УДК 519.688 DOI: 10.18698/1812-3368-2017-2-4-11

ИСПОЛЬЗОВАНИЕ ФОРМАТОВ ХРАНЕНИЯ РАЗРЕЖЕННЫХ МАТРИЦ ПРИ РЕАЛИЗАЦИИ МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ

А.И. Богоявленский albg83@yandex.ru

Военно-космическая академия им. А.Ф. Можайского Министерства обороны Российской Федерации, Санкт-Петербург, Российская Федерация

Аннотация

Применение метода конечных элементов сводит краевую задачу для уравнения в частных производных к решению системы линейных алгебраических уравнений в матричной форме. По построению, матрица коэффициентов системы линейных алгебраических уравнений (также называемая матрицей жесткости) является разреженной. Выделение памяти для хранения разреженной матрицы коэффициентов в полной форме оказывается чрезвычайно неэффективным решением, в некоторых случаях делающим использование метода конечных элементов невозможным вследствие ограничений по доступной памяти. Существует ряд форматов представления разреженных матриц, предназначенных для их хранения и использования с максимальной эффективностью. Известные реализации таких форматов, как *CCS* (*compressed column storage*) разработаны в предположении, что сохраняемая матрица доступна в полной форме, и *CCS* создается из нее. Предложена дополнительная структура данных и алгоритмы, позволяющие инициализировать *CCS* до начала сборки матрицы коэффициентов с тем, чтобы собирать матрицу коэффициентов с записью ненулевых коэффициентов непосредственно в формате *CCS*, минуя стадию инициализации матрицы коэффициентов в полной форме

Ключевые слова

Метод конечных элементов, разреженные матрицы, форматы хранения разреженных матриц

Поступила в редакцию 18.08.2016

Введение. Система линейных алгебраических уравнений (СЛАУ) для n неизвестных в матричной форме записывается как

$$Kx = f, \tag{1}$$

где K — матрица коэффициентов СЛАУ размерностью $n \times n$; x — вектор неизвестных; f — вектор правой части.

Система линейных алгебраических уравнений вида (1) с разреженной матрицей коэффициентов имеет место при решении краевой задачи для уравнения в частных производных методом конечных элементов (МКЭ). При использовании МКЭ для расчетной области генерируется сетка — совокупность распреде-

ленных по расчетной области узлов, объединенных в элементы. Распределение узлов по элементам называют сеточной связностью. Сеточная связность определяет портрет матрицы K, т. е. множество пар индексов строк и столбцов ее ненулевых коэффициентов [1]. Портрет матрицы K вычисляется так называемой процедурой сборки. Во время исполнения сборки определяются матрицы коэффициентов для отдельных элементов, которые затем суммируются в матрицу K по схеме, однозначно задаваемой сеточной связностью. В настоящей работе рассмотрена только часть процедуры сборки, связанная с определением портрета матрицы K.

По построению, только те коэффициенты K_{ij} являются ненулевыми, индексы (i, j) которых соответствуют узлам сетки, соединенным ребрами. Поскольку каждый узел соединен лишь с несколькими другими узлами и общее число связей между узлами пропорционально числу n, а число коэффициентов в матрице K равно n^2, матрица K является разреженной, т. е. число ненулевых коэффициентов N_{nz} на порядки меньше, чем нулевых.

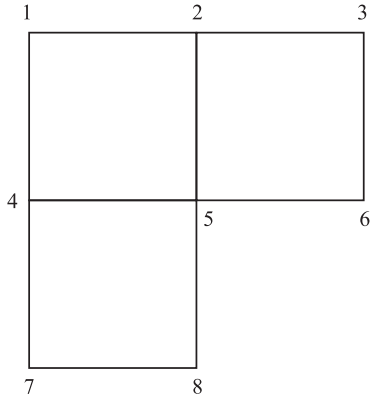
Разреженную матрицу целесообразно хранить не в полной форме (массива чисел размерностью $n \times n$), а в некоторой структуре данных, которая максимально использует разреженность для экономии памяти, и в то же время позволяет использовать сохраненную матрицу так же, как если бы она оставалась в полной форме.

К числу наиболее эффективных по затратам памяти и скорости использования в вычислениях форматам данных для хранения разреженных матриц относится формат компактного хранения по столбцам *CCS* (*compressed column storage*), предложенный в работе [2]. Формат *CCS* хранит только ненулевые коэффициенты, их строковые индексы и $n+1$ вспомогательных индексов. Пренебрегая разностью байтов, отводимых на хранение в памяти компьютера целых чисел и чисел с плавающей точкой, потребление памяти матрицей, сохраненной в формате *CCS*, оценивается как $2N_{nz}+n+1$. Для формата *CCS* построены алгоритмы умножения матрицы на вектор и извлечения матричной диагонали, работающие без «распаковки» матрицы в полную форму. Указанных алгоритмов операций достаточно для реализации решения системы (1) методом сопряженных градиентов с диагональным предобуславливателем, когда матрица K симметрична, и аналогичными методами, когда матрица K несимметрична [3].

