Модуль SCF: модифицируемая факторизация матрицы на основе дополнения Шура

А. О. Махорин*

Март 2014 г.

Аннотация

В данной пояснительной записке рассмотрены математическое обоснование и программная реализация модифицируемой факторизации невырожденной квадратной матрицы на основе дополнения Шура. Соответствующий программный модуль SCF (Schur-Complement-Based Factorization) входит в состав пакета GLPK и используется для факторизации базисной матрицы в рамках симплекс-метода. (Примером реализации, где используется аналогичный подход, может служить пакет LUSOL [1, 2, 3].)

1 Математическое обоснование

1.1 Общая идея

Рассмотрим в качестве примера следующую систему уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \tag{1}$$

Допустим, что нам требуется заменить второй столбец матрицы коэффициентов (a_{12}, a_{22}, a_{32}) новым столбцом $(\overline{a}_{12}, \overline{a}_{22}, \overline{a}_{32})$. Тогда, используя обычную «физическую» замену столбца, мы получим новую систему:

$$\begin{cases} a_{11}x_1 + \overline{a}_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + \overline{a}_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + \overline{a}_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \tag{2}$$

Общая идея, лежащая в основе рассматриваемой реализации, состоит в том, чтобы вместо «физического» удаления заменяемого столбца ис-

*Кафедра прикладной информатики, Московский авиационный институт, Москва, Россия. E-mail: <mao@gnu.org>.

пользовать его «логическое» удаление, фиксируя соответствующую переменную в нуле. Так, применительно к рассматриваемому примеру новая система будет следующей:

$$\begin{cases} a_{11}x_1 + a_{12}\widetilde{x}_2 + a_{13}x_3 + \overline{a}_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}\widetilde{x}_2 + a_{23}x_3 + \overline{a}_{22}x_2 = b_2 \\ a_{31}x_1 + a_{32}\widetilde{x}_2 + a_{33}x_3 + \overline{a}_{32}x_2 = b_3 \\ \widetilde{x}_2 = 0 \end{cases} \tag{3}$$

Очевидно, что системы (2) и (3) эквивалентны (если не принимать во внимание побочную старую переменную $\widetilde{x}_2$).

Аналогичный способ можно использовать и для замены строк. Пусть, например, в системе (1) требуется заменить вторую строку (a_{21}, a_{22}, a_{23}) новой строкой $(\overline{a}_{21}, \overline{a}_{22}, \overline{a}_{23})$. В этом случае мы можем «логически» удалить старую строку, включая в нее побочную переменную, что делает такую строку «свободной» (правая часть «свободной» строки является несущественной, поэтому ее можно заменить нулем):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \widetilde{y} = 0 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \\ \overline{a}_{21}x_1 + \overline{a}_{22}x_2 + \overline{a}_{23}x_3 = b_2 \end{cases} \tag{4}$$

1.2 Расширенная матрица и ее факторизованная форма

В общем случае элементарная модификация заданной матрицы сводится к добавлению подходящей строки и столбца. Это означает, что если мы начинаем с некоторой начальной матрицы A_0, то после серии модификаций мы будем иметь *расширенную матрицу*

$$\begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix}, \tag{5}$$

где A_1, A_2, A_3 — подматрицы, соответствующие добавленным строкам и столбцам. Текущая матрица A, получаемая в результате выполненных модификаций, частично состоит из строк и столбцов начальной матрицы A_0 и частично из новых строк и столбцов, которые были добавлены в расширенную матрицу. Таким образом, A также является частью расширенной матрицы, что можно записать в виде следующего равенства:

$$\begin{pmatrix} A & \widetilde{A}_1 \\ \widetilde{A}_2 & \widetilde{A}_3 \end{pmatrix} = P \begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix} Q, \tag{6}$$

где P и Q — перестановочные матрицы.

Конечной целью факторизации является решение систем уравнений с текущей матрицей A. Поскольку строки и столбцы расширенной матрицы, не входящие в A, «логически» удалены, то решение системы $Ax = b$ можно получить как решение расширенной системы

$$\begin{pmatrix} A & \widetilde{A}_1 \\ \widetilde{A}_2 & \widetilde{A}_3 \end{pmatrix} \begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{7}$$

а решение транспонированной системы $A^T x = b$ — как решение расширенной транспонированной системы

$$\begin{pmatrix} A^T & \widetilde{A}_2^T \\ \widetilde{A}_1^T & \widetilde{A}_3^T \end{pmatrix} \begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{8}$$

где $\widetilde{x}$ — вектор побочных переменных.

В соответствии с (6) решение расширенных систем (7) и (8) с точностью до перестановок компонент векторов переменных и правых частей, определяемых перестановочными матрицами P и Q, эквивалентно решению систем с расширенной матрицей (5). Для этой цели используется следующая факторизованная форма расширенной матрицы:

$$\begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix} = \begin{pmatrix} R_0 & \\ R & I \end{pmatrix} \begin{pmatrix} S_0 & S \\ & C \end{pmatrix}. \tag{9}$$

Здесь матрицы R_0 и S_0 определяют факторизацию начальной матрицы $A_0 = R_0 S_0$. (В принципе, можно использовать любую подходящую разреженную факторизацию, которая позволяет эффективно решать системы с матрицами R_0 и S_0.) Заметим, что начальная матрица A_0, а значит, и матричные факторы R_0 и S_0 не изменяются в процессе модификации текущей матрицы A. Матрицы $R = A_2 S_0^{-1}$, $S = R_0^{-1} A_1$, а также матрица $C = A_3 - RS = A_3 - A_2 A_0^{-1} A_1$, называемая дополнением Шура (матрицы A_0), изменяются при каждой элементарной модификации текущей матрицы A.

