Новосибирский центр информационных технологий "УНИПРО"
http://unipro.ru

Пакет программ USPARS для решения разреженных СЛАУ методом Гаусса. версия 2.2.1

Руководство пользователя

14.07.2023

Оглавление

Предисловие

Технические возможности современных компьютеров позволяют решать системы линейных алгебраических уравнений (СЛАУ), включающие миллионы уравнений, матрицы коэффициентов обладают свойством разреженности. Такие СЛАУ возникают, в частности, в результате дискретизации двух- или трехмерных краевых задач для уравнений в частных производных. Разреженность матриц коэффициентов является специфическим свойством таких СЛАУ, используемым при конструировании методов их решения и разработке соответствующего программного обеспечения. В программах, созданных на базе итерационных алгоритмов, разреженность обеспечивает сравнительную дешевизну операции умножения матрицы на вектор.

Метод Гаусса, основанный на треугольной факторизации матрицы коэффициентов, является альтернативным способом решения СЛАУ с разреженными матрицами коэффициентов. Неполный список библиотек, разработанных с использованием метода Гаусса, включает

- PARDISO
- MUMPS
- Super LU
- WSMP

В этот список следует добавить Intel®Math Kernel Library, одной из компонент которой является версия PARDISO, несколько отличающаяся от оригинальной, упомянутой выше. В контексте обзора методов и программных продуктов для решения СЛАУ прямыми методами следует упомянуть сравнительно новые идеи использования сжатия промежуточных данных в процессе вычислений, основанные на алгоритмах малоранговой аппроксимации [1, 2].

Пакет USPARS представляет собой еще один элемент этого ряда. Его функциональность предназначена для использования на многоядерных вычислительных системах с общей памятью при решении СЛАУ методом Гаусса. Пакет может быть использован как в In-Core (IC) моде, когда все промежуточные данные в процессе вычислений размещаются в оперативной памяти вычислительной системы, так и в Out-Of-Core (OOC) моде с возможностью сохранения части промежуточных данных на диске. Использование дисковой памяти для временного хранения промежуточных результатов в OOC моде позволяет несколько увеличить размеры СЛАУ, доступных для решения на заданной вычислительной системе. Следует, однако отметить, что из-за включения в процесс решения обменов данными между оперативной памятью и диском платой за возможность увеличения размеров будет рост вычислительного времени.

Базовая версия USPARS распространяется бесплатно при условии некоммерческого использования. Она является ограниченной по поддержке и функциональности, в частности OOC доступен только в платной версии. Функционал, который не доступен бесплатно, отмечен в данном руководстве серым цветом.

Адрес поддержки: uspars-dev@unipro.ru

Структура документа

Матрицы коэффициентов уравнений считаются представленными в формате CSR (Compressed Sparse Row) компактного представления разреженных матриц. Описание этого типа можно найти в главе 1. Там же вводится понятие портрета разреженной матрицы, приведена сводка определений специфических типов данных, используемых в пакете USPARS.

Функциональность USPARS разбита на два уровня. Основные вычислительные функции пакета принадлежат нижнему, скрытому от пользователя, уровню. Предназначения этих функций узко специализированы, а интерфейсы не документированы. Пользователям доступен верхний уровень с набором функций, описанном в главе 2.

Документ дополнен Приложениями:

- Приемы работы с функциональностью USPARS проиллюстрированы в примерах в главе A;
- Математический минимум, лежащий в основе алгоритмов пакета – см. B;
- Практические рекомендации по работе с пакетом можно найти в C;
- Python-интерфейсы к функциям USPARS – см. D.
- Помощь в переходе с MKL PARDISO на USPARS – см. E.

1. Определения и типы данных

В главе приведены определения, используемые далее в тексте.

1.1 Формат CSR хранения разреженных матриц

Матрицы, в которых число ненулевых элементов значительно меньше общего числа элементов, принято называть *разреженными*[1]. Аббревиатура CSR расшифровывается как *Compressed Sparse Row*. При записи матрицы в этот формат происходит 'сжатие' разреженных строк. Для того, чтобы произвести сжатие строки, нужно знать ее длину, число ненулевых элементов в ней, позиции ненулевых элементов и их значения. Позиции ненулевых элементов в строке — это целые индексы соответствующих столбцов, значения элементов — числа с плавающей точкой.

Формат CSR является одним из наиболее популярных форматов хранения разреженных матриц, используемых в библиотеках по работе с разреженными матрицами. В частности, формат используется в компоненте PARDISO библиотеки Intel® MKL. Формат CSR в пакете USPARS использует три массива, но также известна разновидность CSR, в которой используются четыре массива.

1.1.1. Разреженная матрица общего положения

Для представления разреженной матрицы общего положения[2] в формате CSR нужны три линейных массива. Для матрицы A порядка `N`, имеющей `NNZ` ненулевых элементов, эти массивы таковы:

- `ia` – целочисленный массив длины `N+1`;
- `ja` – целочисленный массив длины `NNZ`;
- `a` – вещественный или комплексный массив длины `NNZ`. Его точность представления (`float` или `double`) определяет точность представления матрицы.

В зависимости от используемых интерфейсов переменные `N, NNZ,` как и элементы массивов `ia` и `ja` могут иметь типы `int` или `long long int`, Ненулевые элементы матрицы упакованы в перечисленные три массива следующим образом. Первый элемент `ia[0]` массива `ia` всегда равен нулю, `ia[1]` равен числу ненулевых элементов в первой строке матрицы. `k=ia[1]- ia[0]` ненулевых элементов первой строки помещены в качестве первых `k` элементов массива `a` в том же порядке, как они появляются в строке. Их столбцовые индексы помещаются в первые `k` элементов массива `ja`. Процесс повторяется для второй строки матрицы – столбцовые индексы ненулевых элементов упаковываются в массив `ja` в порядке их появления в строке матрицы, соответствующие значения ненулевых элементов попадают в `a`, значение `ia[2]` становится равным числу ненулевых элементов в первых двух строках.

Продолжая процесс для оставшихся строк матрицы, ненулевые элементы матрицы упаковывают в три массива. Нетрудно понять, что последний элемент `ia[N]` равен `NNZ`. Отметим, что значения элементов массива `ja` оказываются упорядоченными внутри каждой строки. Это требование обязательно для корректной работы функциональности. Разность

$$\texttt{ia[i+1]} - \texttt{ia[i]}, \quad (\texttt{i} = 0, 1, \ldots, \texttt{N-1})$$

всегда равна числу ненулевых элементов матрицы A, лежащих в `i`-ой строке. По этому правилу если `ia[i+1] = ia[i],` то в i-ой строке матрицы все элементы равны нулю. Соответствие между ненулевыми элементами матрицы и элементами массива `a` оказывается установленным следующей формулой

$$\texttt{a[k]} = A_{i,ja[k]} \quad \text{для} \quad \texttt{ia[i]} \le \texttt{k} < \texttt{ia[i+1]}, \quad (\texttt{i} = 0, 1, \ldots, \texttt{N-1}).$$

[1] Иногда разреженными называют матрицы, если существуют методы решения СЛАУ с такими матрицами коэффициентов, в которых учет структуры ненулевых элементов приводит к выигрышу в памяти и времени решения.

[2] Хранение симметричных (вещественных и комплексных) и комплексных эрмитовых матриц описано в подразделе 1.1.2

Замечание. В приведенном описании все индексы, включая индексы элементов матрицы подчиняются правилам языка Си: начинаются с нуля (0-based indices — индексы по базе 0). Далее в этом документе используется предположение об индексах по базе нуль.

Тройку массивов (`ia, ja, a`), используемой для описания разреженной матрицы A в формате CSR, будем называть *CSR-матрицей* A. Для будущих ссылок приведем в явном виде условия, которым должны удовлетворять элементы индексных массивов:

1. Упорядоченность `ia`

$$0 = ia[0] \le ia[1] \le ia[2] \le \ldots \le ia[n-1] \le ia[N] = NNZ. \qquad (1.1)$$

2. Упорядоченность `ja` в строке

$$0 \le ja[k] < ja[k+1] < N \text{ для } ia[i] \le k < k+1 < ia[i+1],\ 0 \le i < N. \qquad (1.2)$$

Приведем пример представления матрицы

$$\begin{pmatrix} 1 & 0 & 5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 8 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad (1.3)$$

в формате CSR. Содержимое массивов представлено в следующей таблице

```
ia = 0,  2,  4,  6,  7,  10, 12;
ja = 0,  2,  1,  3,  0,  2,  3,  1,  4,  5,  4,  5;
 a = 1., 5., 2., 8., 1., 3., 1., 3., 2., 1., 1., 1.
```

Формально говоря, некоторые из элементов матрицы, которые представлены в CSR-формате, совсем не обязательно должны быть нулевыми. Например, следующая тройка массивов

```
ia = 0,  2,  5,  7,  9,  12, 14;
ja = 0,  2,  1,  3,  4,  0,  2,  1,  3,  1,  4,  5,  4,  5;
 a = 1., 5., 2., 8., 0., 1., 3., 0., 1., 3., 2., 1., 1., 1.
```

представляет все ту же матрицу (1.3), но теперь ее так называемый портрет (см. раздел 1.2) симметричен. Для этих целей два нулевых элемента матрицы были внесены в семейство 'ненулевых'.

1.1.2. Хранение симметричной разреженной матрицы в CSR формате

При наличии у матрицы свойства симметрии (симметричность $A_{ij} = A_{ji}$ вещественной или комплексной матрицы, или эрмитовость $A_{ij} = \overline{A_{ji}}$ комплексной матрицы) все ее элементы однозначно восстанавливаются по элементам любого (верхнего или нижнего) из треугольников матрицы. В пакете USPARS для представления таких матриц выбран верхний треугольник, включая диагональ.

Пусть симметричная матрица A порядка N должна быть представлена в формате CSR. В описание, представленное в разделе 1.1.1 требуется внести небольшие коррективы. Для симметричных матриц `NNZ` обозначает число ненулевых элементов матрицы A, лежащих в ее верхнем треугольнике, включая диагональ.

Упаковка элементов верхнего треугольника матрицы в три массива CSR происходит аналогично описанию из предыдущего раздела. Разность

$$ia[i+1] - ia[i],\ (i=0, 1, \ldots, N-1)$$

равна числу ненулевых элементов матрицы A, лежащих в `i`-ой строке верхнего треугольника, включая диагональ.

Условия на элементы индексных массивов, представляющих симметричную либо эрмитову матрицу (ср. (1.1), (1.2)):

3. Упорядоченность `ia`

$$0 = ia[0] \le ia[1] \le ia[2] \le \ldots \le ia[N-1] \le ia[N] = NNZ; \qquad (1.4)$$

4. Упорядоченность `ja` в строке и принадлежность верхнему треугольнику

$$i \le ja[k] < ja[k+1] < N \text{ для } ia[i] \le k < k+1 < ia[i+1],\ 0 \le i < N. \qquad (1.5)$$

Завершим раздел примером представления симметричной матрицы

$$\begin{pmatrix} 4 & 0 & 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & -3 & -2 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 1 & 0 & 2 \\ 0 & -2 & 0 & 0 & 4 & 0 \\ 0 & 0 & 1 & 2 & 0 & 8 \end{pmatrix}$$

в формате CSR

```
ia = 0,  2,  5,  6,  8,  9, 10;
ja = 0,  2,  1,  3,  4,  2,  3,  5,  4,  5;
 a = 4., 2., 3.,-3.,-2., 1., 1., 2., 4., 8.
```

Замечание Для порядка матрицы `N`, числа ненулевых элементов `NNZ`, элементов массивов `ia` и `ja` должна быть возможность их корректного представления. В случае использования 32-битных целых эти параметры должны не превосходить величины $2^{31} = 2147483648$. Легко понять, что при увеличении размеров матриц наиболее быстро возникает ограничение на число ненулевых элементов. Этот параметр не является независимым в описании матриц. однако значения элементов массива `ia` могут достигать числа ненулевых элементов. Для преодоления возникающего затруднения в USPARS могут быть использованы 64-битные целые (см раздел 2.2.)

1.2. Портреты разреженных матриц

Так называемые *портреты разреженных матриц* являются удобным инструментом для визуализации структуры ненулевых элементов в матрице. На Рис 1.2.1 изображены такие портреты для случая матриц невысокого порядка. Точки представляют собой позиции ненулевых элементов в матрице. При генерации портретов мы предполагали, что элементы диагонали отличны от нуля, и это нашло явное отражение на рисунках.

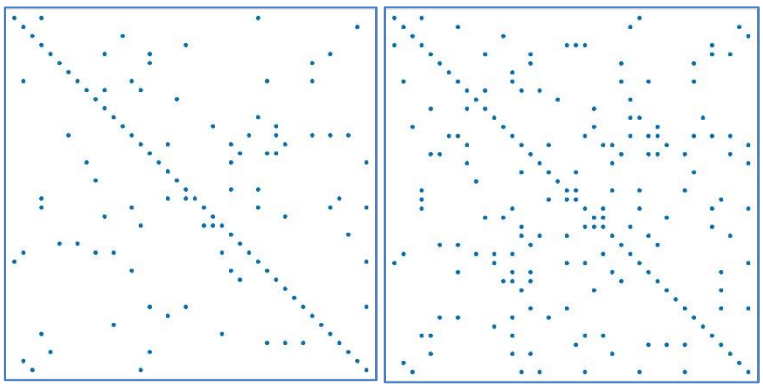

Рис 1.2.1 Слева: несимметричный портрет разреженной матрицы.
Справа: симметризация портрета, изображенного слева.

На портрете справа точки расположены симметрично относительно диагонали матрицы, в то время как портрет слева таким свойством не обладает. Говорят, что матрица, портрет которой изображен справа, обладает *симметричным портретом*. Это не означает, что она сама симметрична — для этого нужно, чтобы свойством симметрии обладали элементы матрицы в симметричных позициях.

1.3. Специальные типы данных пакета USPARS

В списке аргументов функций пакета USPARS указатели на массивы чисел с плавающей точкой приведены к типу `void*`.

Для передачи в функцию информации о фактическом типе элементов массива используется дополнительный параметр перечисляемого типа `usp_scalar_type`, список возможных значений которого представлен в Таблице 1.3.1.

Таблица 1.3.1: Возможные значения параметров типа `usp_scalar_type`

Значение	**Соответствие стандартному типу**
`USP_FLOAT`	`float`
`USP_DOUBLE`	`double`
`USP_COMPLEX_FLOAT`	`_Complex float`
`USP_COMPLEX_DOUBLE`	`_Complex double`

В пакете USPARS применяется несколько типов треугольной факторизации разреженных матриц. Тип факторизации определяется фактом принадлежности матрицы тому или иному классу (см раздел B.1). Этому типу поставлена в соответствие переменная перечисляемого типа `usp_factorize_type` со значениями, представленными в Таблице 2.2.2

Функции `uspars_param_get()` и `uspars_param_set()` (см разделы 2.5, 2.6) предназначены для извлечения некоторых данных, полученных в результате вычислений, или редактирования значений параметров, управляющих ходом вычислительного процесса. Соответствующий список определяется переменной перечисляемого типа `usp_param_data` (см Таблицы 2.5.2 и 2.6.2).

1.4. Перестановки

В теории и практике алгоритмов для СЛАУ с разреженными матрицами коэффициентов часто используются матрицы перестановок. Квадратная $N \times N$ −матрица P называется *матрицей перестановок*, если в каждой ее строке и в каждом столбце имеется в точности по одному ненулевому элементу, и все ненулевые элементы равны единице. Иначе говоря, матрица P получается в результате перестановки **строк** единичной матрицы. Этот факт можно записать в виде

$$P_{ij} = \begin{cases} 1 & \text{для } j = p_i \\ 0 & \text{для } j \neq p_i \end{cases}, \qquad (j = 1,2,\dots,N), \qquad (1.6)$$

где использовано понятие *вектора перестановок* p с компонентами p_i. Ясно, что вектор p полностью определяет матрицу P. Умножение матрицы перестановок P на вектор a

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = P \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}$$

дает вектор b с компонентами

$$b_i = a_{p_i}, \qquad (i = 1,2,\dots,N), \tag{1.7}$$

где компоненты вектора перестановок p определены формулами (1.6). Если матрица перестановок умножается справа на строку

$$[d_1 \quad d_2 \quad \cdots \quad d_N] = [c_1 \quad c_2 \quad \cdots \quad c_N]P,$$

то соответствующие формулы имеют вид

$$d_j = c_{q_j}, \qquad (j = 1,2,\dots,N), \tag{1.8}$$

где вектор перестановок q определен матрицей перестановок P^t, которая является обратной к P:

$$PP^t = I.$$

Из формул (1.7), (1.8) легко получить формулы для элементов матриц AP и PA. Такие произведения возникают при использовании так называемого matching'а и pivoting'а (см разделы B.3 и C.3, соответственно). Наконец, с целью уменьшения заполненности треугольных факторов, получаемы на этапе факторизации, применяются так называемые *симметричные перестановки* $\tilde{A} = PAP^t$ матриц. Элементы матриц A и $\tilde{A}$ связаны следующими формулами

$$\tilde{a}_{ij} = a_{p_i p_j} \qquad (1 \le i, j \le N). \tag{1.9}$$

2. Пользовательская функциональность

Среди аргументов каждой из функций пользовательского уровня пакета USPARS имеется специальный аргумент (типа void**). В кодах функций тип этой переменной приводится к типу указатель-на-указатель на переменную типа structure (для краткости речи называемую USP-структурой), поля которой хранят указатели на массивы исходных и промежуточных данных. Эти данные не раскрываются для пользователя.