По мнению автора настоящей работы, в программной реализации вычислений с форматом *CCS* [4] есть существенный изъян с точки зрения использования этой структуры данных для хранения и решения СЛАУ МКЭ. Предполагается, что формат *CCS* инициализируется из матрицы, сохраненной в полной форме. Подход, изложенный в данной работе, основан на наблюдении, что в сеточной связности содержится достаточно информации для инициализации формата *CCS* заранее с тем, чтобы во время сборки матрицы K сохранять ее ненулевые коэффициенты прямо в формате *CCS*, исключив стадию инициализации памяти для хранения матрицы K в полной форме. Последнее открывает возможность решать сеточные задачи МКЭ большой размерности на обычных компьютерах.

Формальная постановка задачи. Рассмотрим простую двумерную сетку (рисунок) и собранную на ее основе матрицу K коэффициентов СЛАУ МКЭ в полной символьной форме:

$$\begin{bmatrix} k_{11} & k_{12} & 0 & k_{14} & 0 & 0 & 0 & 0 \\ k_{21} & k_{22} & k_{23} & 0 & k_{25} & 0 & 0 & 0 \\ 0 & k_{32} & k_{33} & 0 & 0 & k_{36} & 0 & 0 \\ k_{41} & 0 & 0 & k_{44} & k_{45} & 0 & k_{47} & 0 \\ 0 & k_{52} & 0 & k_{54} & k_{55} & k_{56} & 0 & k_{58} \\ 0 & 0 & k_{63} & 0 & k_{65} & k_{66} & 0 & 0 \\ 0 & 0 & 0 & k_{74} & 0 & 0 & k_{77} & k_{78} \\ 0 & 0 & 0 & 0 & k_{85} & 0 & k_{87} & k_{88} \end{bmatrix}. \quad (2)$$



Модельная сетка

Структура формата *CCS* состоит из трех одномерных массивов, которые обозначим как *vals(), row_inds()* и *col_ptr()* [4]. Массив *vals()* хранит ненулевые коэффициенты матрицы K, массив *row_inds()* — их строчные индексы, для каждого столбца матрицы массив *col_ptr()* — указатель на индекс ненулевого коэффициента массива *vals()*, являющийся последним в этом столбце. Пример заполненных массивов *CCS* для матрицы (2) приведен в таблице.

Содержимое формата *CCS* для матрицы (2)

vals()													
k_{11}	k_{21}	k_{41}	k_{12}	k_{22}	k_{32}	k_{52}	k_{23}	k_{33}	k_{63}	k_{14}	k_{44}	k_{54}	k_{74}
k_{25}	k_{45}	k_{55}	k_{65}	k_{85}	k_{36}	k_{56}	k_{66}	k_{47}	k_{77}	k_{87}	k_{58}	k_{78}	k_{88}
row_inds()													
1	2	4	1	2	3	5	2	3	6	1	4	5	7
2	4	5	6	8	3	5	6	4	7	8	5	7	8
col_ptr()													
0	*3*	*7*	*10*	*14*	*19*	*22*	*25*	*28*					

Для предварительной инициализации формата *CCS* необходимо заранее определить число ненулевых коэффициентов и портрет полной матрицы. При расчете портрета полной матрицы следует учесть, что ее ненулевые коэффициенты могут складываться из нескольких коэффициентов элементных матриц, и что

полное число ненулевых элементов для каждого столбца или строки заранее неизвестно.

Для предварительного вычисления портрета матрицы предложено применять вспомогательную структуру данных, использующую связные списки для хранения строковых индексов ненулевых коэффициентов матрицы K. Число связных списков равно числу столбцов. Связные списки соотнесены со столбцами матрицы в полной форме в порядке перечисления. Каждый связный список содержит индексы строк ненулевых коэффициентов своего столбца, упорядоченные по возрастанию. Пример такой структуры для матрицы (2) приведен ниже (для наглядности матрица дополнена первой строкой с номерами столбцов):

1	2	3	4	5	6	7	8
1	1	2	1	2	3	4	5
2	2	3	4	4	5	7	7
4	3	6	5	5	6	8	8.
	5		7	6			
				8			

Вспомогательная структура заполняется обходом таблицы сеточной связности по тому же алгоритму, который используется при построении матрицы K, с тем различием, что ненулевые коэффициенты не вычисляют, а определяют только их индексы. Для ее формирования традиционная реализация алгоритма добавления элементов в связный список дополняется с учетом указанных выше особенностей [5]. Учитывается, что индекс ненулевого коэффициента должен содержаться в списке только один раз, и что добавленные в список индексы должны быть отсортированы по возрастанию.