1.3 Решение систем уравнений с текущей матрицей

Пусть A — текущая матрица, для которой задана ее факторизация в виде (6) и (9). Как уже было отмечено в предыдущем подразделе, решение системы $Ax = b$ сводится к решению расширенной системы (7), которая с учетом (6) и (9) получается следующей:

$$P \begin{pmatrix} R_0 & \\ R & I \end{pmatrix} \begin{pmatrix} S_0 & S \\ & C \end{pmatrix} Q \begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix},$$

откуда

$$\begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = Q^T \begin{pmatrix} S_0 & S \\ & C \end{pmatrix}^{-1} \begin{pmatrix} R_0 & \\ R & I \end{pmatrix}^{-1} P^T \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Таким образом, расширенный вектор решения можно вычислить по следующим формулам:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = P^T \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{10}$$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} R_0 & \\ R & I \end{pmatrix}^{-1} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \Rightarrow \begin{cases} v_1 = R_0^{-1} u_1 \\ v_2 = u_2 - R v_1 \end{cases} \tag{11}$$

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} S_0 & S \\ & C \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \Rightarrow \begin{cases} w_2 = C^{-1} v_2 \\ w_1 = S_0^{-1}(v_1 - S w_2) \end{cases} \tag{12}$$

$$\begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = Q^T \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}. \tag{13}$$

Аналогично, решение транспонированной системы $A^T x = b$ сводится к решению расширенной транспонированной системы (8), которая с учетом (6) и (9) получается следующей:

$$Q^T \begin{pmatrix} S_0^T & \\ S^T & C^T \end{pmatrix} \begin{pmatrix} R_0^T & R^T \\ & I \end{pmatrix} P^T \begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix},$$

откуда

$$\begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = P \begin{pmatrix} R_0^T & R^T \\ & I \end{pmatrix}^{-1} \begin{pmatrix} S_0^T & \\ S^T & C^T \end{pmatrix}^{-1} Q \begin{pmatrix} b \\ 0 \end{pmatrix},$$

что дает необходимые формулы для вычисления расширенного вектора решения:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = Q \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{14}$$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} S_0^T & \\ S^T & C^T \end{pmatrix}^{-1} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \Rightarrow \begin{cases} v_1 = S_0^{-T} u_1 \\ v_2 = C^{-T}(u_2 - S^T v_1) \end{cases} \tag{15}$$

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} R_0^T & R^T \\ & I \end{pmatrix}^{-1} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \Rightarrow \begin{cases} w_2 = v_2 \\ w_1 = R_0^{-T}(v_1 - R^T w_2) \end{cases} \tag{16}$$

$$\begin{pmatrix} x \\ \widetilde{x} \end{pmatrix} = P \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}. \tag{17}$$

1.4 Элементарные модификации текущей матрицы

Здесь рассмотрены только два вида элементарных модификаций: замена строки и замена столбца. Модификации других видов (например, добавление строки и столбца, удаление строки и столбца, и т. п.) могут быть получены аналогично.

Замена строки. Пусть в текущей матрице A требуется заменить ее i-ю строку новой строкой α^T. В этом случае мы расширяем матрицу в левой части (6) следующим образом:

$$\left(\begin{array}{cc|c} A & \widetilde{A}_1 & e_i \\ \widetilde{A}_2 & \widetilde{A}_3 & 0 \\ \hline \alpha^T & 0 & 0 \end{array}\right), \tag{18}$$

где e_i — единичный вектор, содержащий единицу в i-й позиции. Для прямой системы этот вектор соответствует побочной переменной, которая делает старую строку свободной и тем самым «логически» удаляет эту строку.

Модификация (18) приводит к соответствующей модификации расширенной матрицы (5), новые компоненты которой можно определить исходя из равенства (6):

$$\left(\begin{array}{cc} P^T & \\ & 1 \end{array}\right) \left(\begin{array}{cc|c} A & \widetilde{A}_1 & e_i \\ \widetilde{A}_2 & \widetilde{A}_3 & 0 \\ \hline \alpha^T & 0 & 0 \end{array}\right) \left(\begin{array}{cc} Q^T & \\ & 1 \end{array}\right) =$$

$$= \left(\begin{array}{c|c} P^T \begin{pmatrix} A & \widetilde{A}_1 \\ \widetilde{A}_2 & \widetilde{A}_3 \end{pmatrix} Q^T & P^T \begin{pmatrix} e_i \\ 0 \end{pmatrix} \\ \hline \begin{pmatrix} \alpha^T & 0 \end{pmatrix} Q^T & 0 \end{array}\right) = \left(\begin{array}{cc|c} A_0 & A_1 & b \\ A_2 & A_3 & f \\ \hline d^T & g^T & 0 \end{array}\right),$$

где

$$\begin{pmatrix} b \\ f \end{pmatrix} = P^T \begin{pmatrix} e_i \\ 0 \end{pmatrix}, \quad \begin{pmatrix} d \\ g \end{pmatrix} = Q \begin{pmatrix} \alpha \\ 0 \end{pmatrix}. \tag{19}$$

Наконец, чтобы матрица (18) соответствовала новой текущей матрице следует поменять местами ее i-ю и последнюю (новую) строчки. В соответствии с равенством (6) для этого необходимо поменять местами i-ю и последнюю строчки новой перестановочной матрицы P, при этом новая расширенная матрица остается неизменной.

