) использовать скалярное произведение, определяемое формулой (4).

- Перед завершением работы решателя (один раз на последней итерации) для проверки выдать норму относительной невязки по формуле $(f - Ax^j) / r^0$.
- В методе GMRES для вещественных и для комплексных СЛАУ исследовать скорость сходимости от выбора глубины метода. Определить оптимальную глубину метода при решении каждой отдельно взятой СЛАУ.
- Встроить схему сглаживания невязки в программы итерационных методов.
- Протестировать разработанные программы на матрицах небольшой размерности.
- Исследовать сходимость разработанных методов решения СЛАУ на матрицах большой размерности, выданных преподавателем. Указать число итераций и время решения СЛАУ. Построить график зависимости нормы относительной невязки от числа итераций для каждого метода (со сглаживанием невязки и без – на одном графике). При этом по оси Y сделать логарифмический масштаб.
- Выполнить распараллеливание операций скалярного произведения, линейного комбинирования векторов и диагонального предобусловливания.
- Выполнить распараллеливание операции умножения разреженной матрицы на вектор в разреженном строчно-столбцовом формате, использовав прием копирования вектора-результата.
- Произвести балансировку нагрузки между нитями параллельной программы умножения разреженной матрицы на вектор, распределив равное количество элементов матрицы по нитям параллельной программы.
- Сравнить производительность последовательной и параллельной версий программ решения СЛАУ. Исследовать зависимость ускорения от числа процессоров.

- Выполнить копирование входного вектора при распараллеливании умножения разреженной матрицы на вектор. Исследовать зависимость ускорения от числа процессоров.
- При засечке времени запускать программы с высоким приоритетом:

  start /high имя_программы
- Заменить подпрограммы скалярного произведения, линейного комбинирования векторов и умножения разреженной матрицы на вектор на аналогичные подпрограммы из библиотеки математических функций Intel(R) MKL [4]. При этом предварительно сконвертировать матрицу СЛАУ в формат CSR [2, 4]. Исследовать изменение производительности программ (как последовательной, так и параллельной версии).

Варианты заданий

1. COCG для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
2. COCG для СЛАУ с комплексно-симметричными матрицами. BCG для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
3. COCG для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
4. COCG для СЛАУ с комплексно-симметричными матрицами. BCGStab для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
5. COCG для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с комплексными матрицами. Факторизация LL^T для комплексной матрицы.

6. COCR для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
7. COCR для СЛАУ с комплексно-симметричными матрицами. BCG для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
8. COCR для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
9. COCR для СЛАУ с комплексно-симметричными матрицами. BCGStab для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
10. COCR для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с комплексными матрицами. Факторизация LL^T для комплексной матрицы.
11. ЛОС для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
12. ЛОС для СЛАУ с комплексно-симметричными матрицами. BCG для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
13. ЛОС для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
14. ЛОС для СЛАУ с комплексно-симметричными матрицами. BCGStab для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.

15. ЛОС для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с комплексными матрицами. Факторизация LDL^T для комплексной матрицы.
16. GMRES для СЛАУ с комплексными матрицами. COCG для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
17. GMRES для СЛАУ с комплексными матрицами. COCR для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
18. GMRES для СЛАУ с комплексными матрицами. ЛОС для СЛАУ с комплексно-симметричными матрицами. GMRES для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
19. GMRES для СЛАУ с комплексными матрицами. COCG для СЛАУ с комплексно-симметричными матрицами. BCG для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
20. GMRES для СЛАУ с комплексными матрицами. COCR для СЛАУ с комплексно-симметричными матрицами. BCG для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
21. GMRES для СЛАУ с комплексными матрицами. ЛОС для СЛАУ с комплексно-симметричными матрицами. BCG для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
22. GMRES для СЛАУ с комплексными матрицами. COCG для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
23. GMRES для СЛАУ с комплексными матрицами. COCR для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.

24. GMRES для СЛАУ с комплексными матрицами. ЛОС для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
25. GMRES для СЛАУ с комплексными матрицами. COCG для СЛАУ с комплексно-симметричными матрицами. BCGStab для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
26. GMRES для СЛАУ с комплексными матрицами. COCR для СЛАУ с комплексно-симметричными матрицами. BCGStab для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.
27. GMRES для СЛАУ с комплексными матрицами. ЛОС для СЛАУ с комплексно-симметричными матрицами. BCGStab для СЛАУ с вещественными матрицами. Факторизация LDL^T для комплексной матрицы.
28. ЛОС для СЛАУ с комплексными матрицами. ЛОС для СЛАУ с комплексно-симметричными матрицами. ЛОС для СЛАУ с вещественными матрицами. Факторизация LL^T для комплексной матрицы.

Список литературы

Основной список

1. Соловейчик Ю.Г., Рояк М.Э., Персова М.Г. Метод конечных элементов для решения скалярных и векторных задач: Учеб. пособие. - Новосибирск: Изд-во НГТУ, 2007. – 896 с. («Учебники НГТУ»).
2. Saad Y. Iterative Methods for Sparse Linear Systems. – 2nd ed.– SIAM, Philadelphia, 2003. – 528 p. (Первое издание доступно в электронном доступе: http://www-users.cs.umn.edu/~saad/PS/all_pdf.zip)

Дополнительный список

3. Тыртышников Е.Е. Матричный анализ и линейная алгебра. – М.: ФИЗМАТЛИТ, 2007. – 480 с.
4. Intel(R) MKL Reference Manual [Электронный ресурс] – 3254 p. – Режим доступа: http://software.intel.com/sites/products/documentation/hpc/composerxe/en-us/mklxe/mkl_manual_win_mac/index.htm
5. Sogabe T., Zhang S.-L. A COCR method for solving complex symmetric linear systems // Journal of Computational and Applied Mathematics, 199(2007), p. 297-303.
6. Van der Vorst H.A., Melissen J.B.M. A Petrov-Galerkin type method for solving Ax=b, where A is symmetric complex // IEEE Transaction on Magnetics, Vol.26, No. 2(1990), p. 706-708.
7. Zhou L., Walker H. Residual smoothing techniques for iterative methods // SIAM J. Sci. Computing, vol. 15, no. 2, 1994, pp. 297-312.
8. OpenMP Application Program Interface [Электронный ресурс] – 326 p.– Режим доступа: http://www.openmp.org/mp-documents/spec30.pdf