There are the results of numerical experiments using a number of benchmark problems, in each of which the refinement of space and angular meshes varied. The results produced with a new approach and the ones obtained using DD-scheme were analyzed and compared.

Key words: 1D transfer equation, 1D neutrons transfer equation, 1D one-group stationary particles transfer equation, difference schemes, kinetic approximation, DD-scheme, 2D transfer equation, 2D one-group stationary particles transfer equation, nonlinear edge scheme.

УДК 004.428

ПРОГРАММА АДАПТИВНОГО ВЫБОРА МЕТОДА РЕШЕНИЯ ПОТОКА СЛАУ В БИБЛИОТЕКЕ PMLP/LPARSOL

Ю. Г. Бартенев, В. А. Ерзунов

Российский федеральный ядерный центр –
Всероссийский НИИ экспериментальной физики, Саров

В докладе дается описание алгоритмов работы и параметров программы адаптивного выбора метода решения потока СЛАУ в библиотеке PMLP/LParsol. Исследования проведены для нескольких задач различных методик. Результаты счета задач с адаптивным выбором и без него показали, что он может дать существенное ускорение счета задач путем выбора наиболее оптимального метода решения СЛАУ на разных участках счета, отличающихся по сложности и требующих разные методы для решения. А также может освободить пользователя от необходимости подбора оптимального метода решения произвольной задачи и послужить для повышения надежности решения СЛАУ, возникающих в процессе компьютерного моделирования.

Ключевые слова: Система линейных алгебраических уравнений СЛАУ, разреженные матрицы, вычислительные системы с распределенной памятью, вычислительные системы, процессоры, адаптивный выбор метода решения СЛАУ.

Введение

Адаптивный выбор метода решения СЛАУ реализован в составе библиотеки PMLP/LParSol [1] для использования в программных комплексах ИТМФ. Он предназначен для ускорения счёта задач путем выбора оптимального с точки зрения быстродействия метода (среди заданных) решения СЛАУ на разных участках счёта, отличающихся по сложности и соответственно, требующих разные методы для решения. Он в значительной мере освобождает пользователя от необходимости подбора метода решения произвольной задачи и повышает надежность решения СЛАУ, возникающих в процессе счёта.

В ИТМФ РФЯЦ-ВНИИЭФ более 10 лет эксплуатировался предыдущий вариант механизма адаптивного выбора метода решения [2]. С появлением новых вычислительных систем, новых методик, новых методов решения СЛАУ появилась необходимость в совершенствовании механизма выбора метода решения потока СЛАУ. Новый вариант основан на других принципах оценки качества работы того или иного метода решения. В предыдущем варианте механизма адаптивного вы-

бора выигрыш в скорости решения осуществлялся за счёт решения на **одном** шаге по времени подряд идущих СЛАУ разными методами. При этом на каждом шаге по времени происходило, при необходимости, многократное построение различных методов решения. Однако, сейчас практически во всех задачах на шагах по времени не происходит изменение «портрета» (структуры) матрицы, а в некоторых – и на всей задаче. Поэтому предыдущий вариант вместо ускорения мог приводить к замедлению счёта и в последнее время не использовался. Использование же в задачах приёма понижения точности решения первой СЛАУ на шаге по времени, что сейчас активно используется (методика МИМОЗА [3]), в предыдущем варианте механизма адаптивного выбора было запрещено.

Описание алгоритма

Для адаптивного метода пользователь определяет набор предобусловливателей и их параметров, среди которых нужно будет выбрать наиболее быстрый на предстоящем участке счета задачи. Адаптивный метод после оценки предположительного времени решения СЛАУ разными методами на последнем шаге по времени (на основе ранее проведенных испытаний на предыдущих шагах по времени) и времени решения СЛАУ текущим методом на этом шаге определяет метод, который, предположительно, будет решать СЛАУ наиболее быстро на следующем шаге (чаще всего оказывается, что на следующем шаге следует использовать текущий метод, что дополнительно может минимизировать затраты на построение предобусловливателя). Для задания параметров метода решения потока СЛАУ в задаче используется структура ***parsol***, передаваемая в библиотеку PMLP/LParSol [1] для построения и решения заданным в ней методом. Для этого в процессе счёта задачи адаптивный метод решения СЛАУ меняет параметры в этой структуре и подготавливает новый выбранный метод (предобусловливатель) для продолжения счёта.