Для того, чтобы решить систему линейных уравнений с разреженной матрицей коэффициентов, нужно последовательно вызвать четыре функции пакета USPARS:

1. `uspars_alloc()` создает USP-структуру для передачи данных между функциями;
2. `uspars_init?()` производит подготовку к факторизации матрицы коэффициентов;
3. `uspars_fact()` производит собственно факторизацию матрицы коэффициентов;
4. `uspars_solve()` производит исключение неизвестных.

Особо отметим, что указанный порядок

`uspars_alloc()` ⇒ `uspars_init?()` ⇒ `uspars_fact()` ⇒ `uspars_solve()` (2.1)

вызова функций предопределен тем, что каждая функция в этой последовательности подготавливает входные данные для следующей функции. Нарушение порядка (2.1) вызова функций приводит к специфической ошибке. Для передачи данных используется USP-структура, и внешне это никак не проявляется.

Замечание 2.0.1. *Если требуется решить несколько систем линейных уравнений с одной и той же матрицей коэффициентов, но разными правыми частями, то это можно сделать после того, как отработала функция* `uspars_fact()`, *вызвав функцию* `uspars_solve()` *несколько раз с соответствующими векторами правых частей.*

Каждая из перечисленных функций возвращает код завершения. Значение кода сигнализирует о том, что вычисления завершились корректно, либо об ошибке. Во избежание нештатного поведения прежде, чем вызывать следующую функцию **настоятельно** рекомендуется проанализировать код завершения.

2.1. uspars_alloc(): создание USP-структуры

Синтаксис

```
int uspars_alloc(void **tt)
```

Include

```
uspars.h
```

Описание

Функция служит для создания USP-структуры **tt, используемой функциями пакета для обмена данными. Примеры использования функции uspars_alloc() см в директории examples.

Коды возврата

Код	Что означает
0	Успешное завершение
3	Ошибка выделения памяти

2.2. uspars_init?(): подготовка к факторизации матрицы коэффициентов

Синтаксис

```
int uspars_init(void **tt, int n, int *ia, int *ja, void *a,
                usp_scalar_type utype, usp_factorize_type ftype,
                int *options)

int uspars_init32(void **tt, int n, int *ia, int *ja, void *a,
                usp_scalar_type utype, usp_factorize_type ftype,
                int *options)

int uspars_init3264(void **tt, int n, long long int *ia, int *ja,
                void *a, usp_scalar_type utype, usp_factorize_type ftype,
                int *options)

int uspars_init64(void **tt, long long int n, long long int *ia,
                long long int *ja, void *a, usp_scalar_type utype,
                usp_factorize_type ftype, int *options)
```

Include

```
uspars.h
```

Описание

Функции uspars_init?() предназначены для подготовки к факторизации матрицы коэффициентов системы линейных уравнений

$$Ax = f.$$

Отметим, что функции `uspars_init` и `uspars_init32` эквивалентны и пользователь может использовать любую из них.

До вызова функции `uspars_init?()` должна быть вызвана функция `uspars_alloc()` (см раздел 2.1), которая создает USP-структуру `**tt`. Последовательность действий `uspars_init?()` представлена в следующем списке.

a) Указатели на массивы, определяющие матрицу коэффициентов в формате CSR (см раздел 1.1), заносятся в соответствующие поля USP-структуры `**tt`.
b) Балансировка (в англоязычной литературе могут применяться термины *scaling* или *balancing*) матрицы коэффициентов путем умножения ее на специально подобранные диагональные матрицы (см подраздел B.2). Балансировка может быть включена или выключена (о способах переключения см описание параметра `options[USP_SCALING]`). Взамен матрицы A возникает матрица
$$A_1 = \begin{cases} D_l A D_r & \text{балансировка включена;} \\ A & \text{балансировка выключена.} \end{cases}$$
c) Перестановка строк матрицы (см подраздел B.3) с целью увеличить минимум абсолютных значений ее диагональных элементов. Такая операция называется *matching* и применяется только для матриц общего вида (`ftype == USP_LU`) (о том, как она включается, см описание параметра `options`). Формула для описания этой операции
$$A_2 = \begin{cases} D_{lm} P_m A_1 D_{rm} & \text{matching с балансировкой} \\ P_m A_1 & \text{matching} \\ A_1 & \text{без использования matching} \end{cases}$$
Здесь P_m обозначает матрицу перестановок.

d) Симметризация портрета матрицы коэффициентов.
e) Симметричная перестановка строк и столбцов матрицы (алгоритм перестановки определяется параметром `options[USP_RTYPE]`) с целью уменьшения степени заполнения ненулевыми элементами треугольных факторов. В результате получается матрица $A_3 = PA_2P^t$ ($P = I$, если не используются перестановки).
f) Символьная факторизация матрицы A_3. В результате становится определенным портрет разреженной матрицы L (и матрицы U в случае LU-разложения), и, соответственно объем требуемой памяти для хранения ее ненулевых элементов. Выделение памяти под массивы промежуточных данных и ненулевые элементы треугольных факторов. Указатели на эти массивы сохраняются в полях структурной переменной `tt`.

Таблица 2.2.1: Аргументы функций `uspars_init?()`

тип	имя	Назначение
in/out	`tt`	Указатель-на-указатель на USP-структуру
input	`n`	Порядок матрицы, >0
input	`ia, ja, a`	Тройка массивов, определяющих CSR-матрицу (см раздел 1.1) коэффициентов A Тип указателя `a` приведен к типу `void*`, информация об истинном типе элементов этого массива передается через значение параметра `utype`
input	`utype`	Информация о типе ненулевых элементов матрицы (см Таблицу 1.3.1)
input	`ftype`	Информация о типе факторизации матрицы (см Таблицу 2.2.2)
input	`options`	Массив (длиной `USP_OPTIONS`) опций, используемых функциями пакета USPARS. Значение параметра `USP_OPTIONS` определено в `uspars.h`

*) Индексы элементов матрицы, элементы массивов `ia`, `ja` представляют собой сдвиги относительно начал соответствующих массивов и начинаются с нуля (C-шная нумерация).

**) Элементы массивов ia, ja удовлетворяют условиям (1.1), (1.2) в общем случае, условиям (1.4), (1.5) в симметричном или эрмитовом случае.

Таблица 2.2.2: Возможные значения параметров типа `usp_factorize_type`

Значение	Смысловое наполнение
`USP_LLH`	Разложение Холецкого (см (B.4)) вещественной симметричной или комплексной эрмитовой положительно определенной матрицы
`USP_LDLT`	LDLT-разложение симметричной (вещественной или комплексной) матрицы (B.12)
`USP_LDLH`	LDLH-разложение комплексной эрмитовой матрицы (B.12)
`USP_LU`	LU-разложение (вещественной или комплексной) матрицы общего вида (B.16)

Список *допустимых значений* элементов массива `options`:

- `options[USP_MSGLVL]`
  - 0: 'молчаливые' вычисления (никакой печати на экран не производится).
  - 1: средний уровень 'говорливости' вычислений — выполнение некоторых функций сопровождается выдачей на экран (***значение по умолчанию***)[3].
  - 2: максимальный уровень 'говорливости' (функции сообщают о входе-выходе, об ошибочных ситуациях, ...)[4].
- `options[USP_SCALING]` определяет, включать или нет балансировку матрицы коэффициентов (см раздел B.2). Значение этого параметра принимается во внимание только при использовании LU-, LDLT- и LDLH-факторизаций. В случае LLH-факторизации значение этого параметра игнорируется, балансировка не применяется.
  - 0: не использовать балансировку ***(значение по умолчанию для LLH)***.
  - 1: использовать балансировку (***значение по умолчанию для LU, LDLT, LDLH***).
- `options[USP_MATCHING]` определяет, включать или нет matching (см раздел B.3). Значение этого параметра принимается во внимание только при использовании LU-факторизации, (раздел B.3). В противном случае значения этого параметра игнорируются.
  - 0: не использовать matching (***значение по умолчанию для LLH, LDLT, LDLH***).
  - 1: использовать matching с максимизацией минимального элемента на диагонали (***значение по умолчанию для LU***).
  - 2: использовать matching с максимизацией произведения диагональных элементов
  - 3: использовать matching с максимизацией произведения диагональных элементов и последующей балансировкой матрицы таким образом, чтобы диагональные элементы были равны 1, а все остальные были меньше, либо равны 1.
- `options[USP_RTYPE]` определяет перестановку строк и столбцов матрицы, применяемую для сокращения числа ненулевых элементов в треугольных факторах. Перестановка определяется *вектором перестановки*, который может быть задан пользователем, вычисляется с помощью специализированных программ либо не используется вовсе в зависимости от значения `options[USP_RTYPE]`:
  - 0: вектор перестановки не используется
  - 1: вектор перестановки рассчитывается функциями USPARS-компоненты Atlant (***значение по умолчанию***);
  - 2: вектор перестановки вычисляется с помощью Metis, если пакет установлен (libmetis.so для Linux, metis.dll для Windows). В противном случае значение опции переключается в 1 и вычисление вектора перестановки происходит с помощью компоненты Atlant;
  - 3: вектор перестановки задается пользователем см раздел (2.6).
- `options[USP_ITER_REF]` - ограничение числа шагов итерационного уточнения (см раздел C.2). Если `options[USP_ITER_REF] == 0`, то итерационное уточнение не используется (***значение по умолчанию***). Чтобы итерационное уточнение включилось, значение `options[USP_ITER_REF]` должно быть положительным. Итерационное уточнение

[3] Например, включение отображения на экране «индикатора выполнения» (progress-bar) самого трудоемкого этапа – факторизации. В случае использования USPARS-OOC выводится информация о включенной моде.

[4] Этот уровень может быть полезен при общении с разработчиками пакета USPARS в случае возникновения вопросов по конкретному исполнению программы путем предоставления выдаваемой информации.

останавливается по достижению ограничения числа шагов либо, если выполнен критерий остановки по относительной невязке (C.1). ***Значения по умолчанию*** параметра α в (C.1):

- `1e-6` для обычной точности;
- `1e-14` для двойной точности,

Для изменения значения параметра α следует использовать вызов функции

```
uspars_param_set(&t, USP_ITER_STOP_CRIT, &value);
```

где `t` указатель на USP-структуру, `value` – число типа double, устанавливаемое значение параметра.

- `options[USP_BOOSTING]`– включение диагональной поддержки (см раздел C.3.1) при вычислении LDLT-, LDLH-, и LU-факторизаций, чтобы избежать возникновения ошибки `Zero Pivot (см раздел C.3)`.
  - 0: поддержка не включена;
  - 1: включить диагональную поддержку для нулевых элементов на диагонали. (***значение по умолчанию***);
  - 2: включить усиленную диагональную поддержку.

  При включенной диагональной поддержке некоторые из диагональных элементов в процессе факторизации модифицируются. Для того, чтобы определить какие элементы подвергаются модификации, используется 'уровень малости' δ_{boost} (по умолчанию 10^{-15} для `options[USP_BOOSTING]`=1 и 10^{-8} для `options[USP_BOOSTING]`=2). Чтобы установить значение δ_{boost}, следует использовать вызов функции

  ```
  uspars_param_set(&t, USP_BOOSTING_VALUE, &value),
  ```

  где `t` указатель на соответствующую USP-структуру, `value` – число типа double, устанавливаемое значение δ_{boost}. Если включена диагональная поддержка и не включено итерационное уточнение, то оно автоматически включается на 2 итерации для `options[USP_BOOSTING]`=1 и на 4 итерации для `options[USP_BOOSTING]`=2.

- `options[USP_GLOBAL_PIVOT]` –включение глобальных перестановок (см раздел C.3.2) при вычислении LDLT-, LDLH-, и LU-факторизаций, чтобы избежать возникновения ошибки Zero Pivot.

  Если `options[USP_GLOBAL_PIVOT] == 0`, то глобальные перестановки выключены (***значение по умолчанию***). Чтобы включить глобальные перестановки, значение `options[USP_GLOBAL_PIVOT]` должно быть равно 1.

  Значение по умолчанию параметра малости δ_{gp} (ср (C.5)) равно `1e-8`. Чтобы его изменить, следует использовать вызов функции

  ```
  uspars_param_set(&t, USP_GP_EPS, &value),
  ```

  где `t` указатель на соответствующую USP-структуру, `value` – число типа double, требуемое значение параметра δ_{gp}.

- `options[USP_BRX]` при включенном итерационном уточнении позволяет использовать в качестве решения массив, полученный на итерации с лучшей относительной невязкой по евклидовой норме.
  - 0: решением является массив, полученный на последней итерации (***значение по умолчанию***);
  - 1: решением является массив, полученный на итерации с наименьшей относительной невязкой

- `options[USP_OOC_MEM]` позволяет переключать моды - In-Core (IC) либо Out-Of-Core (OOC), в которых работает пакет.
  - 0: IC мода. Дисковая память не используется (***значение по умолчанию***);

- >0: Native OOC мода. Значение `options[USP_OOC_MEM]` представляет собой ограничения на объем требуемой оперативной памяти (в гигабайтах). Если оказалось, что объем требуемой меньше этого числа, все промежуточные данные сохраняются в оперативной памяти. Если же объем требуемой памяти превосходит это ограничение, то для хранения промежуточных результатов используется дисковая память, и объем используемой оперативной памяти остается в заданных пределах.
- < -1: Forced OOC мода. В этой моде независимо от требуемого объема памяти происходит запись промежуточных данных на диск, но объем оперативной памяти, используемой в процессе факторизации, не превышает (`-options[USP_OOC_MEM]`) гигабайт.

Замечание 2.2.1. *Если значение какого-то элемента массива* `options` *равно -1 (или не соответствует описанным выше вариантам), то в процессе вычислений это значение подменяется значением по умолчанию.*

Коды возврата

Код	Что означает
0	Успешное завершение
1	`*tt==NULL`. Проверить, что `uspars_alloc()` была вызвана до вызова `uspars_init()` и ее код возврата равен 0
2	Не выполнены условия на входные данные (см список ниже)
3	Ошибка выделения памяти. Проверить размеры массивов
9	Попытка использовать недоступный функционал
999	Неклассифицируемая внутренняя ошибка

Условия на входные данные для функций `uspars_init?()` представлены в списке:

- `n > 0`;
- Указатели `ia, ja, a` корректно определены;
- Диагональные элементы комплексной эрмитовой матрицы вещественны;
- Элементы массива `ja` удовлетворяют условиям (1.1), (1.2) ((1.4), (1.5) в симметричном случае).

2.3. `uspars_fact()`: Треугольная факторизация разреженных матриц

Синтаксис

```
int uspars_fact(void **tt)
```

Include

```
uspars.h
```

Описание

Функция предназначена для вычисления треугольных факторизаций (ср. (В.7), (В.12), (В.15))

$A = LL^T$ при $A \in R^{n\times n}$ $A = A^T > 0$ `ftype == USP_LLH;` (2.2)

$A = LDL^T$ при $A \in R^{n\times n}$ $A = A^T$ `ftype == USP_LDLT;` (2.3)

$A = LL^H$ при $A \in C^{n\times n}$ $A = A^H > 0$ `ftype == USP_LLH;` (2.4)

$A = LDL^H$	при	$A \in C^{n\times n}$	$A = A^H$	`ftype`	`==`	`USP_LDLH;` (2.5)
$A = LDL^T$	при	$A \in C^{n\times n}$	$A = A^T$	`ftype`	`==`	`USP_LDLT;` (2.6)
$A = LU$	при	A – квадратная матрица общего вида		`ftype`	`==`	`USP_LU.` (2.7)

Порядок матрицы `n` указатели на массивы (`ia, ja, a`), служащие для представления марицы A в формате CSR, хранятся в полях структурной переменной, указываемой `*tt`, которые были предварительно заполнены в процессе работы функции `uspars_init?()`. USP-структура `**tt` одновременно является и выходным параметром функции `uspars_fact()`– массивы с результатами факторизации также передаются через поля `**tt`.

Коды возврата

Код	Что означает
0	Успешное завершение
1	`*tt==NULL`. Проверить корректность кода.
3	Ошибка выделения памяти. Проверить размеры массивов
4	‘Zero pivot’ (см разделы C.3.1, C.3.2)
6	Нарушена последовательность (2.1) вызова функций
999	Неклассифицируемая внутренняя ошибка

2.4. `uspars_solve()`: Решение систем линейных уравнений с факторизованной матрицей коэффициентов

Синтаксис

```
int uspars_solve(void **t, int m, void *b, usp_scalar_type utype)
```

Include

`uspars.h`

Описание

Функция предназначена для решения системы линейных уравнений

$$AX = B \tag{2.8}$$

с разреженной $n \times n$-матрицей коэффициентов A и m векторов правых частей, представленными в (2.8) прямоугольной $n \times m$ –матрицей B. Матрица коэффициентов предварительно факторизована с помощью вызова функции `uspars_fact()`.