Далее приведены алгоритмы формирования и использования описанной структуры.

Алгоритм решения задачи. Алгоритмы выполняют в псевдокоде в том стиле, который задан в работе [5]. Выбранный стиль псевдокода позволяет абстрагироваться от специфики конкретного языка программирования. Единственным безусловным требованием к языку программирования для реализации проводимых ниже алгоритмов является поддержка работы с указателями, что необходимо для реализации связного списка, либо наличия в языке встроенных структур данных, сочетающих упорядоченное индексированное хранение данных с возможностью произвольной вставки по индексу (например, контейнер *list()* в языке *Python*).

В отличие от массива, размерность связного списка не определена на момент инициализации, он может пополняться элементами в произвольном порядке, в том числе возможна вставка элемента между двумя другими. Связный список хорошо подходит к решению проблемы о сохранении индексов строк ненулевых коэффициентов, так как на начало работы с сеткой полное число ненулевых коэффициентов в строках матрицы неизвестно.

Листинг рекурсивного алгоритма добавления строкового индекса ненулевого коэффициента в связный список в псевдокоде выглядит следующим образом:

```
insert(L, i)
1. if head[L] = NIL then
2.  head[L] = i
3.  return
4. else if head[L] = i then
5.  return
6. else if i < head[L] then
7.  tmp = L
8.  head[L] = i
9.  tail[L] = tmp
10. else
11.  insert(tail[L],i)
```

Здесь L — связный список; i — добавляемый строковый индекс ненулевого коэффициента; *head[L]* — указатель на текущий элемент списка; *tail[L]* — указатель на последующие элементы списка; *tmp* — вспомогательный указатель для выполнения операции вставки.

На основе приведенного алгоритма строится алгоритм добавления во вспомогательную структуру номеров узлов одного конечного элемента сетки:

```
add_elem(S, elem)
1. for i = 1:Nn do
2.  for j = 1:Nn do
3.  col = elem[i]
4.  row_ind = elem[j]
5.  insert(S[col],row_ind)
```

где *elem* — перечень узлов конечного элемента; *Nn* — число узлов, образующих конечный элемент; *S* — формируемая вспомогательная структура, массив связных списков, каждый список соответствует столбцу матрицы в полной форме; *col, row_ind* — индексы столбца и строки ненулевого коэффициента текущей пары узлов.

Для формирования вспомогательной структуры приведенную выше подпрограмму применяют ко всей таблице сеточной связности:

```
support_create(S, elems)
1. for e = 1:Nelem do
2.  add_elem(S, elems[e])
```

Здесь *elems* — таблица сеточной связности; *Nelem* — число элементов сетки.

После формирования вспомогательной структуры она используется, прежде всего, для определения полного числа ненулевых коэффициентов СЛАУ МКЭ:

```
get_total_nonzeros(S)
1. total = 0
2. for j = 1:N do
```

3. *total = total + length[S[j]]*
4. **return** *total*

Здесь *total* — возвращаемая переменная-счетчик, полное число ненулевых коэффициентов; *length[]* — возвращает полное число ненулевых коэффициентов в данном столбце.

Формат *CCS* инициализируется заданием для ее массивов размерностей

vals(total)

row_inds(total)

col_ptrs(N)

после чего заполняются массивы *row_inds()* и *col_ptr()*:

insert_row_inds(row_inds, S)

1. *k = 0*
2. **for** *j = 1:N* **do**
3. **for** *i = 1:length[S[j]]* **do**
4. *row_ind = S[j][i]*
5. *k = k + 1*
6. *row_inds[k] = row_ind*

insert_col_ptr(col_ptr, S)

1. *col_ptr[1] = length(S[j])*
2. **for** *j = 2:N* **do**
3. n*rows = length(S[j])*
4. *col_ptr[j] = col_ptr[j-1]+nrows*

Из сохраненной во вспомогательной структуре информации для любой пары индексов ненулевых элементов (*i, j*) вычисляют индекс *k* его размещения в массиве *vals()*:

map_i_j_to_k(*S, col_ptrs, i, j*)

1. *k = col_ptrs[j]*
2. **for** *r = 1:length[S[j]]* **do**
3. **if** *S[j][r] ≠ i* **then**
4. *k = k + 1*
5. **return** *k*

Представленные выше вспомогательная структура и алгоритмы позволяют собрать матрицу *K* непосредственно в формате *CCS*.