Можно выделить несколько этапов работы адаптивного выбора метода решения СЛАУ: чтение параметров из файла с параметрами и подготовка к работе, испытание назначенных к работе методов, выбор оптимального для дальнейшего счёта, обработка особой ситуации, то есть реакция на решение очередной СЛАУ с недостаточной точностью.

Файл с параметрами имеет имя «*adapt_preconds.pmlp*» и находится в папке, где решается задача.

Чтение файла с параметрами адаптивного выбора происходит только при инициализации библиотеки в задаче. Оно происходит в процессе, номер которого задан в файле параметров задачи. Затем, если чтение произошло без ошибок, параметры рассылаются на другие процессы задачи. Отсутствие файла с параметрами не является ошибкой, а говорит об отказе работать с адаптивным выбором метода решения.

После этого на каждом процессе происходит подготовка к работе (инициализация некоторых переменных и выделение памяти под рабочие массивы), и программа адаптивного метода становится готовой к работе, но её работа не начнётся, пока не наступят специальные события, заданные параметрами.

На каждом шаге по времени, начало которого обозначается в программе пользователя методом ***dls_nonlinearity_start***, испытывается только один предобусловливатель из заданного набора. Проведение испытаний – это замеры времени полного и/или частичного (на основе ранее выполненного полного) построения предобусловливателя, времени решения СЛАУ, запись числа итераций и количества СЛАУ на шаге. После решения каждой СЛАУ в ходе испытаний уточняются времена построения предобусловливателя в зависимости от признака неизменности «портрета» исходной и предобусловливающих матриц и времени, затраченного на одну итерацию (цена итерации). После завершения решения всех СЛАУ испытываемым методом на закончившемся шаге по времени для испытания выбирается следующий метод для использования на следующем шаге. Испытания заканчиваются, когда собрана информация обо всех допущенных к испытаниям методах (предобусловливателях). К дальнейшей работе не допускаются методы, заблокированные по результатам решения очередной СЛАУ. Например, произошла ошибка при построении предобусловливателя, или в результате работы решатель вернул код «*в ходе решения произошла нештатная ситуация*».

Если во время испытаний произошла нештатная ситуация (решатель вернул ненулевой код завершения), то испытываемый метод блокируется, испытания считаются законченными неудачно и начнутся заново с началом очередного шага. Если во время испытаний с каким-либо методом решатель вернул код *«заданная точность не достигнута»*, то этот метод блокируется не навсегда, а только на ближайшие испытания и до следующих, на которых он может быть реабилитирован. При возникновении такого события в решателе решение повторится (продолжится) с любым другим незаблокированным методом из списка. При коде завершения *«в ходе решения произошла нештатная ситуация»* решение текущей СЛАУ продолжится с нулевым начальным приближением.

Испытания начинаются либо сразу после чтения параметров и окончания подготовки, либо после того, как суммарное число итераций решателя СЛАУ на шаге по времени достигнет заданного в параметрах значения. Испытания также могут начаться без ожидания достижения указанного суммарного числа итераций, если программа адаптивного выбора находится в состоянии «готов к включению» и для метода, заданного в основных параметрах ***parsol***, решение СЛАУ завершилось с ненулевым кодом. Если во время испытаний оказались заблокированы (отбракованы) все указанные в списке методы, то работа этой программы прекращается, и восстанавливаются первоначальные параметры метода решения, заданные в структуре ***parsol***. После окончания испытаний программа адаптивного выбора приходит в состояние «*в работе*».

Следует отметить, что в отличие от предыдущего варианта адаптивного механизма, проведение испытаний почти не замедляют счёт задачи.

Для выбора метода решения перед началом очередного шага используются следующие данные, полученные в результате испытаний:

- число решёных СЛАУ на закончившемся шаге; (n_slaes)
- суммарное число итераций на закончившемся шаге; (*n_iters*)
- суммарное число итераций на предыдущем шаге; (*prev_iters*)
- время построения предобусловливателя с новым ‘портретом’ матрицы; (Tfull)
- время построения предобусловливателя со старым ‘портретом’ матрицы; (Tmini)
- суммарное количество итераций на шаге во время испытаний; test_iters)
- время одной итерации. (t_one_iter)