Замена столбца. Пусть в текущей матрице A требуется заменить ее j-й столбец новым столбцом β. В этом случае мы расширяем матрицу в левой части (6) следующим образом:

$$\left(\begin{array}{cc|c} A & \widetilde{A}_1 & \beta \\ \widetilde{A}_2 & \widetilde{A}_3 & 0 \\ \hline e_j^T & 0 & 0 \end{array}\right), \tag{20}$$

где e_j — единичный вектор, содержащий единицу в j-й позиции. Для прямой системы этот вектор фиксирует старый столбец в нуле и тем самым «логически» удаляет этот столбец.

Модификация (20) приводит к соответствующей модификации расширенной матрицы (5), новые компоненты которой можно определить исходя из равенства (6):

$$\left(\begin{array}{cc} P^T & \\ & 1 \end{array}\right) \left(\begin{array}{cc|c} A & \widetilde{A}_1 & \beta \\ \widetilde{A}_2 & \widetilde{A}_3 & 0 \\ \hline e_j^T & 0 & 0 \end{array}\right) \left(\begin{array}{cc} Q^T & \\ & 1 \end{array}\right) =$$

$$= \left(\begin{array}{cc|c} P^T \left(\begin{array}{cc} A & \widetilde{A}_1 \\ \widetilde{A}_2 & \widetilde{A}_3 \end{array}\right) Q^T & & P^T \left(\begin{array}{c} \beta \\ 0 \end{array}\right) \\ \hline \left(\begin{array}{cc} e_j^T & 0 \end{array}\right) Q^T & & 0 \end{array}\right) = \left(\begin{array}{cc|c} A_0 & A_1 & b \\ A_2 & A_3 & f \\ \hline d^T & g^T & 0 \end{array}\right),$$

где

$$\left(\begin{array}{c} b \\ f \end{array}\right) = P^T \left(\begin{array}{c} \beta \\ 0 \end{array}\right), \quad \left(\begin{array}{c} d \\ g \end{array}\right) = Q \left(\begin{array}{c} e_j \\ 0 \end{array}\right). \tag{21}$$

Наконец, чтобы матрица (20) соответствовала новой текущей матрице следует поменять местами ее j-й и последний (новый) столбцы. В соответствии с равенством (6) для этого необходимо поменять местами j-и последний столбцы новой перестановочной матрицы Q, при этом новая расширенная матрица остается неизменной.

1.5 Пересчет факторизации расширенной матрицы

В общем случае в результате элементарной модификации текущей матрицы A новая расширенная матрица получается из текущей расширенной матрицы (5) и имеет следующий вид:

$$\left(\begin{array}{cc} A_0 & \overline{A}_1 \\ \overline{A}_2 & \overline{A}_3 \end{array}\right) = \left(\begin{array}{c|cc} A_0 & A_1 & b \\ \hline A_2 & A_3 & f \\ d^T & g^T & h \end{array}\right), \tag{22}$$

где b, f — добавленные столбцы, d^T, g^T — добавленные строки, h — добавленный диагональный элемент.

В соответствии с (9) для новой расширенной матрицы имеем:

$$\left(\begin{array}{cc} A_0 & \overline{A}_1 \\ \overline{A}_2 & \overline{A}_3 \end{array}\right) = \left(\begin{array}{cc} R_0 & \\ \overline{R} & I \end{array}\right) \left(\begin{array}{cc} S_0 & \overline{S} \\ & \overline{C} \end{array}\right) =$$

$$= \left(\begin{array}{c|cc} R_0 & & \\ \hline R & I & \\ w^T & & 1 \end{array}\right) \left(\begin{array}{c|cc} S_0 & S & v \\ \hline & C & x \\ & y^T & z \end{array}\right) = \left(\begin{array}{c|cc} R_0 S_0 & R_0 S & R_0 v \\ \hline RS_0 & RS + C & Rv + x \\ w^T S_0 & w^T S + y^T & w^T v + z \end{array}\right) =$$

$$= \left(\begin{array}{c|cc} A_0 & A_1 & R_0 v \\ \hline A_2 & A_3 & Rv + x \\ w^T S_0 & w^T S + y^T & w^T v + z \end{array}\right) = \left(\begin{array}{c|cc} A_0 & A_1 & b \\ \hline A_2 & A_3 & f \\ d^T & g^T & h \end{array}\right).$$

Поэтому

$$\begin{aligned} v &= R_0^{-1} b, \\ w &= S_0^{-T} d, \\ x &= f - Rv, \\ y &= g - S^T w, \\ z &= h - v^T w. \end{aligned} \tag{23}$$

Таким образом, новая расширенная матрица получается из текущей расширенной матрицы в результате добавления новых компонент: $\overline{R}$ получается добавлением к R новой строки w^T, $\overline{S}$ получается добавлением к S нового столбца v, а $\overline{C}$ получается добавлением к C нового столбца x, новой строки y^T и диагонального элемента z.