Таблица 2.4.1: Аргументы функции `uspars_solve()`

тип	имя	Назначение
input	`tt`	Указатель-на-указатель на USP-структуру
input	`m`	Число правых частей

in/out	`B`	Массив[5] (длины $n*m$) чисел с плавающей точкой. На входе массив содержит компоненты векторов-столбцов правых частей, располагающихся в памяти по столбцам (j-я компонента k-го вектора имеет адрес B+k*n+j). На выходе подобным же образом в массиве `B` располагаются компоненты соответствующих векторов неизвестных (столбцов матрицы X)
input	`utype`	Информация о фактическом типе элементов матрицы `B`. Значение параметра должно совпадать со значением параметра `utype`, который определял тип элементов матрицы коэффициентов при вызове функции `uspars_init?()`

Коды возврата

Код	Что означает
0	Успешное завершение
1	`*tt==NULL`. Проверить корректность кода.
2	m<0, или `b==NULL` или `utype` не совпадает с типом матричных элементов
3	Ошибка выделения памяти. Проверить размеры массивов
6	Нарушена последовательность (2.1) вызова функций
7	Итерационное уточнение решения (см раздел C.2) не сошлось
999	Неклассифицируемая внутренняя ошибка

2.5. `uspars_param_get()`: доступ к некоторым полям `tt`

Синтаксис

```
int uspars_param_get(void **tt, usp_param_data id, void *a)
```

Include

```
uspars.h
```

Описание

Функция служит для извлечения данных, хранящихся в полях структурной переменной, указываемой указателем `*tt`. Не все эти данные доступны пользователю. Извлекаемые данные специфицируются значением входного параметра `id` (см Таблицу 2.5.2)

Таблица 2.5.1: Аргументы функции uspars_param_get()

тип	имя	имя
input	`tt`	Указатель-на-указатель на USP-структуру
input	`id`	Параметр для спецификации извлекаемых данных
output	`a`	Указатель на массив, куда поместить извлекаемую информацию. Память под элементы массива должна быть выделена до вызова функции `uspars_param_get()`

Таблица 2.5.2: Возможные значения аргумента id функции uspars_param_get()

Значение id	Спецификация данных
`USP_ERR`	код ошибки

[5] Указатель приведен к типу `void*` для универсализации интерфейса, информация о настоящем типе элементов массива `b` передается через параметр `utype`.

USP_NNZL	число ненулевых элементов L-фактора[6]
USP_PERM_VECTOR	вектор перестановок для переупорядочивания элементов исходной матрицы перед факторизацией
USP_MEM_FCT	Параметр для получения информации об оценке объема памяти (в байтах), требуемой для факторизации

Коды возврата

Код	Что означает
0	Успешное завершение
1	*tt==NULL. Проверить корректность кода.
2	некорректные значения аргументов

2.6. uspars_param_set(): редактирование полей tt

Синтаксис

```
int uspars_param_set(void **tt,  usp_param_data id, void *a)
```

Include

```
uspars.h
```

Описание

Функция служит для редактирования информации, хранящейся в полях структурной переменной, указываемой указателем *tt. Тип информации определяется значением входного параметра id (см Таблицу 2.6.2

Таблица 2.6.1: Аргументы функции uspars_param_set()

тип	имя	имя
input	tt	Указатель-на-указатель на USP-структуру
input	id	Что редактировать
output	a	Указатель на массив с данными, которые должны заменить имеющиеся

Таблица 2.6.2: Возможные значения аргумента id функции uspars_param_set()

Значение id	Спецификация данных
USP_BOOSTING_VALUE	число с двойной точностью – уровень допуска для ’поддержки’ (см раздел C.3.1)
USP_ITER_STOP_CRIT	число с двойной точностью – критерий остановки итерационного уточнения (см раздел C.2)
USP_GP_EPS	число с двойной точностью – параметр малости, используемый в алгоритме глобальных перестановок (см раздел C.3.2)
USP_PERM_VECTOR	вектор перестановок, использованный для переупорядочивания элементов исходной матрицы перед факторизацией

[6] Если включена options[USP_GLOBAL_PIVOT], возвращаемое число ненулевых элементов в L-факторе может быть неверным

Коды возврата

Код	Что означает
0	Успешное завершение
1	`*tt==NULL`. Проверить корректность кода.
2	некорректные значения аргументов

2.7. `uspars_destroy()`: разрушение USP-структуры

Синтаксис

```
int uspars_destroy(void **tt)
```

Include

```
uspars.h
```

Описание

Функция служит для освобождения памяти, захваченной на шаге инициализации. На выходе из этой функции значение указателя `*tt` равно `NULL`.

Коды возврата

Код	Что означает
0	Успешное завершение
1	`*tt==NULL`. Проверить корректность кода.

3. Коды ошибки

3.1. Формирование ошибок

Функции USPARS возвращают целое, которое трактуется как код возврата. Возможные значения кодов возврата с соответствующими интерпретациями приведены в описаниях функций.

Коды возврата формируются по значениям кодов ошибок, которые могут возникать в процессе вычислений и которые содержат более детальную информацию о случившейся ошибке. Эта более детальная информация может быть полезной при исследовании проблемы, возникшей в процессе использования функций USPARS. Для извлечения такой информации служит функция uspars_param_get() (см раздел 2.5). В этом разделе мы приводим таблицы кодов ошибок.

3.2. Ошибки входных данных (код возврата 2)

Числовое значение	Интерпретация
201	Параметр `N` не положителен

202	Указатель ia CSR-матрицы равен `NULL`
203	Указатель ja CSR-матрицы равен `NULL`
204	Указатель a CSR-матрицы равен `NULL`
205	Невещественная диагональ матрицы, объявленной эрмитовой
206	Несовпадение типов элементов матрицы коэффициентов и вектора правой части
207	Параметр `NRHS` не положителен
208	Указатель на массив с элементами правой части СЛАУ равен `NULL`
210	Неупорядоченные элементы в массиве `ja` CSR-матрицы
211	Структура входной матрицы не позволяет корректно завершить работу matching
212	Некорректное значение параметра `id` в функции `uspars_param_set()`
213	Некорректное значение параметра `id` в функции `uspars_param_get()`
214	Некорректный реордеринг для OOC. Используйте Atlant

3.3. Ошибки выделения памяти (код возврата 3)

Числовое значение	Интерпретация
301	Ошибки выделения памяти
302	Нехватка памяти для OOC заданной посредством пользовательской функциональности `options[USP_OOC_MEM]`
303	Не существует директории, заданной переменной окружения `USP_OOC_PATH`
304	Невозможно использовать файл с данными, созданный в процессе работы OOC, возможно он поврежден.
305	Не хватает места на диске.

3.4. Ошибки факторизации (код возврата 4)

Числовое значение	Интерпретация
401	Не положительно-определенная входная матрица при использовании разложения Холецкого
402	Нулевой ведущий элемент при выполнении факторизации (Zero pivot)
403	Число глобальных перестановок превысило установленный лимит

3.5. Ошибки при обращении факторов (код возврата 5, зарезервировано для дальнейшего использования)

3.6. Некорректный порядок вызова функций (код возврата 6)

Числовое значение	Интерпретация
601	Некорректный порядок вызова функций: `uspars_fact()` вызвана до `uspars_init?()`
602	Некорректный порядок вызова функций: `uspars_solve()` вызвана до `uspars_init?()`
603	Некорректный порядок вызова функций: `uspars_solve()` вызвана до `uspars_fact()`

3.7. Ошибки фазы решения (код возврата 7)

Числовое значение	Интерпретация
701	Итерационное уточнение не сошлось за заданное число шагов

3.8. Ошибки использования Metis (код возврата 8)

Числовое значение	Интерпретация
801	Динамическая библиотека Metis не найдена
802	Невозможно вызвать Metis в процессе выполнения программы

3.9. Ошибки используемой версии USPARS (код возврата 9)

Числовое значение	Интерпретация
901	Попытка использовать функционал, который не входит в свободно распространяемую версию USPARS

3.10. Неклассифицируемые ошибки (код возврата 999)

Эти ошибки имеют номера[7] от 99900 до 99913.

[7] Эти номера ничего не говорят пользователям, но важны разработчикам

A. Примеры

В директории `examples` располагается исходные коды на языке C примеров использования функций пакета USPARS для решения систем линейных уравнений с разреженными матрицами коэффициентов. Сценарии примеров однотипны:

- Инициализировать матрицу в CSR формате.
- Инициализировать вектор правой части, соответствующий точному решению x_{exct}.
- В соответствии с алгоритмом решения последовательно вызывать функции
  - uspars_init?(),
  - uspars_fact(),
  - uspars_solve().

  Каждая последующая функция вызывается при условии успешного завершения предыдущей, для проверки анализируется код возврата.
- Распечатать векторы точного и численного решения для сравнения.

Разница между точным и численным решениями обусловлена влиянием погрешностей округления. На величину этой разницы влияют как число обусловленности матрицы коэффициентов, так и используемая рабочая точность. Для удобства мы приводим величины относительных погрешностей вычисленных решений вместе со значениями чисел обусловленности матрицы коэффициентов.

A.1. Перестановки и число ненулевых элементов в треугольных факторах

Пример дает сравнительный результат числа ненулевых элементов в матрицах L и U в результате факторизации при использовании различных переупорядочиваний. Величина `nnz(A)` в выводе представляет собой число ненулевых элементов в исходной матрице, а `nnz(L+U)` суммарное в L и U.

Система уравнений образована на матрице https://sparse.tamu.edu/Mathworks/Sieber с числом обусловленности 4.613510e+08. Вектор точного решения состоит из единиц

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2290} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Соответствующая ему правая часть системы линейных уравнений получена умножением матрицы на вектор точного решения.

Ниже приведена таблица результатов для случая решения без предварительного переупорядочивания, с использованием METIS и Atlant:

```
  w/o reorder results:
name              |   n    |nnz(A)   | nnz(L+U) |     error C     |  error L2    |
./mtx/Sieber.mtx  |  2290  |  14873  |  3125286 |   7.622005e-09  | 1.748052e-10|

METIS results:
name              |   n    |nnz(A)   | nnz(L+U) |     error C     |  error L2    |
./mtx/Sieber.mtx  |  2290  |  14873  |    41194 |   2.409236e-09  | 1.068810e-10|

Atlant reorder results:
name              |   n    |nnz(A)   | nnz(L+U) |     error C     |  error L2    |
./mtx/Sieber.mtx  |  2290  |  14873  |    41174 |   1.782625e-09  | 2.290986e-10|
```

Здесь `nnz(A)` обозначает число ненулевых элементов в матрице A, а `nnz(L+U)` – общее число ненулевых элементов в матрицах L и U, `error C` и `error L2` – относительные погрешности решения

$$\frac{\|x - x_{\text{выч}}\|}{\|x\|}$$

в нормах ’максимум модуля’ и евклидовой, соответственно. Исходный код примера см в файле `example_compare_reorder.c`.

A.2. Замер времени работы фаз

В этом примере иллюстрируются способы получения информации о времени выполнения фаз пакета USPARS. В качестве матрицы коэффициентов взята несимметричная матрица https://sparse.tamu.edu/Mathworks/Sieber с числом обусловленности 4.613510e+08. Вектор точного решения состоит из единиц

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2290} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Соответствующая ему правая часть системы линейных уравнений получена умножением матрицы на вектор точного решения. Выдача имеет следующий вид

```
USPARS Init  time: 0.028851 seconds.
USPARS Fact  time: 0.035071 seconds.
USPARS Solve time: 0.001889 seconds.
Results:
name             | n    | nnz(A) |   error C    |  error L2    |
./mtx/Sieber.mtx | 2290 |  14873 | 1.782625e-09 | 2.290986e-10 |
```

В зависимости от используемого компьютера результаты замера времени могут сильно варьировать. Исходный код примера см в файле `example_timer.c`.

A.3. Вещественная симметричная положительно-определенная матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 4 & 0 & -2 & 0 & 0 & 2 \\ 0 & 5 & 0 & -3 & -2 & 0 \\ -2 & 0 & 4 & 0 & 0 & 3 \\ 0 & -3 & 0 & 5 & 0 & 2 \\ 0 & -2 & 0 & 0 & 3.96 & -2 \\ 2 & 0 & 3 & 2 & -2 & 8.16 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 5 \\ 4 \\ -0.04 \\ 13.16 \end{bmatrix}$$

Исходный код примера см в файле `example_LLH_s.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution     Numerical solution
 1.0000000e+00      9.9545372e-01
 1.0000000e+00      9.9966192e-01
 1.0000000e+00      9.9480432e-01
 1.0000000e+00      9.9823838e-01
 1.0000000e+00      1.0017973e+00
 1.0000000e+00      1.0038968e+00
```

Число обусловленности матрицы коэффициентов равно 5.0787e+05, что оправдывает относительное уклонение 0.0034 приближенного решения от точного.

A.4. Вещественная симметричная положительно-определенная матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 4 & 0 & -2 & 0 & 0 & 2 \\ 0 & 5 & 0 & -3 & -2 & 0 \\ -2 & 0 & 4 & 0 & 0 & 3 \\ 0 & -3 & 0 & 5 & 0 & 2 \\ 0 & -2 & 0 & 0 & 3.96 & -2 \\ 2 & 0 & 3 & 2 & -2 & 8.16 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 5 \\ 4 \\ -0.04 \\ 13.16 \end{bmatrix}$$

Исходный код примера см в файле `example_LLH_d.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution            Numerical solution
 1.0000000000000000e+00    1.0000000000038496e+00
 1.0000000000000000e+00    1.0000000000002858e+00
 1.0000000000000000e+00    1.0000000000043996e+00
 1.0000000000000000e+00    1.0000000000014913e+00
 1.0000000000000000e+00    9.9999999999847777e-01
 1.0000000000000000e+00    9.9999999999670042e-01
```

Число обусловленности матрицы коэффициентов равно 5.0787e+05, относительное уклонение 2.88e-12 приближенного решения от точного вполне оправдано.

A.5. Комплексная эрмитова положительно-определенная матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 5.01 & 0 & -2+i & 0 & 0 & 2.09 \\ 0 & 4.99 & 0 & -2.98 & -2+2i & 0 \\ -2-i & 0 & 4 & 0 & 0 & 3.19 \\ 0 & -2.98 & 0 & 5.03 & 0 & 0 \\ 0 & -2-2i & 0 & 0 & 3.99 & -2 \\ 2.09 & 0 & 3.19 & 0 & -2 & 8.98 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 2.1+12.55i \\ -9.99-9.94i \\ 6.19+28.14i \\ 2.05+14.16i \\ -6.01+15.95i \\ 24.26+63.54i \end{bmatrix}$$

Исходный код примера см в файле `example_LLH_c.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution                    Numerical solution
 1.0000000e+00 +  1.0000000e+00*i    1.0010201e+00 +  9.9902976e-01*i
 1.0000000e+00 +  2.0000000e+00*i    1.0015880e+00 +  2.0003891e+00*i
 1.0000000e+00 +  3.0000000e+00*i    1.0017089e+00 +  2.9991865e+00*i
 1.0000000e+00 +  4.0000000e+00*i    1.0009408e+00 +  4.0002303e+00*i
 1.0000000e+00 +  5.0000000e+00*i    1.0009670e+00 +  5.0015926e+00*i
 1.0000000e+00 +  6.0000000e+00*i    9.9880081e-01 +  6.0007300e+00*i
```

Число обусловленности матрицы коэффициентов равно 88.23. Относительная погрешность полученного решения 4.58e-04 с помощью одной итерации уточнения может быть улучшена до 1.85e-06.

A.6. Комплексная эрмитова положительно-определенная матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 2 & 0 & 0.51i & 0 & 0 & 0 & 0.74 \\ 0 & 1 & 0 & 0 & 0.8 & 0 & 0 \\ -0.51i & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & -0.3i \\ 0 & 0.8 & 0 & 0 & 1 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 0.6 & 2 & 1+0.8i \\ 0.74 & 0 & 0 & 0.3i & 0 & 1-0.8i & 2 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 5.65+8.71i \\ 6+2i \\ 3.51+2.49i \\ 10.1-10.1i \\ 10.2-0.2i \\ 16.4+3.6i \\ 17.14+5.14i \end{bmatrix}$$

Исходный код примера см в файле `example_LLH_z.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution                                  Numerical solution
 1.0000000000000000e+00 +  1.0000000000000000e+00*i    9.9999999999754519e-01 +  1.0000000000010525e+00*i
 2.0000000000000000e+00 + -2.0000000000000000e+00*i    1.9999999999896669e+00 + -2.0000000000028564e+00*i
 3.0000000000000000e+00 +  3.0000000000000000e+00*i    2.9999999999994627e+00 +  2.9999999999987477e+00*i
 4.0000000000000000e+00 + -4.0000000000000000e+00*i    4.0000000000003704e+00 + -3.9999999999991331e+00*i
 5.0000000000000000e+00 +  5.0000000000000000e+00*i    5.0000000000129159e+00 +  5.0000000000035705e+00*i
 6.0000000000000000e+00 + -6.0000000000000000e+00*i    5.9999999999922489e+00 + -6.0000000000021423e+00*i
 7.0000000000000000e+00 +  7.0000000000000000e+00*i    7.0000000000057723e+00 +  6.9999999999975264e+00*i
```

Относительная погрешность решений 1.21e-12 вполне согласуется со значением числа обусловленности 1.38e+05 матрицы коэффициентов.

A.7. Вещественная симметричная матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 4 & 0 & -2 & 0 & 0 & 2 \\ 0 & 3.26 & 0 & -3 & -2 & 0 \\ -2 & 0 & 1.001 & 0 & 0 & 1 \\ 0 & -3 & 0 & 4 & 0 & 2 \\ 0 & -2 & 0 & 0 & 4 & -2 \\ 2 & 0 & 1 & 2 & -2 & 8 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ -2.26 \\ 1.999 \\ -3 \\ 8 \\ -9 \end{bmatrix}$$

Исходный код примера см в файле `example_LDLH_s.c`. Ниже приведена выдача этого примера:

Относительная погрешность решений 5.63e-05 вполне согласуется со значением числа обусловленности 1.88e+03 матрицы коэффициентов.

