Лекция 3: Метод неполной факторизации (ILU)

(конспект Алексея Легкого, МФТИ)

1 Введение

Один из возможных методов построения предобуславливателя основан на идее выполнения $A = LU$ разложения матрицы, но выполнения этого разложения не точно, а с отбрасыванием части элементов матриц L и U. Напомним алгоритм Гаусса выполнения этого разложения:

$$A = \begin{pmatrix} a & c^T \\ b & D \end{pmatrix} \underset{\text{итерация Гаусса}}{\rightarrow} \begin{pmatrix} 1 & c^T \\ -z & I \end{pmatrix} \begin{pmatrix} a & c^T \\ b & D \end{pmatrix} = \begin{pmatrix} a & c^T \\ 0 & A_1 \end{pmatrix}, z = \frac{b}{a}, A_1 = D - zc^T$$

$$L = \begin{pmatrix} 1 & 0 \\ z & L_1 \end{pmatrix} U = \begin{pmatrix} a & c^T \\ 0 & U_1 \end{pmatrix}, \; L_1 U_1 = A_1$$

Заметим, что у матрицы L диагональ единичная и её можно не хранить, тогда для хранения L и U можно использовать единую матрицу, где в верхней диагональной части будет U, а под диагональю часть L без диагонали. В таком случае для простоты опуская некоторые шаги вычислений, алгоритм LU разложения может быть записан как:

Алгоритм 1 KIJ версия LU факторизации

цикл для $k = 1 : n - 1$ **выполним**
  цикл для $i = k + 1 : n$ **выполним**
    цикл для $j = k + 1 : n$ **выполним**
      $a_{ij} = a_{ij} - a_{ik} * a_{kj}$
    завершим цикл для
  завершим цикл для
завершим цикл для

Это так называемая KIJ версия алгоритма, однако циклы можно переставить местами и получить IKJ версию алгоритма. Хотя ILU может быть получен через обе версии на практике оказывается, что IKJ версия значительно упрощает необходимые для проведения вычислений структуры данных, в частности KIJ LU требует на каждом k-ом шаге производить 1-ранговое обновление $n - k$ строк и столбцов что является очень тяжёлой операций для разреженных матриц. Выпишем упрощённую IKJ версию:

цикл для $i = 2 : n$ **выполним**
  цикл для $k = 1 : i - 1$ **выполним**
    цикл для $j = k + 1 : n$ **выполним**
      $a_{ij} = a_{ij} - a_{ik} * a_{kj}$
    завершим цикл для
  завершим цикл для
завершим цикл для

2 Краут (Crout) версия LU факторизации

Приведём класическую формулировку Краут версии Гауссова исключения переменных. На k-ом шаге вычисляются элементы $a_{k+1:n,k}$ (части нижнетрегульной L) и $a_{k,k:n}$ (часть верхнетреугольной матрицы U), а 1-ранговое обновление остальной подматрицы откладывается, притом к $a_{k+1:n,k}$ и $a_{k,k:n}$ на этом шаге применяются все накопленные с предыдущих шагов 1-ранговые обновления. Также предполагается, что L хранится по столбцам, а U по строкам.

Алгоритм 2 Краут LU факторизация

1: **цикл для** $k = 1 : n$ **выполним**
2: **цикл для** $\{i | i \in 1 : k-1$ и $a_{ki} \neq 0\}$ **выполним**
3: $a_{k,k:n} = a_{k,k:n} - a_{ki} * a_{i,k:n}$
4: **завершим цикл для**
5: **цикл для** $\{i | i \in 1 : k-1$ и $a_{ik} \neq 0\}$ **выполним**
6: $a_{k+1:n,k} = a_{k+1:n,k} - a_{ik} * a_{k+1:n,i}$
7: **завершим цикл для**
8: $a_{ik} = a_{ik}/a_{kk}$, $\forall i \in k+1 : n$
9: **завершим цикл для**

Заметим, что при работе алгоритма доступ к элементам U осуществляется по строкам, а к L по столбцам, поэтому при реализации разреженных структур данных для хранения этих матриц можно использовать, например, аналоги CSR и CSC форматов хранения матриц соответственно. Модифицируем алгоритм 2 для разреженных матриц и добавим стратегию отбора элементов.