Следует заметить, что для решения систем уравнений с текущей матрицей (см. подразд. 1.3) требуется решать вспомогательные системы с матрицей C, что предполагает наличие некоторой факторизованной формы этой матрицы. Таким образом, пересчет факторизации расширенной матрицы также включает в себя пересчет используемой факторизации матрицы C. Однако эти вопросы изложены в другой статье и поэтому здесь не рассматриваются.

2 Программная реализация

2.1 Основная часть

Основная часть (файлы `scf.c` и `scf.h`) включает в себя описание структурного типа данных `SCF`, который представляет факторизацию текущей матрицы A, а также набор подпрограмм, выполняющих основные операции.

Структура SCF

Структурный тип данных `SCF` служит для представления факторизации (6) и (9) текущей матрицы A. Эта структура имеет следующие поля:

`int n` — n, порядок текущей матрицы A;

`int n0` — n_0, порядок начальной матрицы A_0;

`int type` — тип используемой факторизации A_0:

1 — простая LU-факторизация (модуль LUF). Для этого типа факторизации $A_0 = F_0 V_0$, $F_0 = P_0 L P_0^T$, $V_0 = P_0 U Q_0$, где F_0 и V_0 — неявные треугольные матричные сомножители, L_0 — нижняя треугольная матрица с единичной диагональю, U_0 — верхняя треугольная матрица, P_0 и Q_0 — перестановочные матрицы. В данном случае $R_0 = F_0$ и $S_0 = V_0$;

2 — блочно-треугольная LU-факторизация (модуль BTF). Для этого типа факторизации $A_0 = P_0 \widetilde{A}_0 Q_0$, где $\widetilde{A}_0$ — верхняя блочно-треугольная

матрица, P_0 и Q_0 — перестановочные матрицы, при этом для каждого диагонального блока $\widetilde{A}_0$ используется простая LU-факторизация. В данном случае $R_0 = I$ (единичная матрица) и $S_0 = A_0$.

`union { LUF *luf; BTF *btf; } a0` — факторизация A_0. В случае `type` $= 1$ используется указатель `a0.luf` на объект типа `LUF` (простая LU-факторизация). В случае `type` $= 2$ используется указатель `a0.btf` на объект типа `BTF` (блочно-треугольная LU-факторизация);

`int nn_max` — nn_{max}, максимальное число дополнительных строк и столбцов расширенной матрицы (5). Этот параметр ограничивает число возможных элементарных модификаций;

`int nn` — nn, текущее число дополнительных строк и столбцов расширенной матрицы (5), $0 \le nn \le nn_{max}$;

`SVA *sva` — указатель на область разреженных векторов (объект типа SVA) для хранения строк матрицы R и столбцов матрицы S. Следует заметить, что модуль SCF использует ту же область SVA, которая используется модулем LUF или BTF для хранения компонент факторизации матрицы A_0, при этом строки R и столбцы S записываются в правую (статическую) часть SVA;

`int rr_ref` — ссылочный номер разреженного вектора в SVA, который соответствует первой строке матрицы R. Всего резервируется nn_{max} векторов для возможного расширения R;

`int ss_ref` — ссылочный номер разреженного вектора в SVA, который соответствует первому столбцу матрицы S. Всего резервируется nn_{max} векторов для возможного расширения S;

`IFU ifu` — объект типа `IFU`, представляющий факторизацию матрицы C, имеющей порядок nn (модуль IFU). Поскольку порядок C предполагается небольшим (он ограничен параметром nn_{max}), то модуль IFU использует плотный формат представления матриц;

`int *pp_ind, *pp_inv` — указатели на массивы длины $1 + n_0 + nn$, представляющие перестановочную матрицу $P = (p_{ij})$: если $p_{ij} = 1$, то `pp_ind`$[i] = j$ и `pp_inv`$[j] = i$, $1 \le i, j \le n_0 + nn$. Всего резервируется $1 + n_0 + nn_{\max}$ ячеек для возможного расширения P (ячейки с индексом 0 не используются);

`int *qq_ind, *qq_inv` — указатели на массивы длины $1 + n_0 + nn$, представляющие перестановочную матрицу $Q = (q_{ij})$: если $q_{ij} = 1$, то `qq_ind`$[i] = j$ и `qq_inv`$[j] = i$, $1 \le i, j \le n_0 + nn$. Всего резервируется $1 + n_0 + nn_{\max}$ ячеек для возможного расширения Q (ячейки с индексом 0 не используются).

Подпрограмма scf_r0_solve

```
void scf_r0_solve(SCF *scf, int tr, double x[/*1+n0*/]);
```

Данная подпрограмма предназначена для решения системы $R_0 x = b$ (если параметр `tr` равен нулю) или системы $R_0^T x = b$ (если параметр `tr` отличен от нуля), где R_0 — левый матричный сомножитель начальной матрицы A_0 порядка n_0.