A.8. Вещественная симметричная матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 4 & 0 & -2 & 0 & 0 & 2 \\ 0 & 3.26 & 0 & -3 & -2 & 0 \\ -2 & 0 & 1.00001 & 0 & 0 & 1 \\ 0 & -3 & 0 & 4 & 0 & 2 \\ 0 & -2 & 0 & 0 & 4 & -2 \\ 2 & 0 & 1 & 2 & -2 & 8 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ -2.26 \\ -1.99999 \\ -3 \\ 8 \\ -9 \end{bmatrix}$$

Исходный код примера см в файле `example_LDLH_d.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution            Numerical solution
 1.0000000000000000e+00    1.0000000000291038e+00
-1.0000000000000000e+00   -9.9999999999998579e-01
 1.0000000000000000e+00    1.0000000000582077e+00
-1.0000000000000000e+00   -9.9999999999998923e-01
 1.0000000000000000e+00    1.0000000000000071e+00
-1.0000000000000000e+00   -1.0000000000000002e+00
```

Число обусловленности матрицы коэффициентов равно 1.90e+03. Относительное уклонение 2.66e-11 приближенного решения с помощью одной итерации уточнения может быть улучшено до 3.39e-15.

A.9. Попытка использовать разложение Холецкого для не положительно-определенной матрицы

В этом примере коэффициенты системы уравнений составляют матрицу bfwb62, все компоненты вектора точного решения равны единице:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{62} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Вектор правой части получен умножением матрицы коэффициентов на точное решение. Попытка решить эту систему уравнений, используя разложение Холецкого (`USP_fact=LLH`), вызывает сообщение об ошибке 4 на стадии факторизации. Напротив, значение `USP_fact=LDLH` приводит к успеху:

```
---- Trying to solve system with LLH
USPARS factorization failed.
Nonzero return code ier=4 was obtained from uspars_fact()
---- Solving system with LDLH
Results:
            name | n    | nnz(A) |   error C    |   error L2   |
./mtx/bfwb62.mtx |  62  |     202| 6.770540e-16 | 3.614540e-16 |
```

Исходный код примера см в файле `example_pos_def.c`

A.10. Комплексная симметричная матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 1 & 0 & 5i & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 8 & 0 & 0 \\ 5i & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & -3i \\ 0 & 8 & 0 & 0 & 2 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 1 & 1+i \\ 0 & 0 & 0 & -3i & 0 & 1+i & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1+15i \\ 44 \\ 9+5i \\ 0.004-21i \\ 26.6 \\ 13.5+7i \\ -1-6i \end{bmatrix}$$

Исходный код примера см в файле `example_LDLT_c.c`. Для лучшей точности дополнительно используется одна итерация уточнения. Ниже приведена выдача этого примера:

```
The system was successfully solved.
      Exact solution                          Numerical solution
 1.0000000e+00 +  0.0000000e+00*i  1.0000000e+00 +  1.7029897e-07*i
 2.0000000e+00 +  0.0000000e+00*i  2.0000002e+00 +  7.6784645e-10*i
 3.0000000e+00 +  0.0000000e+00*i  3.0000000e+00 + -1.7029897e-08*i
 4.0000000e+00 +  0.0000000e+00*i  4.0000005e+00 + -2.1909364e-07*i
 5.0000000e+00 +  0.0000000e+00*i  5.0000000e+00 + -1.9196161e-10*i
 6.0000000e+00 +  0.0000000e+00*i  6.0000005e+00 + -5.7588494e-08*i
 7.0000000e+00 +  0.0000000e+00*i  7.0000000e+00 +  1.5073585e-11*i
```

Число обусловленности матрицы коэффициентов равно 12.58. Относительное уклонение 6.65e-8 даже превосходит ожидания.

A.11. Комплексная симметричная матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 1 & 0 & 5i & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 8 & 0 & 0 \\ 5i & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & -3i \\ 0 & 8 & 0 & 0 & 2 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 1 & 1+i \\ 0 & 0 & 0 & -3i & 0 & 1+i & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1+15i \\ 44 \\ 9+5i \\ 0.004-21i \\ 26.6 \\ 13.5+7i \\ -1-6i \end{bmatrix}$$

Исходный код примера см в файле `example_LDLT_z.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution                                  Numerical solution
 1.0000000000000000e+00 +  0.0000000000000000e+00*i   1.0000000000000000e+00 +  1.9032394707859828e-16*i
 2.0000000000000000e+00 +  0.0000000000000000e+00*i   2.0000000000000000e+00 +  2.8931319859074138e-18*i
 3.0000000000000000e+00 +  0.0000000000000000e+00*i   3.0000000000000004e+00 +  2.2204460492503131e-16*i
 4.0000000000000000e+00 +  0.0000000000000000e+00*i   4.0000000000000000e+00 +  0.0000000000000000e+00*i
 5.0000000000000000e+00 +  0.0000000000000000e+00*i   5.0000000000000000e+00 + -7.2328299647685346e-19*i
 6.0000000000000000e+00 +  0.0000000000000000e+00*i   6.0000000000000000e+00 + -2.1698489894305601e-16*i
 7.0000000000000000e+00 +  0.0000000000000000e+00*i   7.0000000000000000e+00 +  0.0000000000000000e+00*i
```

Число обусловленности матрицы коэффициентов равно 12.58. Относительное уклонение 4.85e-17 приближенного решения от точного превосходит ожидания.

A.12. Комплексная эрмитова матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 1 & 0 & 5i & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 8 & 0 & 0 \\ -5i & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & -3i \\ 0 & 8 & 0 & 0 & 2 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 1 & 1+i \\ 0 & 0 & 0 & 3i & 0 & 1-i & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1+15i \\ 44 \\ 9+5i \\ 0.004-21i \\ 26.6 \\ 13.5+7i \\ -1-6i \end{bmatrix}$$

Исходный код примера см в файле `example_LDLH_c.c`. Для лучшей точности дополнительно используется одна итерация уточнения. Ниже приведена выдача этого примера:

```
The system was successfully solved.
      Exact solution                Numerical solution
 1.0000000e+00 +  0.0000000e+00*i 1.0000000e+00 + -2.1674416e-07*i
 2.0000000e+00 +  0.0000000e+00*i 2.0000002e+00 + -2.3225937e-09*i
 3.0000000e+00 +  0.0000000e+00*i 3.0000000e+00 +  5.4186060e-09*i
 4.0000000e+00 +  0.0000000e+00*i 4.0000000e+00 +  1.7601997e-07*i
 5.0000000e+00 +  0.0000000e+00*i 5.0000000e+00 +  5.8064842e-10*i
 6.0000000e+00 +  0.0000000e+00*i 6.0000000e+00 +  1.7419441e-07*i
 7.0000000e+00 +  0.0000000e+00*i 7.0000000e+00 + -7.0595190e-13*i
```

Число обусловленности матрицы коэффициентов равно 12.63. Относительное уклонение 3.26e-08 приближенного решения от точного находится в обоснованных пределах.

A.13. Комплексная эрмитова матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 1 & 0 & 5i & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 8 & 0 & 0 \\ -5i & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & -3i \\ 0 & 8 & 0 & 0 & 2 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 1 & 1+i \\ 0 & 0 & 0 & 3i & 0 & 1-i & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1+15i \\ 44 \\ 9+5i \\ 0.004-21i \\ 26.6 \\ 13.5+7i \\ -1-6i \end{bmatrix}$$

Исходный код примера см в файле `example_LDLH_z.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution                                        Numerical solution
 1.0000000000000000e+00+0.0000000000000000e+00*i 1.0000000000000000e+00 -2.4223047810003419e-16*i
 2.0000000000000000e+00+0.0000000000000000e+00*i 2.0000000000000000e+00 +1.1835803775002936e-17*i
 3.0000000000000000e+00+0.0000000000000000e+00*i 3.0000000000000004e+00 -2.2204460492503131e-16*i
 4.0000000000000000e+00+0.0000000000000000e+00*i 4.0000000000000000e+00 +0.0000000000000000e+00*i
 5.0000000000000000e+00+0.0000000000000000e+00*i 5.0000000000000000e+00 -2.9589509437507341e-18*i
 6.0000000000000000e+00+0.0000000000000000e+00*i 6.0000000000000000e+00 -8.8768528312522019e-16*i
 7.0000000000000000e+00+0.0000000000000000e+00*i 7.0000000000000000e+00 -1.0842021724855044e-19*i
```

Число обусловленности матрицы коэффициентов равно 12.63. Относительное уклонение 3.75e-17 приближенного решения от точного находится в обоснованных пределах.

A.14. Вещественная матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 1 & 0 & 5 & 0 & 0 & 0 & 10 \\ 0 & 2 & 0 & 8 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -3 \\ 0 & 1.01 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 86 \\ 40 \\ 10 \\ -17 \\ 16 \\ 12 \\ 3 \end{bmatrix}.$$

Исходный код примера см в файле `example_LU_s.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution    Numerical solution
 1.0000000e+00     1.0000591e+00
 0.0000000e+00     1.7764522e-05
 3.0000000e+00     2.9999802e+00
 4.0000000e+00     4.0000134e+00
 5.0000000e+00     4.9999957e+00
 6.0000000e+00     5.9999909e+00
 7.0000000e+00     7.0000043e+00
```

Число обусловленности матрицы коэффициентов равно 73.89. Относительное уклонение 5.75e-06 приближенного решения от точного находится в обоснованных пределах.

A.15. Вещественная матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 0 & 0 & 105 & 0 & 0 & 0 & 107 \\ 0 & 1 & 0 & 0 & 18 & 0 & 0 \\ 1 & 0 & 1 & 0 & 111 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & -103 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 212 \\ 17 \\ 113 \\ -105 \\ 0 \\ 1 \\ -3 \end{bmatrix}.$$

Исходный код примера см в файле `example_LU_d.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution            Numerical solution
 1.0000000000000000e+00    1.0000000000000637e+00
-1.0000000000000000e+00   -9.9999999999999045e-01
 1.0000000000000000e+00    1.0000000000000000e+00
-1.0000000000000000e+00   -1.0000000000000004e+00
 1.0000000000000000e+00    9.9999999999999944e-01
-1.0000000000000000e+00   -9.9999999999999956e-01
 1.0000000000000000e+00    1.0000000000000000e+00
```

Число обусловленности матрицы коэффициентов равно 3.06e+04. Относительное уклонение 2.44e-14 приближенного решения от точного находится в обоснованных пределах.

A.16. Комплексная матрица коэффициентов с одинарной точностью

Система уравнений

$$\begin{pmatrix} 0 & 0 & 105 & 0 & 0 & 0 & 300-4i \\ 0 & 1+i & 0 & 0 & 1+8i & 0 & 0 \\ 1 & 0 & 1 & 0 & 111-20i & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & -103 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & i & 0 & 1 & -1+10i \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 300+101i \\ 7i \\ 113-20i \\ -105 \\ 0 \\ 1 \\ -2+9i \end{bmatrix}.$$

Исходный код примера см в файле `example_LU_c.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution                      Numerical solution
 1.0000000e+00 +  0.0000000e+00*i    9.9996775e-01 +  5.3781223e-05*i
-1.0000000e+00 +  0.0000000e+00*i   -1.0000030e+00 +  7.5085632e-07*i
 1.0000000e+00 +  0.0000000e+00*i    1.0000001e+00 +  0.0000000e+00*i
-1.0000000e+00 +  0.0000000e+00*i   -1.0000001e+00 + -4.4926441e-07*i
 1.0000000e+00 +  0.0000000e+00*i    1.0000004e+00 + -4.1516284e-07*i
-1.0000000e+00 +  0.0000000e+00*i   -1.0000005e+00 +  4.1516284e-07*i
 1.0000000e+00 +  0.0000000e+00*i    1.0000000e+00 +  0.0000000e+00*i
```

Число обусловленности матрицы коэффициентов равно 9.45e+05. Относительное уклонение 2.37e-05 приближенного решения от точного находится в обоснованных пределах.

A.17. Комплексная матрица коэффициентов с двойной точностью

Система уравнений

$$\begin{pmatrix} 104 & 0 & 105i & 0 & 0 & 0 & 30-4i \\ 0 & 1+i & 0 & 0 & 1+8i & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 97 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0.1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0.96i & 0 & 111-20i & 0 & 0 \\ 0 & 0 & 0 & i & 0 & 1 & -3i \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 134+101i \\ 2+7i \\ 97-2i \\ 1.1-i \\ 1-i \\ 112-19.04i \\ 1-4i \end{bmatrix}.$$

Исходный код примера см в файле `example_LU_z.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution                                  Numerical solution
 1.0000000000000000e+00 +  0.0000000000000000e+00*i    9.9999999999737788e-01 + -1.3273233099706723e-12*i
-0.0000000000000000e+00 + -1.0000000000000000e+00*i   -7.2840132591628681e-16 + -1.0000000000000058e+00*i
 1.0000000000000000e+00 +  0.0000000000000000e+00*i    1.0000000000013194e+00 + -2.6025693228422387e-12*i
-0.0000000000000000e+00 + -1.0000000000000000e+00*i    7.2980095736518887e-16 + -1.0000000000000009e+00*i
 1.0000000000000000e+00 +  0.0000000000000000e+00*i    1.0000000000000009e+00 +  7.2984424493577328e-16*i
-0.0000000000000000e+00 + -1.0000000000000000e+00*i   -8.8817841970012523e-16 + -1.0000000000000007e+00*i
 1.0000000000000000e+00 +  0.0000000000000000e+00*i    1.0000000000000000e+00 +  1.8503717077085926e-17*i
```

Число обусловленности матрицы коэффициентов равно 1.098e+08. Относительное уклонение 1.57e-12 приближенного решения от точного находится в обоснованных пределах.

A.18. Вещественная симметричная положительно-определенная матрица коэффициентов с одинарной точностью, две правые части

Система уравнений

$$\begin{pmatrix} 4 & 0 & -2 & 0 & 0 & 2 \\ 0 & 5 & 0 & -3 & -2 & 0 \\ -2 & 0 & 4 & 0 & 0 & 3 \\ 0 & -3 & 0 & 5 & 0 & 2 \\ 0 & -2 & 0 & 0 & 3.96 & -2 \\ 2 & 0 & 3 & 2 & -2 & 8.16 \end{pmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \\ x_{51} & x_{52} \\ x_{61} & x_{62} \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & -4 \\ 5 & -1 \\ 4 & -4 \\ -0.04 & 7.96 \\ 13.16 & -7.16 \end{bmatrix}.$$

Исходный код примера см в файле `example2rhs_LLH_s.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
```

```
 Exact solution    Numerical solution

 rhs 0:
 1.0000000e+00     9.9828684e-01
 1.0000000e+00     9.9987251e-01
 1.0000000e+00     9.9804211e-01
 1.0000000e+00     9.9933606e-01
 1.0000000e+00     1.0006772e+00
 1.0000000e+00     1.0014684e+00

 rhs 1:
 1.0000000e+00     1.0017132e+00
-1.0000000e+00    -9.9987239e-01
 1.0000000e+00     1.0019579e+00
-1.0000000e+00    -9.9933606e-01
 1.0000000e+00     9.9932289e-01
-1.0000000e+00    -1.0014684e+00
```

Число обусловленности матрицы коэффициентов равно 5.0787e+05. Относительное уклонение 6.92e-04 приближенного решения от точного вполне оправдано.

A.19. Вещественная матрица коэффициентов с одинарной точностью, две правые части

Система уравнений

$$\begin{pmatrix} 1 & 0 & 5 & 0 & 0 & 0 & 10 \\ 0 & 2 & 0 & 8 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -3 \\ 0 & 1.01 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{pmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \\ x_{51} & x_{52} \\ x_{61} & x_{62} \\ x_{71} & x_{72} \end{bmatrix} = \begin{bmatrix} 86 & 16 \\ 40 & 10 \\ 10 & 4 \\ -17 & -2 \\ 16 & 4.01 \\ 12 & 2 \\ 3 & 1 \end{bmatrix}.$$

Исходный код примера см в файле `example2rhs_LU_s.c`. Ниже приведена выдача этого примера:

```
 The system was successfully solved.
 Exact solution    Numerical solution

 rhs 0:
 1.0000000e+00     1.0000591e+00
 0.0000000e+00     1.7764522e-05
 3.0000000e+00     2.9999802e+00
 4.0000000e+00     4.0000134e+00
 5.0000000e+00     4.9999957e+00
 6.0000000e+00     5.9999909e+00
 7.0000000e+00     7.0000043e+00

 rhs 1:
 1.0000000e+00     1.0000657e+00
 1.0000000e+00     1.0000175e+00
 1.0000000e+00     9.9997813e-01
 1.0000000e+00     1.0000131e+00
 1.0000000e+00     9.9999565e-01
 1.0000000e+00     9.9999130e-01
 1.0000000e+00     1.0000044e+00T
```

Число обусловленности матрицы коэффициентов равно 73.89. Относительное уклонение 2.81e-05 приближенного решения от точного вполне оправдано.