Алгоритм 3 Краут версия ILUC

1: **цикл для** $k = 1 : n$ **выполним**
2: пусть строка z: $z_{1:k-1} = 0$, $z_{k:n} = a_{k,k:n}$
3: **цикл для** $\{i | i \in 1 : k-1$ и $l_{ki} \neq 0\}$ **выполним**
4: $z_{k:n} = z_{k:n} - l_{ki} * u_{i,k:n}$
5: **завершим цикл для**
6: пусть столбец w: $w_{1:k} = 0$, $w_{k+1:n} = a_{k+1:n,k}$
7: **цикл для** $\{i | i \in 1 : k-1$ и $u_{ik} \neq 0\}$ **выполним**
8: $w_{k+1:n} = w_{k+1:n} - u_{ik} * l_{k+1:n,i}$
9: **завершим цикл для**
10: Применим стратегию отбора элементов для строки z
11: Применим стратегию отбора элементов для столбца w
12: $u_{k,:} = z$
13: $l_{:,k} = w/u_{k,k}$, $l_{k,k} = 1$
14: **завершим цикл для**

3 Эффективная реализация

Представленный выше алгоритм имеет 2 проблемных с точки зрения структур разреженных матриц места:

- В строках 4 и 8 требуется только $k : n$ часть i-ой строки U и $k+1 : n$ для L. Отыскание с какого элемента эта часть начинается в разреженной матрице может быть дорогостоящей операцией.
- В строках 3 и 7 можно видеть, что производится доступ к ненулям k-ой строки L и к ненулям k-ого столбца U, хотя эти матрицы хранятся в столбцовом и строчном формате соответственно. Поэтому необходимо, чтобы этот доступ не был сложным.

Далее все структуры будем описывать только для U, потому что для L они аналогичны.

Будем предполагать, что ненули в строках U хранятся в порядке возрастания по номеру столбца. Тогда можно завести массив указателей Ufirst(n), где на k-ом шаге Ufirst(j) (для $j < k$) указывает на номер первого ненулевого элемента в j-ой строке, который должен использоваться для обновления k-ой строки. Этот массив должен обновляться после каждой итерации при необходимости инкрементируя своё значение, также после k-ой итерации добавляется указатель для k-ой строки.

Для решения второй проблемы введём неявный связанный список Ulist ненулевых элементов в столбце k матрицы U. Ulist(k) содержит номер первого ненуля в k-ом столбце и Ulist(Ulist(k)) содержит номер следующего и т.д. В конце k-ого шага Ulist обновляется так, чтобы стать связанным списком для $k + 1$-го столбца. Ulist обновляется при обновлении Ufirst: когда Ufirst(i) увеличивается, чтобы указывать на ненулевое значение с индексом столбца c, тогда i добавляется в связанный список для столбца c.

Итого необходимо ввести всего 4 дополнительных массива длины n: Ufirst, Ulist, Lfirst, Llist, – которые составляют так называемую *би-индексную структуру*

4 Немного о стратегиях отбрасывания

Можно придумать множество различных алгоритмов отбрасывания, в т.ч. основанных на примерной оценки числа обусловленности предобусловленной матрицы, однако здесь приведём лишь простейшую стратегию, которая аналогична используемой в ILUT:

- Каждый элемент L и U, чья величина меньше предзаданного τ (относительно нормы k-ого столбца и k-ой строки соответственно) отбрасывается
- В k-ом столбце L оставляется не более Lfil самых больших по величине элементов, остальные отбрасываются, аналогично и для строк U.