На входе в подпрограмму элементы вектора правых частей b должны быть записаны в ячейки массива `x[1]`, ..., `x[n0]`. На выходе подпрограмма записывает элементы вектора решения x в те же самые ячейки.

В случае `scf->type` = 1 имеет место $R_0 = F_0$ (см. описание структуры `SCF`), поэтому для решения системы $F_0 x = b$ ($F_0^T x = b$) данная подпрограмма использует подпрограмму `luf_f_solve` (`luf_ft_solve`), входящую в модуль LUF.

В случае `scf->type` = 2 имеет место $R_0 = I$ (единичная матрица), поэтому подпрограмма просто оставляет содержимое массива `x` неизменным, так как $x = b$.

Подпрограмма scf_s0_solve

```
void scf_s0_solve(SCF *scf, int tr, double x[/*1+n0*/],
      double w1[/*1+n0*/], double w2[/*1+n0*/], double w3[/*1+n0*/]);
```

Данная подпрограмма предназначена для решения системы $S_0 x = b$ (если параметр `tr` равен нулю) или системы $S_0^T x = b$ (если параметр `tr` отличен от нуля), где S_0 — правый матричный сомножитель начальной матрицы A_0 порядка n_0.

На входе в подпрограмму элементы вектора правых частей b должны быть записаны в ячейки массива `x[1]`, ..., `x[n0]`. На выходе подпрограмма записывает элементы вектора решения x в те же самые ячейки.

Подпрограмма использует ячейки `[1]`, ..., `[n0]` трех рабочих массивов `w1`, `w2` и `w3`. (В случае `scf->type` = 1 массивы `w2` и `w3` не используются и могут быть заданы как `NULL`.)

В случае `scf->type` = 1 имеет место $S_0 = V_0$ (см. описание структуры `SCF`), поэтому для решения системы $V_0 x = b$ ($V_0^T x = b$) данная подпрограмма использует подпрограмму `luf_v_solve` (`luf_vt_solve`), входящую в модуль LUF.

В случае `scf->type` = 2 имеет место $S_0 = A_0$ (начальная матрица), поэтому для решения системы $A_0 x = b$ ($A_0^T x = b$) данная подпрограмма использует подпрограмму `btf_a_solve` (`btf_at_solve`), входящую в модуль BTF.

Подпрограмма scf_r_prod

```
void scf_r_prod(SCF *scf, double y[/*1+nn*/], double a, const double
     x[/*1+n0*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения

$$\overline{y} = y + \alpha R x, \tag{24}$$

где R — $nn \times n_0$-матрица в факторизации (9) текущей расширенной матрицы, x — заданный n_0-вектор, α — заданный скаляр, y — заданный nn-вектор.

На входе в подпрограмму элементы векторов x и y должны находится в ячейках массивов `x[1]`, ..., `x[n0]` и `y[1]`, ..., `y[nn]`, соответственно. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ в массив `y` на место исходного вектора y.

Поскольку матрица $R = (r_{ij})$ хранится в строчном разреженном формате, то компоненты результирующего вектора вычисляются как скалярные произведения:

$$\overline{y}_i = y_i + \alpha(i\text{-я строка } R)x = y_i + \alpha \sum_{j=1}^{n_0} r_{ij} x_j \tag{25}$$

для $i = 1, \ldots, nn$.

Подпрограмма scf_rt_prod

```
void scf_rt_prod(SCF *scf, double y[/*1+n0*/], double a, const double
     x[/*1+nn*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения

$$\overline{y} = y + \alpha R^T x, \tag{26}$$

где R — $nn \times n_0$-матрица в факторизации (9) текущей расширенной матрицы, x — заданный nn-вектор, α — заданный скаляр, y — заданный n_0-вектор.

На входе в подпрограмму элементы векторов x и y должны находится в ячейках массивов `x[1]`, ..., `x[nn]` и `y[1]`, ..., `y[n0]`, соответственно. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ в массив `y` на место исходного вектора y.

Поскольку матрица R хранится в строчном разреженном формате, то результирующий вектор вычисляется как линейная комбинация строк этой матрицы:

$$\overline{y} = y + \alpha \sum_{i=1}^{nn} R_i^T x_i, \tag{27}$$

где R_i^T — i-я строка матрицы R.

Подпрограмма scf_s_prod

```
void scf_s_prod(SCF *scf, double y[/*1+n0*/], double a, const double
     x[/*1+nn*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения

$$\overline{y} = y + \alpha S x, \tag{28}$$

где S — $n_0 \times nn$-матрица в факторизации (9) текущей расширенной матрицы, x — заданный nn-вектор, α — заданный скаляр, y — заданный n_0-вектор.

На входе в подпрограмму элементы векторов x и y должны находится в ячейках массивов `x[1]`, ..., `x[nn]` и `y[1]`, ..., `y[n0]`, соответственно. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ в массив `y` на место исходного вектора y.

Поскольку матрица S хранится в столбцовом разреженном формате, то результирующий вектор вычисляется как линейная комбинация столбцов этой матрицы:

$$\overline{y} = y + \alpha \sum_{j=1}^{nn} S_j x_j, \tag{29}$$

где S_j — j-й столбец матрицы S.

Подпрограмма scf_st_prod