A.20. Использование итерационного уточнения для улучшения точности

Система уравнений образована на матрице rw496 с вектором точного решения, состоящего из единиц

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{496} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

и соответствующей ему правой частью. Исходный код примера см в файле `example_iter_ref.c` Матрица не очень хорошо обусловлена (cond(A)=1.99e+10), что сказывается на точности решения. Включение итерационного уточнения способно улучшить точность результата, как показывают результаты расчетов в этом примере:

```
  Results w/o iter ref:
            name |   n    |  nnz(A)  |     error C     |     error L2     |
 ./mtx/rw496.mtx |   496  |     1859 |   6.065465e+02  |   3.089256e+01  |

  Results with iter ref:
            name |   n    |  nnz(A)  |     error C     |     error L2     |
 ./mtx/rw496.mtx |   496  |     1859 |   6.858562e-07  |   3.441423e-08  |
```

В этих таблицах в колонках `error C` и `error L2` располагаются относительные погрешности $\frac{\|x-x_{\text{выч}}\|_{\max}}{\|x\|_{\max}}$ и $\frac{\|x-x_{\text{выч}}\|_2}{\|x\|_2}$, соответственно, в нормах

$$\|x\|_{\max} = \max_{1\le i\le 496} |x_i|, \quad \|x\|_2 = \sqrt{\sum_{i=1}^{496} |x_i|^2}\,.$$

A.21. Диагональная поддержка чтобы ‘обойти’ проблему Zero Pivot

Пример демонстрирует эффект включения диагональной поддержки (см C.3.1) при возникновении кода ошибки 4 на стадии факторизации. Система уравнений образована (см также A.22) на матрице impcol_c[8] с вектором точного решения, состоящего из единиц

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{137} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

и соответствующей ему правой частью. Исходный код примера см в файле `example_boosting.c`. Ниже приведена выдача этого примера:

```
---- Trying to solve system with boosting turned off.
USPARS factorization failed.
 Nonzero return code 4 was obtained from uspars_fact()

---- Solving system with boosting turned on.
  Results:
             name |   n    |  nnz(A)  |     error C     |     error L2     |
./mtx/impcol_c.mtx |   137  |      411 |   6.661338e-16  |   9.485275e-17  |
```

A.22. Глобальные перестановки чтобы ‘обойти’ проблему Zero Pivot

Пример демонстрирует использование глобальных перестановок (см C.3.2) для преодоления возникновения ошибки 4 на стадии факторизации. Система уравнений (см также A.21) образована на матрице impcol_c с вектором точного решения, состоящего из единиц

[8] Отметим, что матрица impcol_c обусловдена совсем неплохо. Ее числу рбусловденности равно 1.77e+04

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{137} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

и соответствующей ему правой частью. Исходный код примера см в файле `example_gp.c`. Ниже приведена выдача этого примера:

```
---- Trying to solve system with global pivot turned off.
USPARS factorization failed.
Nonzero return code ier=4 was obtained from uspars_fact()

---- Global pivot turned on.
Results:
              name | n   | nnz(A)  |   error C    |  error L2    |
./mtx/impcol_c.mtx | 137 |     411 | 4.440892e-16 | 7.705874e-17 |
```

A.23. Использование дисковой памяти при решении системы

Пример демонстрирует использование OOC (см C.4) в двух режимах. В первом случае из-за небольшого размера матрицы и выбора соответствующего значения параметра, сброса промежуточных значений на диск не происходит, и система решается в IC режиме. Во втором же промежуточные значения принудительно сбрасываются на диск при факторизации, и считываются в дальнейшем на стадии решения. Система уравнений (см также A.21) образована на матрице impcol_c с вектором точного решения, состоящего из единиц

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{137} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

и соответствующей ему правой частью. Исходный код примера см в файле `example_OOC.c`. Ниже приведена выдача этого примера:

```
---- USP_OOC_MEM = 10000
Results w/o force OOC:
              name |   n   |  nnz(A)  |     error C    |    error L2    |
./mtx/impcol_c.mtx |   137 |      411 |   4.440892e-16 |   6.773842e-17 |

---- USP_OOC_MEM = -10000
Results w force OOC:
              name |   n   |  nnz(A)  |     error C    |    error L2    |
./mtx/impcol_c.mtx |   137 |      411 |   4.440892e-16 |   6.773842e-17 |
```

A.24. Использование long long int интерфейсов

Пример демонстрирует использование long long int (см 1.1.1, 2.2) интерфейсов. В нем представлены три запуска:

- Массивы `ia` и `ja` имеют тип `int`, вызывается метод `uspars_init32`
- Массив `ia` имеет тип `int`, массив `ja` – `long long int`, вызывается функция `uspars_init3264`
- Массивы `ia` и `ja` имеют тип `long long int`, вызывается функция `uspars_init64`

Система уравнений во всех случаях решается следующая:

$$\begin{pmatrix} 0 & 0 & 105 & 0 & 0 & 0 & 107 \\ 0 & 1 & 0 & 0 & 18 & 0 & 0 \\ 1 & 0 & 1 & 0 & 111 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & -103 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 212 \\ 17 \\ 113 \\ -105 \\ 0 \\ 1 \\ -3 \end{bmatrix}.$$

Исходный код примера см в файле `example_3264.c`. Ниже приведена выдача этого примера:

```
The system was successfully solved.
 Exact solution            Numerical solution (32)
 1.0000000000000000e+00    1.0000000000000637e+00
-1.0000000000000000e+00   -9.9999999999999045e-01
 1.0000000000000000e+00    1.0000000000000000e+00
-1.0000000000000000e+00   -1.0000000000000004e+00
 1.0000000000000000e+00    9.9999999999999944e-01
-1.0000000000000000e+00   -9.9999999999999956e-01
 1.0000000000000000e+00    1.0000000000000000e+00
____________

The system was successfully solved.
 Exact solution            Numerical solution (3264)
 1.0000000000000000e+00    1.0000000000000637e+00
-1.0000000000000000e+00   -9.9999999999999045e-01
 1.0000000000000000e+00    1.0000000000000000e+00
-1.0000000000000000e+00   -1.0000000000000004e+00
 1.0000000000000000e+00    9.9999999999999944e-01
-1.0000000000000000e+00   -9.9999999999999956e-01
 1.0000000000000000e+00    1.0000000000000000e+00
____________

The system was successfully solved.
 Exact solution            Numerical solution (64)
 1.0000000000000000e+00    1.0000000000000637e+00
-1.0000000000000000e+00   -9.9999999999999045e-01
 1.0000000000000000e+00    1.0000000000000000e+00
-1.0000000000000000e+00   -1.0000000000000004e+00
 1.0000000000000000e+00    9.9999999999999944e-01
-1.0000000000000000e+00   -9.9999999999999956e-01
 1.0000000000000000e+00    1.0000000000000000e+00
____________
```

B. *Линейно-алгебраические и алгоритмические основы*

B.1. Метод исключения Гаусса

Система уравнений записывается в виде

$$Ax = f, \qquad (B.1)$$

где векторы x и f имеют по N компонент, матрица A имеет размеры $N \times N$. С алгоритмической точки зрения целесообразно выделить несколько классов, к которым может принадлежать матрица коэффициентов.

B.1.1. Вещественные симметричные или комплексные эрмитовы положительно определенные матрицы

В вещественном случае класс матриц выделяется условием

$$A = A^T > 0, \qquad (B.2)$$

в комплексном условие выглядит так:

$$A = A^H > 0. \qquad (B.3)$$

Здесь A^T обозначает транспонирование матрицы A, $A^H = \overline{A}^T$ – эрмитово сопряжение. Для такого класса матриц можно использовать *разложение Холецкого*

$$A = \begin{cases} LL^T & \text{вещественный симметричный случай} \\ LL^H & \text{комплексный эрмитов случай} \end{cases} \qquad (B.4)$$

Заметим, что здесь L – нижнетреугольная, а L^T и L^H автоматически получаются верхнетреугольными[9]. В случае разреженной матрицы A нижнетреугольная матрица L также оказывается разреженной, но обычно со значительно большим числом ненулевых элементов (*эффект заполнения*).

Число ненулевых элементов в L часто значительно больше, чем число ненулевых в исходной матрице. Это число зависит от порядка строк (и столбцов), выбранного для записи исходной матрицы. Чтобы уменьшить число ненулевых элементов в L, снизив тем самым нагрузку на память и уменьшив требующееся число операций с плавающей точкой, применяют специальные алгоритмы перестановки, которые при сравнительно небольших дополнительных затратах позволяют достигать существенной экономии. Это означает, что вместо исходной матрицы факторизации подвергается произведение PAP^T, где P – матрица перестановки (см 1.4).

Таким образом, алгоритм решения системы линейных уравнений (B.1) с матрицей коэффициентов, удовлетворяющей условиям (B.2) или (B.3), состоит из следующих шагов:

1. Симметричная перестановка строк и столбцов матрицы

$$A_1 = PAP^T \qquad (B.5)$$

[9] Соответствующая функциональность в пакете LAPACK – подпрограмма ?POTRF, где символ ? обозначает одну из букв {S, D, C, Z}, определяющих соответствующий тип данных (S = ‘single precision’, D=’double precision’, C=’complex’, Z=’double complex’)

и компонент вектора правой части

$$g = Pf; \qquad \text{(В. 6)}$$

2. Треугольная факторизация матрицы

$$A_1 = \begin{cases} LL^T & \text{вещественный симметричный случай} \\ LL^H & \text{комплексный эрмитов случай} \end{cases} \qquad \text{(В. 7)}$$

3. Решение систем уравнений с треугольными матрицами коэффициентов

$$Ly = g; \quad \begin{matrix} L^T z = y & \text{вещественный симметричный случай} \\ L^H z = y & \text{комплексный эрмитов случай} \end{matrix} \qquad \text{(В. 8)}$$

4. Перестановка компонент вектора неизвестных

$$x = P^T z. \qquad \text{(В. 9)}$$

Считается, что условия положительной определенности (В.2), (В.3) достаточно для численной устойчивости алгоритма.

В.1.2. Симметричные (вещественные или комплексные) и комплексные эрмитовы матрицы коэффициентов}

Класс матриц выделяется условиями

$$A = \begin{cases} A^T & \text{симметричный случай} \\ A^H & \text{комплексный эрмитов случай} \end{cases} \qquad \text{(В. 10)}$$

Этот класс полностью содержит в себе положительно определенные матрицы, выделенные в предыдущем подразделе. Тем самым, все, что описано ниже в этом подразделе, применимо и к предыдущему.

Соответствующее разложение называют *LDLT-разложением* (*LDLH-разложением* в эрмитовом случае)

$$PAP^T = \begin{cases} LDL^T & \text{симметричный случай} \\ LDL^H & \text{комплексный эрмитов случай} \end{cases} \qquad \text{(В. 11)}$$

В этих формулах P обозначает матрицу перестановок, используемую для повышения устойчивости алгоритма, $L-$ нижнетреугольная с единицами на диагонали, D обозначает блочно-диагональную матрицу с (симметричными или эрмитовыми) блоками небольшого размера на диагонали[10].

Все сказанное в предыдущем подразделе относительно структуры треугольного множителя применимо и к случаю LDLT-разложения. Соответственно, алгоритм решения системы линейных уравнений (В.1) с матрицей коэффициентов, удовлетворяющей условиям (В.10), разбивается на шаги:

1. Симметричная перестановка строк и столбцов (В.5) матрицы и компонент вектора правой части (В.6).
2. Треугольная факторизация матрицы (ср. (В.11))

$$A_1 = \begin{cases} LDL^T & \text{симметричный случай} \\ LDL^H & \text{комплексный эрмитов случай} \end{cases} \qquad \text{(В. 12)}$$

3. Решение систем уравнений с матрицами коэффициентов, полученных в результате факторизации

$$\begin{matrix} Ly = g; \\ Dw = y; \\ L^T z = w. \end{matrix} \qquad \text{(В. 13)}$$

В эрмитовом случае последняя система уравнений примет вид

[10] Соответствующие функции в LAPACK имеют названия `?SYTRF, CHETRF, ZHETRF`, где знак `?` обозначает одну из букв `S, D, C, Z`.

$$\begin{aligned} Ly &= g; \\ Dw &= y; \\ L^H z &= w. \end{aligned} \tag{B.14}$$

4. Перестановка компонент вектора неизвестных (B.9)

B.1.3. Матрицы коэффициентов общего вида

В отличие от предыдущих подразделов, никаких ограничений на матрицу A коэффициентов системы линейных уравнений (B.1) не накладывается.

Треугольное разложение матрицы коэффициентов имеет вид

$$A = LU \tag{B.15}$$

где $L -$ нижнетреугольная матрица с единицами на диагонали, $U -$ верхнетреугольная матрица.

Как и в предыдущем подразделе, к разреженной матрице коэффициентов могут применяться перестановки строк и столбцов, предназначенные для уменьшения числа ненулевых элементов в треугольных факторах и повышения устойчивости вычислений к ошибкам округлений. Только перестановки слева и справа, вообще говоря, различны. Алгоритм решения системы линейных уравнений оказывается идейно похожим на описанные в предыдущих подразделах.

1. Перестановки строк и столбцов матрицы коэффициентов и компонент вектора правой части
$$\begin{aligned} A_1 &= P_1 A P_2; \\ g &= P_1 f. \end{aligned} \tag{B.16}$$
2. Треугольная факторизация матрицы (ср. (B.15))
$$A_1 = LU \tag{B.17}$$
3. Решение систем уравнений с треугольными матрицами коэффициентов
$$\begin{aligned} Ly &= g; \\ Uz &= y. \end{aligned} \tag{B.18}$$
4. Перестановка компонент вектора неизвестных
$$x = P_2 z. \tag{B.19}$$

B.1.4. Структура треугольных факторов

Треугольные факторы, на которые раскладывается исходная матрица, также обладают разреженной структурой, но с одной существенной оговоркой. Число ненулевых элементов в треугольных факторах обычно превышает число ненулевых элементов в исходной матрице. В этом проявляется так называемый *эффект заполнения* (*fill-in*). Для иллюстрации рассмотрим симметричную положительно определенную матрицу

$$A = \begin{pmatrix} 7 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Вычисляя ее разложение Холецкого $A = LL^T$, получаем

$$L = \begin{pmatrix} 2.6458 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3780 & 0.9258 & 0 & 0 & 0 & 0 & 0 \\ 0.3780 & -0.1543 & 0.9129 & 0 & 0 & 0 & 0 \\ 0.3780 & -0.1543 & -0.1826 & 0.8944 & 0 & 0 & 0 \\ 0.3780 & -0.1543 & -0.1826 & -0.2236 & 0.8660 & 0 & 0 \\ 0.3780 & -0.1543 & -0.1826 & -0.2236 & -0.2887 & 0.8165 & 0 \\ 0.3780 & -0.1543 & -0.1826 & -0.2236 & -0.2887 & -04082 & 0.7071 \end{pmatrix}$$

Разложение Холецкого той же матрицы, но с симметрично переставленными строками и столбцами, имеет вид

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Сравнение двух разложений иллюстрирует утверждение, что число ненулевых элементов в треугольных факторах метода Гаусса зависит от выбранного порядка строк и столбцов матрицы.

Эффект заполнения влияет на объем требуемой оперативной памяти. Нетрудно понять, что от заполнения также зависит число арифметических операций, требующихся для осуществления треугольной факторизации. Задача о нахождении оптимального (в смысле минимизации эффекта заполнения) упорядочивания строк и столбцов матрицы NP-сложная, и потому не имеет практического значения. Тем не менее, существуют алгоритмы ([4], [5], [6]), в результате применения которых эффект заполнения оказывается существенно сниженным.

В пакете USPARS для отыскания перестановок матриц с целью снижения заполненности треугольных факторов разработана компонента Atlant (см B.4). Имеется возможность использовать пакет Metis ([7]), если он проинсталлирован на вычислительной системе, а также использовать перестановку, определяемую пользователем или вообще не подвергать исходную матрицу никаким перестановкам (см. раздел 2.2).

B.2. Балансировка коэффициентов уравнений

В алгоритмах решения систем линейных уравнений для улучшения устойчивости вычислений к погрешностям округлений можно применять так называемое *масштабирование строк и столбцов матрицы*–взамен системы уравнений

$$Ax = f$$

рассматривается система

$$A_1 y = D_l f$$

в которой матрица коэффициентов получается из исходной матрицы умножением

$$A_1 = D_l A D_r$$

слева и справа на специально подобранные диагональные масштабирующие матрицы D_l и D_r. Элементы диагоналей масштабирующих матриц выбираются так, чтобы в строках и столбцах матрицы A_1 максимальные элементы были порядка единицы. Такой прием приводит к уменьшению числа обусловленности матрицы коэффициентов, и, соответственно, уменьшает погрешность численного решения системы линейных уравнений.

B.3. Matching

С целью увеличения надежности численной факторизации матрицы можно так переставить уравнения в системе

$$Ax = f$$

чтобы «усилить» диагональ матрицы коэффициентов. Для этого применяется перестановка P_m

строк исходной матрицы A, в результате которой на главную диагональ попадают большие по модули элементы матрицы; для элементов правой части f применяется та же самая перестановка:

$$A_1 = P_m A, \;\; f_1 = P_m f.$$

В пакете USPARS реализованы несколько алгоритмов:

1) Алгоритм максимизации минимального диагонального элемента (см [8])
2) Алгоритм максимизации произведения диагональных элементов с возможностью балансировки матрицы (см [8])

B.4. Atlant

В состав пакета USPARS входит компонента Atlant для переупорядочивания разреженных матриц с целью снижения заполняемости формируемых в процессе разложения матрицы факторов (fill-in reducing reordering). В компоненте в качестве основного метода переупорядочивания используется комбинированный алгоритм на основе многоуровневого метода вложенных сечений [9] и эффективного авторского метода переупорядочивания подграфов малого размера на основе алгоритма минимальной степени [5].

C. Практические рекомендации по использованию пакета USPARS

C.1. Использование многопоточности

Функции пакета USPARS распараллелены под многоядерные системы средствами OpenMP. Для того, чтобы в программе, использующей функции USPARS, исполнение USPARS происходило в многопоточном режиме, следует установить число используемых omp-потоков[11]. Например, если планируется использовать 16 потоков, то перед самым первым вызовом функций USPARS можно поставить оператор `omp_set_num_threads(16)`.