Заключение. Изложенный подход к формированию матрицы коэффициентов СЛАУ МКЭ рассмотрен на примере одного из форматов эффективного хранения разреженных матриц — *CCS*. Возможно применение других форматов (координатный (*coordinate*, *COO*), сжатый диагональный (*compressed diagonal*, *CDS*) или *skyline* [6]). Существенно, что можно вычислять и использовать портрет матрицы коэффициентов до этапа ее сборки, исключая этап инициализации матрицы коэффициентов в полной форме. Последнее на порядок снижает потребность выделения памяти при решении задач МКЭ, и тем в большей степени, чем больше размерность сетки.

ЛИТЕРАТУРА

1. *Писсанецки С.* Технология разреженных матриц / пер. с англ. М.: Мир, 1988. 410 с.

2. *Iain S. Duff, Roger G. Grimes, John G. Lewis.* Sparse matrix test problems // ACM Transactions on Mathematical Software. 1989. Vol. 15. No. 1. P. 1–14.

3. *Saad Y.* Iterative methods for sparse linear systems. 2nd ed. SIAM, 2003. 528 p.

4. *Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.* Numerical recipes the art of scientific computing. Cambridge University Press., 2007. 1256 p.

5. *Кормен Томас Х., Лейзерсон Чарльз И., Ривест Рональд Л., Штайн Клиффорд.* Алгоритмы. Построение и анализ / пер. с англ. М.: Издательский дом «Вильямс», 2005. 1296 с.

6. *Bathe Klaus-Jürgen*. Finite element procedures. Prentice Hall, 1995. 1037 p.

Богоявленский Александр Игоревич — младший научный сотрудник Военно-космической академии им. А.Ф. Можайского Министерства обороны Российской Федерации (Российская Федерация, 197198, Санкт-Петербург, ул. Ждановская, д. 13).

Просьба ссылаться на эту статью следующим образом:

Богоявленский А.И. Использование форматов хранения разреженных матриц при реализации метода конечных элементов // Вестник МГТУ им. Н.Э. Баумана. Сер. Естественные науки. 2017. № 2. С. 4–11. DOI: 10.18698/1812-3368-2017-2-4-11

USAGE OF SPARSE MATRIX STORAGE FORMAT IN IMPLEMENTATION OF FINITE ELEMENT METHOD

A.I. Bogoyavlenskiy albg83@yandex.ru

Mozhaysky Military Space Academy, Saint-Petersburg, Russian Federation

Abstract

Finite element method (FEM) applied to a partial differential equation problem yields a system of algebraic equations in matrix form. Matrix of coefficients for the system (also known as stiffness matrix or nodal assembly matrix) is sparse. Memory allocation to keep and use the stiffness matrix in full form appears to be extremely inefficient in terms of memory usage, in some cases making it impossible to apply FEM because of available memory limitations. There are multiple storage formats purposed to keep and use a sparse matrix with maximum efficiency. However, known algorithmic implementations of such a storage formats like compressed column storage (CCS) are designed with assumption that a sparse matrix is available in full form and CCS is initialized from it. This paper describes a data structure and related algorithms making it possible to initialize CCS before the beginning of the stiffness matrix assembly stage, so that non-zero coefficients are written directly into CCS skipping memory allocation for stiffness matrix in full form

Keywords

Finite element method, sparse matrix, sparse matrix storage format

REFERENCES

[1] Pissanetzky S. Sparse matrix technology. Academic Press, 1984. 336 p. (Russ. ed.: Tekhnologiya razrezhennykh matrits. Moscow, Mir Publ., 1988. 410 p.).

[2] Iain S. Duff, Roger G. Grimes, John G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 1989, vol. 15, no. 1, pp. 1–14.

[3] Saad Y. Iterative methods for sparse linear systems. 2nd ed. SIAM, 2003. 528 p.

[4] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical recipes the art of scientific computing. Cambridge University Press., 2007. 1256 p.

[5] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to algorithms. 2nd ed. McGraw-Hill Science/Engineering/Math, 2001. 1056 p. (Russ. ed.: Algoritmy. Postroenie i analiz. Moscow, Wil'yams Publishing House, 2005. 1296 p.).

[6] Bathe Klaus-Jürgen. Finite element procedures. Prentice Hall, 1995. 1037 p.

Bogoyavlenskiy A.I. — Junior Researcher Scientist of Mozhaysky Military Space Academy (Zhdanovskaya ul. 13, Saint-Petersburg, 197198 Russian Federation).

Please cite this article in English as:

Bogoyavlenskiy A.I. Usage of Sparse Matrix Storage Format in Implementation of Finite Element Method. *Vestn. Mosk. Gos. Tekh. Univ. im. N.E. Baumana, Estestv. Nauki* [Herald of the Bauman Moscow State Tech. Univ., Nat. Sci.], 2017, no. 2, pp. 4–11.
DOI: 10.18698/1812-3368-2017-2-4-11