```
void scf_st_prod(SCF *scf, double y[/*1+nn*/], double a, const double
     x[/*1+n0*/]);
```

Данная подпрограмма предназначена для вычисления матричного произведения

$$\overline{y} = y + \alpha S^T x, \tag{30}$$

где S — $n_0 \times nn$-матрица в факторизации (9) текущей расширенной матрицы, x — заданный $n0$-вектор, α — заданный скаляр, y — заданный nn-вектор.

На входе в подпрограмму элементы векторов x и y должны находится в ячейках массивов `x[1]`, ..., `x[n0]` и `y[1]`, ..., `y[nn]`, соответственно. На выходе подпрограмма записывает элементы вычисленного вектора $\overline{y}$ в массив `y` на место исходного вектора y.

Поскольку матрица $S = (s_{ij})$ хранится в столбцовом разреженном формате, то компоненты результирующего вектора вычисляются как скалярные произведения:

$$\overline{y}_j = y_j + \alpha(j\text{-й столбец } S)x = y_j + \alpha \sum_{i=1}^{n_0} s_{ij} x_i \tag{31}$$

для $j = 1, \ldots, nn$.

Подпрограмма scf_a_solve

```
void scf_a_solve(SCF *scf, double x[/*1+n*/],
      double w[/*1+n0+nn*/], double work1[/*1+max(n0,nn)*/],
      double work2[/*1+n*/], double work3[/*1+n*/]);
```

Данная подпрограмма предназначена для решения системы $Ax = b$, где A — текущая матрица порядка n.

На входе в подпрограмму элементы вектора правых частей b должны быть записаны в ячейки массива `x[1]`, ..., `x[n]`. На выходе подпрограмма записывает элементы вектора решения x в те же самые ячейки.

Размеры рабочих массивов `w`, `work1`, `work2` и `work3` должны быть не меньше, чем указано в спецификации подпрограммы (см. выше). Ячейки этих массивов с индексом 0 не используются.

Вычисление решения выполняется в соответствии с формулами (10)–(13) из подразд. 1.3. Для решения систем с матрицами R_0 и S_0 данная подпрограмма использует подпрограммы `scf_r0_solve` и `scf_s0_solve`, а для вычисления произведений $u_2 - Rv_1$ и $v_1 - Sw_2$ — подпрограммы `scf_r_prod` и `scf_s_prod`. Для решения системы с матрицей C используется подпрограмма `ifu_a_solve`, входящая в состав модуля IFU.

Подпрограмма scf_at_solve

```
void scf_at_solve(SCF *scf, double x[/*1+n*/],
      double w[/*1+n0+nn*/], double work1[/*1+max(n0,nn)*/],
      double work2[/*1+n*/], double work3[/*1+n*/]);
```

Данная подпрограмма предназначена для решения системы $A^T x = b$, где A — текущая матрица порядка n.

На входе в подпрограмму элементы вектора правых частей b должны быть записаны в ячейки массива `x[1]`, ..., `x[n]`. На выходе подпрограмма записывает элементы вектора решения x в те же самые ячейки.

Размеры рабочих массивов `w`, `work1`, `work2` и `work3` должны быть не меньше, чем указано в спецификации подпрограммы (см. выше). Ячейки этих массивов с индексом 0 не используются.

Вычисление решения выполняется в соответствии с формулами (14)–(17) из подразд. 1.3. Для решения систем с матрицами R_0^T и S_0^T данная подпрограмма использует подпрограммы `scf_r0_solve` и `scf_s0_solve`, а для вычисления произведений $u_2 - S^T v_1$ и $v_1 - R^T w_2$ — подпрограммы `scf_rt_prod` и `scf_st_prod`. Для решения системы с матрицей C^T используется подпрограмма `ifu_at_solve`, входящая в состав модуля IFU.

Подпрограмма scf_add_r_row

```
void scf_add_r_row(SCF *scf, const double w[/*1+n0*/]);
```

Данная подпрограмма добавляет новую строку с номером $nn + 1$ к $nn \times n_0$-матрице R, которая является частью факторизации (9) текущей расширенной матрицы. Элементы новой строки должны находиться в ячейках массива `w[1]`, ..., `w[n0]`. Предполагается, что $nn < nn_{max}$.

Подпрограмма scf_add_s_col

```
void scf_add_s_col(SCF *scf, const double v[/*1+n0*/]);
```

Данная подпрограмма добавляет новый столбец с номером $nn + 1$ к $n_0 \times nn$-матрице S, которая является частью факторизации (9) текущей расширенной матрицы. Элементы нового столбца должны находиться в ячейках массива `v[1]`, ..., `v[n0]`. Предполагается, что $nn < nn_{max}$.

Подпрограмма scf_update_aug