Установить число используемых потоков можно, присвоив переменной окружения `OMP_NUM_THREADS` значение равное требуемому числу потоков. В среде Linux это делается командой

```
export OMP_NUM_THREADS=16
```

Соответственно, в среде Windows команда выглядит так

```
set OMP_NUM_THREADS=16
```

C.2. Итерационное уточнение

Третья по списку

- перестановка,
- факторизация,
- решение систем с треугольными матрицами коэффициентов

фаза алгоритма пакета USPARS, применяемого для решения системы линейных уравнений

$$Ax = f,$$

обычно занимает относительно малую долю вычислительного времени. Использование этого обстоятельства лежит в основе использования итерационного уточнения (см `options[USP_ITER_REF]` в разделе 2.2) для улучшения точности решения системы линейных уравнений. Организовывается дополнительный вычислительный процесс, в котором решение получается за несколько однотипных шагов. На каждом шаге процесса должна быть решена система линейных уравнений с использованием треугольных факторов исходной матрицы, полученных в результате факторизации. В качестве вектора правой части служит невязка от уточненного решения, полученного на предыдущем шаге. В вычислениях невязки участвует исходная матрица, причем, если возможно, вычисления проводят в формате повышенной точности. Использование описанного приема во многих случаях позволяет получить решение с лучшей точностью, чем без его использования.

При некоторых условиях получаемый на k-ом шаге итерационного процесса вектор $x^{(k)}$ обеспечивает решение с лучшей невязкой

$$\left\|Ax^{(k)} - f\right\| < \left\|Ax^{(k-1)} - f\right\|.$$

Выход из процесса итерационного уточнения происходит по достижению предела числа итераций

$$k \le k_{\max} \qquad (C.1)$$

либо достижению критерия малости относительной нормы невязки

[11]Желательно, чтобы число потоков не превышало числа физически доступных ядер процессора. В противном случае возможна деградация производительности.

$$\frac{\|Ax^{(k)} - f\|}{\|f\|} < \alpha. \qquad (C.2)$$

Параметры α и $k_{\max}$ заданы по умолчанию либо устанавливаются пользователем (см раздел 2.2). В случае, если невязка в процессе решения достигает значения inf, процесс итерационного уточнения объявляется несошедшимся и функция `uspars_solve()` (см раздел 2.4) возвращает код 7.

C.3. Приемы 'обхода' ошибки Zero Pivot

В процессе треугольной факторизации матрицы диагональный элемент может оказаться равным нулю. Процесс не может быть продолжен потому, что диагональные элементы используются в качестве знаменателей в процессе факторизации. При этом появление нулевого элемента не обязательно связано с вырожденностью исходной матрицы – она может быть хорошо обусловленной. Для преодоления этого затруднения используется так называемый *выбор ведущего элемента в столбце (pivoting)* − перестановка строк, в результате которой на место диагонального элемента попадает ненулевой элемент из того же столбца, расположенный ниже диагонали. После этого факторизацию можно продолжить до появления следующего нулевого элемента на диагонали. Тогда снова требуется перестановка строк. В результате матрица оказывается факторизованной− представленной в виде произведения двух треугольных матриц и матрицы перестановок.

Если на каком-то шаге не удается найти ненулевой ведущий элемент в столбце ниже нулевого диагонального элемента, то это будет означать, что исходная матрица вырождена со всеми вытекающими последствиями для соответствующей системы линейных уравнений.

В случае выполнения вычислений на компьютере арифметические операции над числами подвержены округлениям, и рассуждения выше должны быть модифицированы. Во-первых, даже если в процессе факторизации на диагонали не появляются нули, деление на малые знаменатели может приводить к большим величинам вплоть до `overflow`. Во-вторых, имеет место накопление погрешностей округлений, которое может искажать результат. Таким образом, выбор ведущего элемента необходим, но он должен быть модифицирован.

Ясно, что в качестве ведущего элемента нужно просто брать максимальный по модулю элемент, причем это делать всегда. Различают две стратегии выбора максимального по модулю элемента:

A. Выбор максимального в столбце (Partial pivoting). При рассмотрении диагонального элемента $\tilde{A}_{ii}$ для отыскания ведущего элемента следует найти
$$\max_{i \le k \le N} |\tilde{A}_{ki}|.$$
Найденный максимальный элемент должен быть переставлен на место $\tilde{A}_{ii}$. Для этого нужно переставить две строки.

B. Выбор максимального на диагонали (Diagonal pivoting). Данный способ позволяет выбрать ведущий элемент, не нарушив симметрию матрицы, если таковая присутствовала. Решаемая задача
$$\max_{i \le k \le N} |\tilde{A}_{kk}|$$
Здесь максимальный элемент ищется на диагонали. Найденный элемент переставляется на место $\tilde{A}_{ii}$ с помощью перестановки строк и столбцов.

C. Выбор по все матрице (Full pivoting). Здесь решаемая задача поиска ведущего элемента имеет вид
$$\max_{\substack{i \le k \le N \\ i \le j \le N}} |\tilde{A}_{kj}|$$
Здесь для того, чтобы поставить найденный максимальный элемента на место $\tilde{A}_{ii}$, нужно будет переставить строки и столбцы.

В этих формулах тильда в обозначениях использована для обозначения текущих значений элементов массива, содержащего элементы матрицы, чтобы отличить их от элементов исходного массива (все вычисления производятся 'на месте').

Теоремы об устойчивости решения доказываются при условии, что выбор ведущего элемента производится по всей матрице. Теорем, гарантирующих устойчивость при выборе по столбцу, нет. Более того, известны контрпримеры, показывающие, что при использовании выбора ведущего по столбцу может наблюдаться вычислительная неустойчивость. Однако, по вполне очевидным причинам практическое применение находит использование стратегии Partial pivoting. При этом авторы ссылаются на то, что использование этой стратегии обычно практически всегда приводит к устойчивым результатам.

Все предыдущие рассуждения не принимают во внимание структуру ненулевых элементов в матрице и потому относятся к плотным матрицам. Перед факторизацией разреженной матрицы применяют специальные перестановки строк и столбцов, призванные минимизировать потребляемую память. При этом в матрице возникает некоторая структура ненулевых элементов, которая не должна быть нарушена. В итоге перестановки строк и столбцов разрешены, но только в определенных пределах. После перестановок строк и столбцов, имеющих целью уменьшение эффекта заполнения, в матрице выделяются квадратные блоки на диагонали. Перестановки, осуществляемые с целью выбора ведущего элемента, не должны нарушать эту блочную структуру. Это можно проиллюстрировать следующей формулой

$$\begin{pmatrix} A_{11} & \times & \cdots & \times & \times & \times & \cdots \\ \times & A_{22} & \cdots & \times & \times & \times & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \times & \times & \cdots & A_{i-1,i-1} & \times & \times & \cdots \\ \times & \times & \cdots & \times & \boxed{A_{i,i}} & \times & \cdots \\ \times & \times & \cdots & \times & \times & A_{i+1,i+1} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Квадратные диагональные блоки $A_{i,i}$ могут иметь различные порядки и трактуются как плотные. Крестами обозначены, вообще говоря, прямоугольные блоки, которые могут быть чисто нулевыми или иметь ненулевые элементы.

Факторизация всей матрицы включает факторизацию диагональных блоков и обращение полученных треугольных факторов. Иными словами, диагональный блок на момент факторизации должен быть невырожденным[12]. Во время факторизации диагонального блока $A_{i,i}$ разрешены перестановки строк и столбцов, пересекающиеся с блоком $A_{i,i}$.

Таким образом, при осуществлении LU-факторизации в пакете USPARS выбор главного элемента происходит по схеме Full pivoting по отношению к диагональным блокам. Мы называем такую стратегию Local Full Pivoting.

Процесс факторизации плотного диагонального $m \times m$-блока проиллюстрирован формулой

$$\begin{pmatrix} \times & \times & \times & \cdots & \times \\ \times & a_{kk} & a_{k,k+1} & \cdots & a_{km} \\ \times & a_{k+1,k} & a_{k+1,k+1} & \cdots & a_{k+1,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & a_{mk} & a_{m,k+1} & \cdots & a_{mm} \end{pmatrix} \qquad (C.3)$$

[12] правильнее сказать, хорошо обусловленным

Здесь крестами обозначены уже обработанные строки и столбцы, на очереди обработка диагонального элемента a_{kk}, выделен оставшийся не факторизованным подблок размера $(m-k+1)\times(m-k+1)$. Из-за того, что выбор ведущего элемента ограничен элементами выделенного блока возможна ситуация, когда все элементы этого блока малы по абсолютной величине и не могут быть взяты в качестве ведущих. Отметим, что фактически это может служить сигналом о плохой обусловленности диагонального блока, подвергаемого факторизации, но не обязано означать плохую обусловленность исходной матрицы. Функция `uspars_fact()` в таком случае возвращает код 4 (`Zero pivot`). Такой код сигнализирует о том, что факторизация не была осуществлена из-за того, что в локальной матрице не удалось найти ведущий элемент.

Для преодоления возникающего затруднения в пакете USPARS разработаны специальные приемы, описанные в разделах C.3.1, C.3.2.

C.3.1. Диагональная поддержка

Пусть в процессе факторизации встретилась ситуация `Zero pivot` (функция `uspars_fact()` вернула код 4). Иначе говоря, процесс попал в ситуацию, описываемую формулой (C.3), где все элементы выделенного блока малы

$$|a_{ij}| < \delta_{boost}\|A\|_1, \quad k \le i,j \le m.$$

Здесь $\|A\|_1$ обозначает норму исходной матрицы, которая подвергается факторизации, δ_{boost} – малый положительный параметр. Если блок

$$\begin{pmatrix} a_{kk} & a_{k,k+1} & \cdots & a_{km} \\ a_{k+1,k} & a_{k+1,k+1} & \cdots & a_{k+1,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{mk} & a_{m,k+1} & \cdots & a_{mm} \end{pmatrix}$$

заменить на блок

$$\delta_{boost}\|A\|_1 I_{m-k-1}, \qquad (C.4)$$

пропорциональной единичной матрице порядка $m-k+1$, то факторизацию можно продолжить. Такой прием в англоязычной литературе называется boosting (поддержка). Способ включения опции – см раздел 2.2.

Применяя, если нужно, несколько раз этот прием, факторизацию удается довести до конца. Однако, факторизованной оказывается не исходная матрица, а некоторое ее возмущение. Если вносимое возмущение на шагах поддержки не велико, иначе говоря, не велико δ_{boost}, то можно надеяться, что в результате оказалась факторизованной матрица, близкая к исходной.

Используя полученные треугольные факторы в качестве предобуславливателя, можно решить систему линейных уравнений с исходной матрицей коэффициентов итерационным методом. В пакете USPARS в качестве итерационного метода используется метод простой итерации, в котором невязки вычисляются по исходной системе, а для вычисления следующей итерации используются треугольные факторы, полученные с использованием поддержки. В предположении, что внесенное возмущение было не слишком велико, можно надеяться, что итерации сойдутся, что и подтверждается на многих примерах. Следует иметь в виду, что включение поддержки требует дополнительного времени по сравнению с гипотетической ситуацией, когда все вычисления удалось провести без поддержки.

Использование поддержки проиллюстрировано `example_boosting.c` (см раздел A.21).

C.3.2. Глобальные перестановки

Вводные для описываемого в этом разделе приема те же самые, что и для предыдущего − функция `uspars_fact()` вернула код ошибки 4, выделенный блок

$$\hat{A}_{ii} = \begin{pmatrix} \times & \times & \times & \cdots & \times \\ \times & a_{kk} & a_{k,k+1} & \cdots & a_{km} \\ \times & a_{k+1,k} & a_{k+1,k+1} & \cdots & a_{k+1,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & a_{mk} & a_{m,k+1} & \cdots & a_{mm} \end{pmatrix}$$

в формуле (C.3) является 'нулевым', т. е. удовлетворяющим условию

$$\max_{k \le i,j \le m} |a_{ij}| \quad < \delta_{gp} \max_{1 \le i,j \le N} |a_{ij}| \qquad \text{(C. 5)}$$

Здесь N- порядок исходной матрицы, подвергаемой факторизации. Максимум в правой части неравенства (C.5) вычисляется по элементам исходной матрицы, максимум в левой части – по элементам текущего блока $\hat{A}_{ii}$.

Прием, излагаемый в этом разделе, в применении к ситуации с 'нулевым' подблоком блока A_{ii} можно проиллюстрировать следующим формулами.

$$\left(\begin{array}{ccc|c|ccc|c} A_{11} & \cdots & \times & \times & \times & \cdots & \times & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \times & \cdots & A_{i-1,i-1} & \times & \times & \cdots & \times & 0 \\ \hline \times & \cdots & \times & A_{i,i} & \times & \cdots & \times & 0 \\ \hline \times & \cdots & \times & \times & A_{i+1,i+1} & \cdots & \times & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \times & \cdots & \times & \times & \times & \cdots & A_{nn} & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & I \end{array}\right) \qquad \text{(C. 6)}$$

Здесь выделенная блочная строка и выделенный блочный столбец опираются на 'нулевой' подблок диагонального блока A_{ii}. Чтобы отразить тот факт, что это−часть всего блока, границы нарисованы пересекающими символ A_{ii}. Матрица окаймляется справа блочным столбцом и снизу блочной строкой, ширины которых совпадают с порядком 'нулевого' блока. Все элементы окаймления равны нулю, за исключением правого нижнего блока, где располагается единичная матрица соответствующего размера.

Добавление столбцов в матрицу в терминах решаемой системы линейных уравнений можно трактовать как добавление неизвестных в систему. Дополним вектор правой части снизу нулями так, чтобы число компонент в векторе оказалось равным числу строк в окаймленной матрице. Таким образом, дополнительные строки означают добавленные уравнения.

К расширенной матрице (C.6) применяется симметричная перестановка строк и столбцов так, чтобы добавленный единичный блок из окаймления попал на место 'нулевого' блока. Структура матрицы принимает вид

$$\left(\begin{array}{ccc|c|ccc|c} A_{11} & \cdots & \times & \times & \times & \cdots & \times & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \times & \cdots & A_{i-1,i-1} & \times & \times & \cdots & \times & 0 \\ \hline \times & \cdots & \times & \widetilde{A_{i,i}} & \times & \cdots & \times & 0 \\ \hline \times & \cdots & \times & \times & A_{i+1,i+1} & \cdots & \times & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \times & \cdots & \times & \times & \times & \cdots & A_{nn} & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \hat{A}_{ii} \end{array}\right) \qquad (C.7)$$

На место единичной матрицы в правом нижнем углу попал ’нулевой’ блок, обозначенный $\hat{A}_{ii}$. Чтобы отметить тот факт, что в блоке A_{ii}’нулевой’ блок оказался замещенным единичной матрицей, мы Используем знак тильды. Соответствующие перестановки должны быть сделаны также в векторе правой части.

После проделанного преобразования ситуация, вызывающая Zero pivot исчезает и процесс факторизации может быть продолжен. В случае, если снова встречается Zero pivot, описанные выше преобразования повторяются. Ясно, что таким образом можно дойти до блока A_{nn}, после чего останется некоторое количество добавленных строк и столбцов. Для их исключения следует использовать вычисленную факторизацию модифицированной матрицы, что в конечном счете приведет к задаче факторизации добавленного блока, расположенного в правом нижнем углу. Факторизуя его, получим разложение в треугольные матрицы расширенной матрицы. Решая расширенную систему, мы определим вектор неизвестных, в котором имеются добавленные компоненты. Их следует исключить из ответа.

В случае, если исходная матрица системы была хорошо обусловленной и ошибки Zero pivot появлялись вследствие не слишком удачного выбранного порядка строк и столбцов матрицы, последний нижний блок будет не слишком плохо обусловленным[13] и его факторизация окажется возможной. Практика использования описанного приема подтверждает это. Способ включения опции – см раздел 2.2.

Очевидным недостатком описанного приема является увеличение расхода памяти. Соответственно, увеличивается и время факторизации, если сравнивать его с гипотетическим временем, которое бы потребовалось если бы ошибка Zero pivot не возникала и факторизацию удалось бы осуществить для исходной матрицы.

C.4. Две моды использования пакета USPARS

Возможности пакета USPARS решить конкретную СЛАУ зависят от многих факторов, среди которых не последнюю роль играют характеристики вычислительной системы с общей памятью (характеристики “железа”). В результате факторизации матрицы коэффициентов число ненулевых элементов в треугольных факторах может в десятки или даже сотни раз превосходить число ненулевых элементов исходной матрицы коэффициентов. В процессе решения СЛАУ треугольные факторы являются промежуточными результатами и должны быть сохранены до этапа решения. Недостаточность объема доступной оперативной памяти может оказаться критическим обстоятельством при решении СЛАУ большого размера.

Использование дисковой памяти для временного хранения промежуточных результатов с целью ослабления требований к объему оперативной памяти является эффективным средством с затруднением, описанным в предыдущем абзаце. Такого рода идея носит название Out-Of-Core (OOC) и используется во многих пакетах по решению СЛАУ. В пакете USPARS реализованы две моды вычислений:

[13] Такая осторожная формулировка выбрана из-за того, что соответствующие теоремы, обосновывающие это утверждение, неизвестны. Возможно, что их просто нет.

- In-Core (IC) –мода, в которой все промежуточные результаты сохраняются в оперативной памяти;
- OOC – мода, в которой для хранения части промежуточных результатов используется диск.

Из-за обменов с диском полное время вычислений в OOC моде будет больше, чем на той же вычислительной системе, снабженной дополнительной оперативной памятью. Таким образом, OOC-мода позволяет решать СЛАУ, для которых при использовании IC-моды объем оперативной памяти оказывается недостаточным.