```
int scf_update_aug(SCF *scf, double b[/*1+n0*/], double d[/*1+n0*/],
      double f[/*1+nn*/], double g[/*1+nn*/], double h, int upd,
      double w1[/*1+n0*/], double w2[/*1+n0*/], double w3[/*1+n0*/]);
```

Данная подпрограмма выполняет пересчет факторизации (9) расширенной матрицы, чтобы новая факторизация соответствовала новой расширенной матрице (22).

На входе в подпрограмму элементы векторов b, d, f и g, определяющих новую расширенную матрицу, должны находиться в ячейках массивов `b[1]`, ..., `b[n0]`, `d[1]`, ..., `d[n0]`, `f[1]`, ..., `f[nn]`, `g[1]`, ..., `g[nn]`, соответственно. Подпрограмма использует эти массивы как рабочие, поэтому на выходе их содержимое не сохраняется. Параметр `h` задает диагональный элемент h для новой расширенной матрицы.

Параметр `upd` задает метод пересчета факторизации матрицы C (подробнее см. описание модуля IFU):

1 — метод Бартелса—Голуба;

2 — метод вращения Гивенса.

Длина рабочих массивов `w1`, `w2` и `w3` должна быть не меньше $1 + n_0$. Ячейки этих массивов с индексом 0 не используются.

Подпрограмма возвращает один из следующих признаков:

0 — пересчет факторизации расширенной матрицы успешно выполнен;

1 — достигнуто максимальное число элементарных модификаций (на входе в подпрограмму $nn = nn_{max}$);

2 — невозможно выполнить пересчет факторизации матрицы C (подробнее см. описание модуля IFU).

Для пересчета факторизации подпрограмма вначале вычисляет векторы v, w, x, y, а также скаляр z по формулам (23), записывая указанные векторы на место векторов b, d, f и g, соответственно. Для решения систем с матрицами R_0 и S_0^T используются подпрограммы `scf_r0_solve` и `scf_s0_solve`, а для вычисления произведений $f - Rv$ и $g - S^T w$ — подпрограммы `scf_r_prod` и `scf_st_prod`. Далее вычисленный вектор w добавляется в качестве новой $(nn+1)$-й строки к матрице R, а вычисленный вектор v — в качестве нового $(nn+1)$-го столбца к матрице S, для чего используются подпрограммы `scf_add_r_row` и `scf_add_s_col`. После этого выполняется пересчет факторизации матрицы C, чтобы ее новая факторизация соответствовала новой матрице

$$\overline{C} = \begin{pmatrix} C & x \\ y^T & z \end{pmatrix}, \tag{32}$$

для чего используется подпрограмма `ifu_bg_update` (если `upd` = 1) или подпрограмма `ifu_gr_update` (если `upd` = 2). В заключение перестановочные матрицы P и Q расширяются для получения новых перестановочных матриц

$$\overline{P} = \begin{pmatrix} P & \\ & 1 \end{pmatrix}, \quad \overline{Q} = \begin{pmatrix} Q & \\ & 1 \end{pmatrix}, \tag{33}$$

и счетчик элементарных модификаций nn увеличивается на единицу.

2.2 Интерфейс

Интерфейс к факторизации (файлы `scfint.c` и `scfint.h`) включает в себя описание структурного типа данных `SCFINT`, представляющего интерфейс, а также набор подпрограмм, реализующих интерфейс.

Структура SCFINT

Структурный тип данных `SCFINT` представляет интерфейс к факторизации (6) и (9) текущей матрицы A. Эта структура имеет следующие поля:

`int valid` — текущая факторизация является действительной только в том случае, если установлен данный флажок;

`SCF scf` — объект типа `SCF`, представляющий факторизацию (6) и (9) текущей матрицы A (см. подразд. 2.1);

`union { LUFINT *lufi; BTFINT *btfi; } u` — интерфейс к факторизации начальной матрицы A_0. В случае `scf.type` = 1 используется указатель `u.lufi` (простая LU-факторизация). В случае `scf.type` = 2 используется указатель `u.btfi` (блочно-треугольная LU-факторизация).

`double *w1, *w2, *w3` — указатели на рабочие массивы, имеющие размерность 1 + `n0_max`;

`double *w3, *w4` — указатели на рабочие массивы, имеющие размерность 1 + `n0_max` + `nn_max`;

`nn_max` — nn_{max}, управляющий параметр, определяющий максимальное число элементарных модификаций. Если необходимо, вызывающая программа может изменить значение этого управляющего параметра. (Новое значение nn_{max} вступает в силу не сразу, а только после вызова подпрограммы `scfint_factorize`.)

Примечание. Управляющий параметр `n0_max`, определяющий максимальный порядок начальной матрицы A_0, при котором не требуется переразмещение массивов, соответствует управляющему параметру `n_max`, который является частью структур `LUFINT` и `BTFINT`. В случае необходимости этот параметр увеличивается автоматически.

Подпрограмма scfint_create

```
SCFINT *scfint_create(int type);
```

Данная подпрограмма создает программный объект типа SCFINT (интерфейс к факторизации).

Параметр `type` определяет тип используемой факторизации начальной матрицы A_0 (подробнее см. описание структуры `SCF` в подразд. 2.1):

1 — простая LU-факторизация;

2 — блочно-треугольная LU-факторизация.

На выходе данная подпрограмма возвращает указатель на созданный интерфейс. Вызывающая программа может использовать этот указатель для выполнения необходимых операций с факторизацией.

Подпрограмма scfint_factorize