Для того, чтобы выбрать ту или иную моду, следует использовать значение `options[USP_OOC_MEM]` - см раздел 2.2, в котором подробно описано как включить ту или иную моду. Директория, в которой располагаются файлы с промежуточными данными, полученными в процессе факторизации, определена значением переменной окружения `USP_OOC_PATH.` В случае, если такая переменная не задана, используется рабочая директория. Префикс у файлов с промежуточными данными задается через переменную окружения `USP_OOC_NAME.` Если эта переменная не задана, используется имя `ooc_lu`. По окончании работы USPARS созданные файлы всегда удаляются во время вызова функции `uspars_destroy().`

D. Python-интерфейсы USPARS

Интерфейсы USPARS доступны в том числе и на Python под названием PyUSPARS. Из особенностей их использования можно выделить:

- Использование разреженных матриц, заданных в стандартном для Python Scipy CSR формате
- Наличие дополнительных методов – расчета невязки, погрешности, числа обусловленности, а так же выгрузки матрицы задачи после применения перестановок (см. D.3.4)
- Интерфейсы, соответствующие Python-стилю
    - Автоматическое выделение и освобождение памяти, и, как следствие, отсутствие методов uspars_alloc, uspars_destroy
    - Уменьшение числа аргументов методов uspars_init, uspars_fact, uspars_solve (см. D.3.2)
    - Массив options переведен в data descriptors (см. D.3.2)
- Решение при вызове метода solve (см. D.3.2) выдается пользователю отдельно, тогда так uspars_solve (см. 2.4) записывает решение в массив правой части b.

Наличие Scipy-совместимых интерфейсов spsolve и factorized (см. D.2

В Python уже присутствует возможность решения разреженных систем. Следующие библиотеки обеспечивают линейно-алгебраическую функциональность для разреженных матриц, включая решение систем линейных уравнений:

- Scipy Sparse linear algebra
- PySparse

(Оба пакета используют SuperLU и umfpack в качестве прямых решателей.)

- PyMesh - содержит интерфейсы для SparseLU, SparseQR, SuperLU, umfpack и PARDISO
- https://bitbucket.org/petsc/petsc4py/src/master/ - содержит интерфейсы для всех прямых решателей, которые поддерживает PETSc, MUMPS, PARDISO и другие

В действительности этот список длиннее, поскольку многие в индивидуальном порядке для своих проектов самостоятельно пишут обвязки необходимых решателей. PyUSPARS в данном ряду выделяется производительностью на фоне прямых решателей из Scipy и PySparse, и простотой интерфейса в сравнении с PyMesh и petsc4py.

D.1. Пакет PyUSPARS

D.1.1. Информация о пакете

Пакет поддерживает четыре типа чисел с плавающей точкой

C	python
float	numpy.float32
double	numpy.float64
_Complex float	numpy.complex64
_Complex double	numpy.complex128

Чтобы избежать приведения типов и, как следствие, копирования матриц и векторов, типы элементов матрицы и правой части должны совпадать. PyUSPARS предоставляет методы, совместимые с библиотекой scipy.sparse.linalg – пользователи могут заменить на USPARS свои разреженные решатели, импортировав другую библиотеку. Помимо этого, в PyUSPARS присутствует полноценная обвязка под весь пользовательский функционал C-библиотеки USPARS.

D.1.2. Установка

Системные требования:

```
Linux: Ubuntu 20.04+ (либо Redhat 8.3+)
```

```
Windows: Windows 10 и выше
Python 3.7
```

Для создания изолированной python-среды необходимо скачать и запустить установочный скрипт

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.12.0-Linux-x86_64.sh
$ bash Miniconda3-py37_4.12.0-Linux-x86_64.sh
```

использующий `Miniconda`. После установки следует создать новое окружение с требуемыми зависимостями, дождаться их скачивания, и активировать его

```
$ conda create -n usp python=3.7 pip
$ conda activate usp
```

Теперь пакет PyUspars может быть установлен в Python командой

```
$ pip install pyuspars_package-1.0.6-py3-none-any.whl
```

Замечание D.1.1 *Если на этапе импорта солвера*

```
$ python3
>>> from pyuspars_package. solver import PyUspars
```

библиотека mkl оказывается недоступной

```
OSError: libmkl_intel_lp64.so: cannot open shared object file:
No such file or directory,
```

то соответствующую папку необходимо вручную добавить в `LD_LIBRARY_PATH`. *Например, если при установке* `Miniconda` *установочная директория не изменялась, обновить переменную окружения можно следующей командой*

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/miniconda3/envs/usp/lib/
```

D.1.3. Импорт

Импортируется установленный пакет стандартным для Python образом

```
>>> import pyuspars_package
```

D.1.4. Перечисляемые типы данных

Все перечисляемые типы данных USPARS определены в разделе 1.3 (см Таблицы 1.1, 1.2, 1.3). Для типа `usp_factorize_type`, присутствующего в C-интерфейсах USPARS, аналогом является `FACTORIZE_TYPE`. Импортируется она следующим образом

```
>>> from pyuspars_package.solver import FACTORIZE_TYPE
```

В обвязке `usp_scalar_type` необходимости нет, скалярный тип берется из задаваемой матрицы. Все элементы списков `usp_param_data` и `usp_opt` переведены в data descriptors.

D.2. SciPy-интерфейсы PyUSPARS

`spsolve(A, b)`

Референтная функция - `scipy.sparse.linalg.spsolve`.
Решение системы с правой частью. При вызове этого метода будет применена LU-факторизация для решения систем с матрицей коэффициентов общего вида, LDLT-факторизация для симметричных матриц, и LDLH – для эрмитовых. На стороне обвязки контролируется появление ошибки. При ее появлении будет запущен метод `uspars_param_get` для получения расширенного кода. Со стороны USPARS будет выдано сообщение по типу:

```
Error in uspars_fact: Zero pivot has been appeared during factorization process
```

а со стороны обвязки будет выдан `RuntimeError` с указанием кода ошибки, например:

```
Traceback (most recent call last):
File .stdin> line 1, in <module>
RuntimeError: Calculation failed. USPARS error code - 402

>>> from pyuspars_package.scipy import spsolve
>>> x = spsolve(A, b)
```

`factorized(A)`

Референтная функция - scipy.sparse.linalg.factorized.
Возвращает функцию, для решения системы линейных уравнений с факторизованной матрицей коэффициентов и заданным вектором правой части. Она может быть использована многократно для других правых частей.

```
>>> from pyuspars_package.scipy import factorized
>>> solve = factorized(A)
>>> x = solve(B)
```

D.3. Обвязка C-библиотек

В этом разделе описываются методы, которые позволяют использовать в Python весь функционал USPARS (ср. с D.2).

D.3.1. Передача значений массива `options`

В пакете PyUSPARS предусмотрена возможность задания `options` (см. 2.2) для передачи их значений в C-функции USPARS. Для этих целей используются дескрипторы. Каждый из них имеет тип `int`.
Пример задания нулевого уровня ‘говорливости’ и включения итерационного уточнения с ограничением 10 числа итераций приведен в следующем листинге.

```
>>> from pyuspars_package.solver import PyUspars
>>> tt = PyUspars()
>>> tt.msglvl = 0
>>> tt.iter_ref = 10
```

Полный список дескрипторов, используемых для передачи значений опций, их допустимые значения и значения по умолчанию приведены в Таблице D.1. Подробное описание эффектов, достигаемых за счет включения тех или иных опций, см в разделе 2.2 и в описаниях соответствующих приемов. Заметим, если для какого-то дескриптора задать значение, не входящего в список допустимых, то это вызовет переключение значения дескриптора в значение по умолчанию.

D.3.2. Решение линейных систем

`__init__()`

Создание объекта PyUspars. Инициализация USPARS, внутри себя вызывает метод `uspars_alloc` (см. 2.1). При удалении объекта автоматически вызывается деструктор `uspars_destroy` (см. 2.7).

```
>>> from pyuspars_package.solver import PyUspars
>>> tt = PyUspars()
```

	значение	действие
tt.msglvl	0	Нулевое логирование
	1	Отображение текущего статуса факторизации в процентах
	2	Полное логирование
tt.scaling	0	Не использовать балансировку
	1	Использовать балансировку
tt.matching	0	Не использовать matching

	1	Использовать matching с максимизацией минимального диагонального элемента
	2	Использовать matching с максимизацией произведения диагональных элементов
	3	Использовать matching с максимизацией произведения диагональных элементов и балансировкой матрицы
tt.rtype	0	Не использовать перестановки строк и столбцов матрицы с целью уменьшения эффекта заполнения в треугольных факторах
	1	Использовать Atlant для перестановок строк и столбцов матрицы с целью уменьшения эффекта заполнения в треугольных факторах
	2	Использовать Metis для перестановок строк и столбцов матрицы с целью уменьшения эффекта заполнения в треугольных факторах
	3	Использовать внешний вектор перестановок строк и столбцов матрицы с целью уменьшения эффекта заполнения в треугольных факторах (см пример D.4.2)
tt.iter_ref	0	Не использовать итерационное уточнение
	>0	Ограничение числа итераций итерационного уточнения
tt.boosting	0	Не использовать диагональную поддержку
	1	Использовать диагональную поддержку с $\delta_{boost} = 10^{-15}$ и двумя итерациями уточнения
	2	Использовать диагональную поддержку с $\delta_{boost} = 10^{-8}$ и четырьмя итерациями уточнения
tt.global_pivot	0	Не использовать глобальные перестановки
	1	Использовать глобальные перестановки
tt.warn_as_err	0	Выводить предупреждения на экран
	1	Прерывать решение при наличии предупреждений
tt.par_solve	0	Не использовать OMP-параллелизацию solve
	1	Использовать OMP-параллелизацию solve
tt.brx	0	Решением является массив, полученный на последней итерации
	1	Решением является массив, полученный на итерации с наименьшей невязкой
tt.merge_nodes	0	Не объединять в блочном представлении факторов столбцы
	1	Объединять в блочном представлении факторов соседние столбцы шириной больше 4
tt.lu_diag_pivot	0	Использовать local full pivoting для LU
	1	Использовать diagonal pivoting для LU

`init(self, matrix, [ftype, options])`

Стадия инициализации (см. 2.2).

В отличии от C-метода, здесь матрица передается в Scipy CSR формате. Размер матрицы и тип данных берутся непосредственно из **`matrix`**. Параметр **`ftype`** здесь опционален - по умолчанию в качестве типа факторизации будет выбрана LU.

```
>>> import numpy as np
>>> from pyuspars_package.solver import FACTORIZE_TYPE as ftype
>>> tt.rtype = 1
>>> tt.init(A, ftype.LDLH)
```

`fact(self)`

Стадия факторизации (см. 2.3).

Здесь сигнатура метода практически не отличается от C

```
>>> tt.fact()
```

solve(self,b)

Стадия решения (см. 2.4).
Число правых частей и тип данных считываются из **b**. Аргумент задается в формате numpy.array (либо numpy.matrix в виде вектор-столбца) для одной правой части, и в формате numpy.matrix для множественных правых частей.

```
>>> tt.solve(b)
```

D.3.3. Реализация сервиса

Функции `uspars_param_set()` и `uspars_param_get()` пакета USPARS служат для обеспечения доступа/редактирования некоторых полей USP-структуры (см.2.5, 2.6). Средством редактирования в PyUSPARS служит использование дескрипторов данных, список приведен в Таблице D.2. Например, для задания величины параметра малости при использовании диагональной поддержки:

```
>>> tt.boosting_value = 1e-9
```

Таблица D.2: Таблица дескрипторов данных для передачи значений параметров в USPARS

`tt.boosting_value`	вещественное положительное число	Параметр ’малости’ при осуществлении диагональной поддержки C.3.1.
`tt.gp_eps`	`вещественное положительное число`	Параметр ’малости’ при осуществлении диагональной поддержки C.3.2.
`tt.iter_stop_crit`	вещественное положительное число	Критерий остановки по относительной норме невязки при использовании итерационного уточнения C.2
`tt.perm_vector`	Целочисленный массив	Пользовательский вектор перестановки для применения в C-шной функции `uspars_init()`

В процессе вычислений некоторые из дескрипторов данных изменяются. Соответствующие значения можно извлечь. Например, чтобы извлечь число ненулевых элементов в L-факторе, используется команда

```
>>> nnzl = tt.nnzl
```

Это число становится известным после фазы инициализации. Доступные для извлечения дескрипторы данных см в Таблице D.3.

Таблица D.3: Таблица дескрипторов данных для чтения

`tt.boosting_value`	вещественное положительное число	Параметр ’малости’ при осуществлении диагональной поддержки C.3.1.
`tt.gp_eps`	вещественное положительное число	Параметр ’малости’ при осуществлении глобальных перестановок C.3.2
`tt.iter_stop_crit`	вещественное положительное число	Критерий остановки по относительной норме невязки при использовании итерационного уточнения C.2
`tt.perm_vector`	Целочисленный массив	Пользовательский вектор перестановки для применения в C-шной функции `uspars_init()`
`tt.nnzl`	Целое число	После завершения работы функции `uspars_init()` число ненулевых элементов в L-факторе.

D.3.4. Расширенный функционал

calc_residual_norm(self)
Подсчет относительной нормы невязки[14]
```
>>> value=tt.calc_residual_norm()
```

calc_norm_error(self,x)
Подсчет относительной нормы погрешности по заданному вектору точного решения[15]
```
>>> value=tt.calc_norm_error(x_exact)
```

calc_op_criterion(self)
Подсчет критерия Оэттли-Прагера для оценки устойчивости полученного решения
```
>>>value=tt.calc_op_criterion()
```

get_permuted_matrix(self)
Возвращает исходную матрицу после применения к ней перестановок строк и столбцов. Может быть вызвана после стадии инициализации
```
>>> A_perm=tt.get_permuted_matrix()
```

cond(matrix,k,q)
Число обусловленности матрицы (оператора), действующей в евклидовых пространствах, определяется как отношение $\sigma_{max}(A) \Big/ \sigma_{min}(A)$ ее старшего и младшего сингулярных чисел. Функция предоставляет оценку числа обусловленности матрицы. Параметры: k – количество оцениваемых старших и младших сингулярных чисел, q – число итераций. Со стороны USPARS будут факторизованы 2 матрицы (A и A^T), а так же 2*q раз решены системы с k правыми частями. Увеличивая значения этих параметров можно добиться улучшения точности расчета с одновременным увеличением времени вычислений.

```
>>> from pyuspars_package.tools import cond
>>> cond_number = cond(A, k=5, q=20)
```

D.4. Примеры

D.4.1. Использование numpy/scipy интерфейсов

Считывание матриц MatrixMarket
```
>>> from scipy.io import mmread
>>> A_mm = mmread("matrix.mtx").tocsr()
```

Перевод плотной матрицы в CSR формат
```
>>> import numpy as np
>>> from scipy import sparse
>>> A_d = [[16., =2, 0, 0], [0, 4., 0, 0], [0, 0, 0, 5.], [0, 0, 1., 0]]
>>> A = sparse.csr_matrix(A_d)
```

Ручное приведение типа
```
>>> import numpy as np
>>> from scipy import sparse
>>> A_d = [[16., -2, 0, 0], [0, 4., 0, 0], [0, 0, 0, 5.], [0, 0, 1., 0]]
>>> b = np.array([1, 1, 1, 1], dtype=np.complex128)
>>> A = sparse.csr_matrix (A_d, dtype=np.complex128)
```

Решение через spsolve
```
>>> from scipy.io import mmread
>>> import numpy as np
>>> from pyuspars_package.scipy import spsolve
>>> A = mmread("matrix.mtx").tocsr()
>>> x_exact = np.full (A.shape[0], 1)
```

[14] Относительных норм невязок в случае многих правых частей

[15] Относительных норм погрешностей в случае многих правых частей, и, соответственно, многих векторов решений

```
>>> b = A*x_exact
>>> x = spsolve(A, b)
```

Решение через factorized

```
>>> from scipy.io import mmread
>>> import numpy as np
>>> from pyuspars_package.scipy import factorized
>>> A = mmread("matrix.mtx").tocsr()
>>> x_exact = np.full(A.shape[0], 1)
>>> b = A*x_exact
>>> solve=factorized(A)
>>> x = solve(b)
```

D.4.2. Использование USPARS интерфейсов

Следующая заготовка едина для всех запусков

```
>>> from pyuspars_package.solver import PyUspars
>>> from pyuspars_package.solver import FACTORIZE_TYPE as ftype
>>> from scipy.io import mmread
>>> import numpy as np
>>> A = mmread("matrix.mtx").tocsr()
>>> x_exact = np.full(A.shape[0], 1)
>>> b = A*x_exact
```

В ней:
(1, 2) импортируется USPARS;
(3, 4) импортируется метод для считывания матрицы в `mtx` формате и библиотеку `numpy`, которая используется для работы с плотными векторами;
(5) считывается матрица и переводится в CSR формат;
(6) создается вектор точного решения, состоящий из единиц;
(7) умножается матрица на вектор и создается правая часть системы линейных уравнений.

Стандартный запуск решения

```
>>> tt = PyUspars()
>>> tt.init(A, ftype.LU)
>>> tt.fact()
>>> x = tt.solve(b)
```

Решение с полным логированием и бустингом

```
>>> tt.boosting = 2
>>> tt.msglvl = 2
>>> tt = PyUspars()
>>> tt.init(A, ftype.LU)
>>> tt.fact()
>>> x= tt.solve(b)
```

Решение с внешним вектором перестановок

```
>>> perm = list(range(A.shape[0] - 1, -1, -1)) #Задаем вектор перестановок
>>> tt = PyUspars()
>>> tt.rtype = 2
>>> tt.perm_vector = perm  #Передаем его в USPARS
>>> tt.init(A, ftype.LU)
>>> tt.fact()
>>> x = tt.solve(b)
```

Задание критерия остановки итерационного процесса

```
>>> tt = PyUspars ()
>>> tt.iter_ref = 2
>>> tt.iter_stop_crit = 1e-12
>>> tt.init(A, ftype.LU)
>>> tt.fact()
>>> x = solve(b)
```

Визуализация портрета разреженной матрицы после перестановок

См. раздел 1.2.

```
>>> import matplotlib.pylab as plt
>>> tt = PyUspars()
>>> tt.init(A)
>>> plt.spy(tt.get_permuted_matrix())
```

Выгрузка значения nnzl

```
>>> tt = PyUspars()
>>> tt.init(A)
>>> print(tt.nnzl)
```

Расчет погрешности и невязки

```
>>> tt = PyUspars()
>>> tt.init(A, ftype.LU)
>>> tt.fact()
>>> x = tt.solve(b)
>>> print ("discrepancy =", tt.calc_norm_discrepancy())
>>> print ("error =", tt.calc_norm_error(x_exact))
```

E. Переход с MKL PARDISO на USPARS

Данный раздел призван упростить пользователям MKL PARDISO переход с него на USPARS. Здесь будут описаны особенности задания матриц, поставлены в соответствие параметры обоих решателей, а также даны некоторые советы по использованию. Поскольку USPARS не имеет FORTRAN-интерфейсов, а PARDISO не имеет Python-интерфейсов (см. D), в разделе будет описаны примеры на C. Интерфейсы MKL PARDISO соответствуют версии из Intel® oneAPI Math Kernel Library for C 2023.1

E.1. Интерфейсы запуска и стадии решения

В USPARS, как и в PARDISO, процесс вычисления состоит из трех стадий – инициализация, факторизация и решение. Основные отличия заключаются в задании параметров по умолчанию, а также в отсутствии в USPARS возможности разделения стадии решения на три – прямой, диагональный и обратный обход.

E.1.1. Выделение памяти под структуру, задание параметров по умолчанию

В PARDISO функция `pardisoinit(…)` позволяет выделить память под внутреннюю структуру `pt`, задать тип матрицы и заполнить массив параметров `iparm` значениями по умолчанию. В USPARS для этого используется функция `uspars_alloc(…)`, а параметры массива `options` вручную заполняются значениями, равными -1 (см. 2.1).

PARDISO:

```
void *pt[64];
MKL_INT iparm[64];
MKL_INT mtype;
pardisoinit(pt, &mtype, iparm);
```

USPARS:

```
void *tt = NULL;
int options[USP_OPTIONS];
for (int i = 0; i < USP_OPTIONS; i++)
  options[i] = -1;
uspars_alloc(&tt);
```

E.1.2. Запуск стадий решения

PARDISO имеет единый интерфейс для запуска всех стадий решения – `pardiso(…)`. Для выбора стадии используется аргумент phase. В USPARS же они имеют различные названия – `uspars_init(…)`, `uspars_fact(…)`, `usparse_solve(…)` (см. 2.2, 2.3, 2.4).

Ниже проведены соответствия между аргументами функции `pardiso(…)` и параметрами USPARS:

```
void pardiso(_MKL_DSS_HANDLE_t pt, const MKL_INT *maxfct, const MKL_INT
*mnum, const MKL_INT *mtype, const MKL_INT *phase, const MKL_INT *n,
const void *a, const MKL_INT *ia, const MKL_INT *ja, MKL_INT *perm,
const MKL_INT *nrhs, MKL_INT *iparm, const MKL_INT *msglvl, void *b,
void *x, MKL_INT *error);
```

- `pt` – внутренняя структура. Задается аналогично, является аргументом всех функций в USPARS.
- `maxfct, mnum` - не имеют аналога в USPARS. Можно считать, что `maxfct=1`, `mnum=1`.
- `mtype` – тип матрицы, определяет тип факторизации. В USPARS аналогом является `usp_factorize_type` (подробнее см. 2.2). Является аргументом `uspars_init(…)`.

- `phase` – стадия решения. В USPARS этот аргумент отсутствует, поскольку стадия решения определяется названием функции.

phase	*Функция USPARS*
11	`uspars_init(…)`
22	`uspars_fact(…)`
33	`uspars_solve(…)`

- `n` – число неизвестных. В USPARS задается аналогично, является аргументом `uspars_init(…)` (см. 2.2).
- `a, ia, ja` – матрица в CSR формате. В USPARS задается аналогично (см. 1.1), является аргументом `uspars_init(…)` (см. 2.2).
- `perm` – вектор перестановок, может быть использован для задания пользовательской, либо для получения вычисленной перестановки. В USPARS за получение информации, полученной в ходе вычисления, отвечает функция `uspars_param_get(…)` (см. 2.5), а за установку параметров, отличных от целочисленных, отвечает функция `uspars_param_set(…)` (см. 2.6). Более подробно о задании и получении вектора перестановок написано в разделе E.3.
- `nrhs` – число правых частей. В USPARS задается аналогично, является аргументом `uspars_solve(…)` (см. 2.4).
- `iparm` – параметры решателя. В USPARS задаются аналогично, является аргументом `uspars_init(…).` (см. 2.2)
- `msglvl` – уровень логирования. В USPARS этот параметр не выделяется в отдельный аргумент, а является частью массива `options` – `options[USP_MSGLVL]`. Помимо включения и выключения логирования, присутствует промежуточный вариант. (см. 2.2)
- `b, x` – правая часть и решение системы. В PARDISO решение может как заменять, так и не заменять правую часть. В USPARS правая часть является аргументом `uspars_solve(…)` и в ходе вычислений всегда заменяется (см. 2.4).
- `error` – индикатор ошибки. В USPARS является кодом возврата всех функций. Подробнее про возможные значения написано в разделе с соответствующей функцией, а их интерпретации в разделе 3.

Отдельное замечание про использование 64-битных целочисленных интерфейсов. В PARDISO для этого вместо функции `pardiso(…)` используется `pardiso_64(…)`. В USPARS отличия заключаются в стадии инициализации, и данному функционалу соответствует метод `uspars_init64(…)` (см. 2.2).

E.2. Формат матрицы и тип факторизации

USPARS имеет тот же способ задания входной матрицы, что и MKL PARDISO – Sparse CSR, при котором матрицы общего вида задаются стандартным образом, а симметричные и эрмитовы – верхнетреугольным (см. 1.1.2). Другими словами, массивы `a, ia, ja`, передаваемые в функцию `pardiso(…)`, могут быть переданы в том же виде и в функцию `uspars_init(…)` (см. 2.2).

Отличия заключаются в задании типов факторизации. В PARDISO через параметр mtype указывается входной тип матрицы, и исходя из этой информации решатель выбирает факторизацию. В USPARS же тип факторизации задается напрямую (см. Таблицу 2.2.2). Ниже приведено соответствие между типом матрицы, его `mtype` в PARDISO и параметрами в USPARS с учетом скалярного типа матрицы (см. Таблицу 1.3.1)

	PARDISO		*USPARS*	
Тип матрицы	`mtype`	`iparm[27]`	`usp_factorize_type`	`usp_scalar_type`
Вещественная, структурно-симметричная	1	0	`USP_LU`	`USP_DOUBLE`
		1		`USP_FLOAT`
Вещественная, симметричная, положительно-определенная	2	0	`USP_LLT`	`USP_DOUBLE`
		1		`USP_FLOAT`
Вещественная, симметричная	-2	0	`USP_LDLT`	`USP_DOUBLE`
		1		`USP_FLOAT`
Комплексная, структурно-симметричная	3	0	`USP_LU`	`USP_COMPLEX_DOUBLE`
		1		`USP_COMPLEX_FLOAT`
Комплексная, эрмитова, положительно-определенная	4	0	`USP_LLH`	`USP_COMPLEX_DOUBLE`
		1		`USP_COMPLEX_FLOAT`
Комплексная, эрмитова	-4	0	`USP_LDLH`	`USP_COMPLEX_DOUBLE`
		1		`USP_COMPLEX_FLOAT`
Комплексная, симметричная	6	0	`USP_LDLT`	`USP_COMPLEX_DOUBLE`
		1		`USP_COMPLEX_FLOAT`
Вещественная, общего вида	11	0	`USP_LU`	`USP_DOUBLE`
		1		`USP_FLOAT`
Комплексная, общего вида	13	0	`USP_LU`	`USP_COMPLEX_DOUBLE`
		1		`USP_COMPLEX_FLOAT`

E.3. Параметры решателей

PARDISO позволяет управлять параметрами решателя через массив `iparm`. В USPARS этот массив называется `options` (см. 2.2). Дополнительно вещественными параметрами можно управлять с помощью `uspars_param_get(…)` (см. 2.5) и `uspars_param_set(…)` (см. 2.6). Параметры сопоставляются следующим образом:

- `iparm[0]`. Использование параметров по умолчанию.
  В USPARS нет отдельной опции для использования всех параметров по умолчанию. При этом любой параметр options может быть использован по умолчанию, если присвоить ему значение -1.
- `iparm[1]`. Переупорядочивание матрицы.
  В USPARS за это отвечает параметр `options[USP_RTYPE]`. Их значения соотносятся следующим образом

Значение `iparm[1]`	Назначение	Значение `options[USP_RTYPE]`
0	Алгоритм упорядочивания минимальной степени	-
2	METIS	2
3	Собственный параллельный алгоритм (см. B.4)	1
-	Не использовать вектор перестановок	0
см. `iparm[4]`	Пользовательский вектор перестановок	3

- `iparm[3]`. Предобуславливатель для CGS/CG.
  В USPARS функционал отсутствует.
- `iparm[4]`. Параметр, отвечающий за назначение аргумента `perm` функции `pardiso(…)`

Значение `iparm[4]`	Назначение	Использование в USPARS	Время вызова
0	-	-	-
1	Использование пользовательского вектора перестановок	`options[USP_RTYPE]=3`, `uspars_param_set(&tt, USP_PERM_VECTOR, perm)`, где `perm` –	`uspars_param_set(…)` должен быть вызван между `uspars_alloc(…)` и `uspars_init(…)`

		пользовательский вектор перестановок	
2	Получение вектора перестановок	`uspars_param_get(&tt, USP_PERM_VECTOR, perm)`	`uspars_param_get(…)` должен быть вызван после `uspars_init(…)`

- `iparm[5]`. Запись решения в отдельный массив.
  В USPARS решение всегда пишется в массив, содержащий правую часть b, что соответствует `iparm[5]=1`.
- `iparm[6]`. Выходной параметр. Итоговое число шагов итерационного уточнения.
  В USPARS всю информацию о ходе вычисления можно получить в расширенном логировании при `options[USP_MSGLVL]=2`.
- `iparm[7]`. Максимальное число шагов итерационного уточнения.
  В USPARS регулируется параметром `options[USP_ITER_REF]` (см. 2.2, C.2). При значении, равном нулю, итерационное уточнение не применяется. Кроме этого, в случае, если сходимость процесса неравномерна, по опции `options[USP_BRX]=1` можно использовать решение, полученное на итерации с наименьшей невязкой.
- `iparm[8]`. Критерий сходимости итерационного процесса.
  В USPARS эта величина является вещественной, и задается через `uspars_param_set(&tt, USP_ITER_STOP_CRIT, value)`, где `value` – значение относительной невязки, при котором итерационный процесс объявляется сошедшимся.
- `iparm[9]`. Возмущение малых диагональных элементов в процессе факторизации.
  В USPARS аналогом этого параметра является `options[USP_BOOSTING]`. При этом он, в отличии от PARDISO, влияет на число итерационных шагов (см. 2.2, C.3.1). Напрямую же изменить величину малости элементов можно через `uspars_param_set(&tt, USP_BOOSTING_VALUE, value)`. Также в USPARS имеется еще один способ избавления от малых диагональных элементов (см. C.3.2)
- `iparm[10], iparm[12]`. Matching, балансировка матрицы
  В PARDISO `iparm[12]` отвечает за Matching, а `iparm[10]` зависящую от нее балансировку. В USPARS за это отвечает один параметр (см. 2.2, B.3).

Значение `iparm[10]`	Значение `iparm[12]`	Назначение	Значение `options[USP_MATCHING]`
0	0	Matching отключен	0
0	1	Matching включен	2
1	1	Matching включен вместе с балансировкой	3

Также в USPARS помимо этого присутствуют еще и дополнительная независимая балансировка (`options[USP_SCALING]=1`, см. B.2), а также альтернативный Matching (`options[USP_MATCHING]=1`).

- `iparm[11]`. Решение сопряженных систем.
  В данной версии функционал отсутствует, находится в разработке.
- `iparm[13]`, `iparm[14]`, `iparm[15]`, `iparm[16]`, `iparm[17]`, `iparm[18]`, `iparm[19]`. Выходные параметры.
  В USPARS всю информацию о ходе вычисления можно получить в расширенном логировании при `options[USP_MSGLVL]=2`.
- `iparm[20]`. 2x2 пивотинг симметричных матриц.
  В USPARS данная опция отсутствует. Можно считать, что `iparm[20]=0`.
- `iparm[21]`, `iparm[22]`. Выходные параметры. Подсчет числа положительных/отрицательных собственных значений.
  В USPARS данный функционал отсутствует.
- `iparm[23]`, `iparm[24]`. Управление параллелизацией.
  В USPARS отключение/изменение алгоритмов параллелизации не предусмотрено. Подробнее в разделе C.1.
- `iparm[26]`. Проверка матрицы перед решением.
  USPARS проводит различные проверки (см. 3.2), управление ими не предусмотрено.
- `iparm[27]`. Одинарная/двойная точность.
  Регулируется аргументом функции `uspars_init(…)`.
  Подробнее в разделе E.2.
- `iparm[29]`. Выходной параметр. Число отрицательных пивотов при ошибке -4.
  USPARS в аналогичной ситуации выдает ошибку без дополнительной информации.
- `iparm[30]`. Частичное решение.
  USPARS всегда находит полное решение и данный функционал отсутствует.
- `iparm[33]`. Режим CNR.
  Является разработкой Intel, в USPARS отсутствует.
- `iparm[34]`. Индексация массивов индексов строк и столбцов.
  В USPARS индикация начинается с нуля. Можно считать, что `iparm[34]=1`.
- `iparm[35]`. Выдача пользователю дополнения Шура.
  В USPARS пользователю дополнение Шура не выдается. Если вы являетесь пользователем USPARS и вам необходим этот функционал, свяжитесь с разработчиками.
- `iparm[36]`. Формат исходной матрицы.
  USPARS поддерживает только CSR формат. Можно считать, что `iparm[36]=0`.

- `iparm[38]`. Low-Rank Update.
  В USPARS данный функционал отсутствует.
- `iparm[42]`. Выдача пользователю диагонали обратной матрицы.
  В USPARS данная возможность отсутствует. Если вы являетесь пользователем USPARS и вам необходим этот функционал, свяжитесь с разработчиками.
- `iparm[55]`. Контроль над диагональными элементами
  В USPARS данная возможность отсутствует. Если вы являетесь пользователем USPARS и вам необходим этот функционал, свяжитесь с разработчиками.
- `iparm[59], iparm[62]`. Режим OOC.
  В PARDISO выделяемая пользователем величина оперативной памяти под OOC определяется переменной окружения `MKL_PARDISO_OOC_MAX_CORE_SIZE`, тогда как в USPARS эта величина задается в опциях напрямую. Контроль между автоматическим и принудительным OOC выполнятся с помощью знака.

Значение `iparm[59]`	`MKL_PARDISO_OOC_MAX_CORE_SIZE`	Назначение	Значение `options[USP_MEM_OOC]`
0	-	IC режим.	0
1	value	Автоматический режим IC/OOC	value
2	value	OOC режим	-value

Замечания для пользователей предыдущих версий USPARS

- Начиная с USPARS 2.2.1 появилась свободно распространяемая версия пакета, имеющая ограниченную поддержку и функционал. OOC режим и 64-битные целочисленные интерфейсы в эту версию не входят.

- Опция `options[USP_LU_DIAG_PIVOT]` исключена из USPARS 2.1.0, поскольку оказывала влияние только на симметричные матрицы, которые решались с LU разложением. Если при ее использовании удавалось достичь улучшения точности, советуем в дальнейшем пользоваться LDLT-разложением для симметричных матриц

- В версии 2.1.0 исключена возможность использования значения

  `USP_GET_LU` Параметр для получения в плотном виде треугольных факторов матрицы при условии, что при выполнении вычислений не использовались никакие перестановки

  аргумента id функции `uspars_param_get()`.

Список литературы

[1] Pieter Ghysels, Xiaoye S Li, Francois-Henry Rouet, Samuel Williams, and Artem Napov. An efficient multicore implementation of a novel hss-structured multifrontal solver using randomized sampling. SIAM Journal on Scientific Computing, 38(5):S358–S384, 2016.

[2] STRUMPACK (STRUctured Matrix PACKage), https://portal.nersc.gov/project/sparse/strumpack/index.html

[3] Intel® Math Kernel Library (Intel® MKL), https://software.intel.com/en-us/intel-mkl, 2017.

[4] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 20(1):359–392, 1998.

[5] Joseph W.H. Liu. Modification of the minimum-degree algorithm by multiple elimination. ACM Transactions on Mathematical Software (TOMS), 11(2):141–153, 1985.

[6] Pinar Heggernes, SC Eisestat, Gary Kumfert, and Alex Pothen. The computational complexity of the minimum degree algorithm. Technical report, Institute for computer applications in science and engineering, Hampton, VA, 2001.

[7] George Karypis and Vipin Kumar. Metis, a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 4.0. 1998.

[8] Iain S Duff and Jacko Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. SIAM Journal on Matrix Analysis and Applications, 20(4):889–901, 1999.

[9] George Karypis and Vipin Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. Journal of parallel and distributed computing, 48(1):71–95, 1998.