```
int scfint_factorize(SCFINT *fi, int n, int (*col)(void *info, int j,
      int ind[], double val[]), void *info);
```

Данная подпрограмма вычисляет факторизацию заданной матрицы A, которая становится начальной матрицей A_0.

Параметр `n` задает порядок матрицы A, $n > 0$.

Для получения элементов матрицы A данная подпрограмма вызывает формальную подпрограмму `col` и передает ей некоторый номер столбца $1 \le j \le n$, а также транзитный указатель `info`, переданный вызывающей программой. В этом случае подпрограмма `col` должна записать строчные индексы и значения ненулевых элементов j-го столбца A в ячейки `ind[1]`, ..., `ind[len]` и `val[1]`, ..., `val[len]`, соответственно, где `len` — длина j-го столбца (число ненулевых элементов в этом столбце), $0 \le$ `len` $\le n$, возвращаемая в качестве результата.

Если факторизация успешно вычислена, данная подпрограмма возвращает нуль. В противном случае возвращается признак, который был выдан подпрограммой `lufint_factorize` (если используется простая LU-факторизация) или подпрограммой `btfint_factorize` (если используется блочная LU-факторизация).

Подпрограмма `scfint_factorize` вычисляет факторизацию заданной матрицы A следующим образом.

В зависимости от типа используемой факторизации данная подпрограмма вызывает подпрограмму `lufint_factorize` (простая LU-факторизация) или `btfint_factorize` (блочная LU-факторизация), которые вычисляют факторизацию A.

Далее подпрограмма определяет новые значения управляющих параметров `n0_max` и `nn_max` и, если необходимо, переразмещает массивы в структурах `SCF` и `SCFINT`, размер которых зависит от этих управляющих параметров.

В завершение данная подпрограмма устанавливает $n_0 = n$, $nn = 0$, $R = \emptyset$, $S = \emptyset$, $C = \emptyset$ ($\emptyset$ — пустая матрица, не имеющая строк и/или столбцов), $P = Q = I$ (I — единичная матрица). С точки зрения факторизации (6) и (9) это означает, что начальная матрица A_0 совпадает с текущей (заданной) матрицей A.

Подпрограмма scfint_update

```
int scfint_update(SCFINT *fi, int upd, int j, int len, const int ind[],
      const double val[]);
```

Данная подпрограмма выполняет пересчет факторизации после замены столбца в текущей матрице A. (На входе в подпрограмму факторизация должна быть действительной.)

Параметр `upd` задает метод пересчета факторизации матрицы C (см. описание подпрограммы `scf_update_aug` в подразд. 2.1).

Параметр j задает номер измененного столбца, $1 \le j \le n$, где n — порядок текущей матрицы A.

Строчные индексы и значения элементов нового j-го столбца A должны находиться в ячейках массивов `ind[1]`, ..., `ind[len]` и `val[1]`, ..., `val[len]`, соответственно, где `len` — длина нового j-го столбца (число ненулевых элементов в этом столбце), $0 \le$ `len` $\le n$.

Признак, возвращаемый данной подпрограммой имеет тот же смысл, что и для подпрограммы `scf_update_aug` (см. подразд. 2.1).

Для пересчета факторизации используется способ, рассмотренный в подразд. 1.4. Данная подпрограмма вычисляет векторы b, f, d и g по формуле (21), вызывает подпрограмму `scf_update_aug` (см. подразд. 2.1), после чего меняет местами j-й и последний (только что добавленный) столбцы матрицы Q.

Подпрограмма scfint_ftran

```
void scfint_ftran(SCFINT *fi, double x[]);
```

Данная подпрограмма предназначена для решения системы $Ax = b$, где A — текущая матрица порядка n. (На входе в подпрограмму факторизация должна быть действительной.)

На входе в подпрограмму элементы вектора правых частей b должны быть записаны в ячейки массива `x[1]`, ..., `x[n]`. На выходе подпрограмма записывает элементы вектора решения x в те же самые ячейки.

Для решения системы данная подпрограмма использует подпрограмму `scf_a_solve` (см. подразд. 2.1).

Подпрограмма scfint_btran

```
void scfint_btran(SCFINT *fi, double x[]);
```

Данная подпрограмма предназначена для решения системы $A^T x = b$, где A — текущая матрица порядка n. (На входе в подпрограмму факторизация должна быть действительной.)

На входе в подпрограмму элементы вектора правых частей b должны быть записаны в ячейки массива `x[1]`, ..., `x[n]`. На выходе подпрограмма записывает элементы вектора решения x в те же самые ячейки.

Для решения системы данная подпрограмма использует подпрограмму `scf_at_solve` (см. подразд. 2.1).

Подпрограмма scfint_delete

```
void scfint_delete(SCFINT *fi);
```

Данная подпрограмма удаляет указанный программный объект типа `SCFINT` (интерфейс к факторизации), освобождая всю занимаемую этим объектом память.

Литература

[1] M. A. Saunders, “LUSOL: A basis package for constrained optimization,” SCCM, Stanford University, 2006.

[2] M. A. Saunders, “Notes 5: Basis Updates,” CME 318, Stanford University, Spring 2006.

[3] M. A. Saunders, “Notes 6: LUSOL—a Basis Factorization Package,” ibid.