Результаты последних четырёх позиций получены во время ранее проведенных испытаний. Поэтому результат выбора зависит от того, насколько своевременно проведены испытания. То есть для точного сравнения методов испытания должны проходить на одном и том же шаге на одних и тех же СЛАУ. Но поскольку это накладно, то ***предполагаем***, что сложность решения СЛАУ на шагах во время испытаний одинакова или почти одинакова. Следующее ***допущение*** заключается в том, что на следующих шагах после испытания будут решаться СЛАУ такой же сложности, что и во время испытаний, или мало отличающиеся. Именно постепенное изменение сложности решения СЛАУ от шага к шагу в любую сторону приводит к тому, что в какой-то момент эти изменения накопятся для того, чтобы программа выбора решила сменить текущий метод на другой, более подходящий для данного этапа счета. Конечно, вышеописанные предположения и допущения на реальной задаче не всегда выполняются, поэтому в процессе счёта есть возможность коррекции результатов испытаний для текущего метода. При скачкообразном изменении сложности потока СЛАУ во время испытаний возможны промахи в выборе самого эффективного метода решения для продолжения счёта. Поэтому испытания должны проводиться периодически, как указано в параметрах, или внепланово, например, при резком изменении сложности решаемых СЛАУ (условие определяется параметрами), а также при задании изменения размера СЛАУ или приказа на перестроение адаптивного метода.

Таким образом, перед началом очередного шага по времени происходит вычисление предполагаемого времени (T_p см. ниже) решения каждым методом из списка незаблокированных, в т. ч. и текущего метода, как если бы эти методы (каждый в отдельности) использовались в решении СЛАУ прошедшего шага. А дальше осуществляется поиск предположительно минимального по времени метода решения СЛАУ и выдача его номера для дальнейшего построения и использования в счёте. Как уже отмечалось, чаще всего использованный на текущем шаге метод оказывается предположительно лучшим и на следующем шаге, до тех пор, пока сложность СЛАУ не изменится существенно.

Обозначим порядковый номер предобусловливателя, выбранного ранее, через j, а очередного испытываемого через i. Для каждого метода сначала вычисляется предполагаемое число итераций для завершившегося шага:

$$p\_iters[i] = test\_iters[i] / test\_iters[j]) * n\_iters. \quad (1)$$

Затем получается время проведения итераций методом i, как если бы он работал вместо метода j на последнем завершенном шаге:

$$T_p[i] = p\_iters[i] * t\_one\_iter[i], \quad (2)$$

к которому остаётся добавить времена построения предобусловливателя либо при полном построении с постоянно изменяющемся «портретом» матрицы

$$T_p[i] = T_p[i] + T_{full}[i] * n\_slaes , \quad (3)$$

либо при изменении «портрета» матрицы только в начале шага

$$T_p[i] = T_p[i] + T_{full}[i] + T_{mini}[i] * (n\_slaes – 1), \quad (4)$$

либо при неизменном ‘портрете’ матрицы на всей задаче

$$T_p[i] = T_p[i] + T_{mini}[i] * n\_slaes. \quad (5)$$

Если подсчёт предполагаемого времени в первых двух случаях (3) и (4) достаточно очевиден, то в последнем (5) случае есть некоторая *некорректность*, способная привести к ошибке при выборе наиболее оптимального предобусловливателя. В случае смены предобусловливателя все равно его придётся строить полностью, то есть абсолютно правильной будет предыдущая формула (4), но только на первом шаге, а в условиях потока шагов и плавного изменения сложности СЛАУ, в основном, будет более правильной последняя (5), которая и используется. Причём трудно угадать будет ли полное построение предобусловливателя на следующем шаге. Если же при неизменном ‘портрете’ матрицы уже произошла полная перестройка предобусловливателя, то на следующем шаге с большой вероятностью её не будет, что не исключает её появление на последующих шагах. Таким образом, остаётся неясным, нужно ли учитывать в этом случае время полного построения предобусловливателя для прогнозирования в выборе метода решения. А для ряда предобусловливателей, например, многосеточных, именно это время может быть основным в процессе счёта некоторых задач.

После вычисления предполагаемого времени решения на шаге для всех доступных методов выбор метода, который, предположительно, будет самым быстрым, как уже говорилось выше, тривиален.

Выбор оптимального метода решения СЛАУ происходит на каждом процессе и должен быть синхронизован. Из-за отличия собираемых времен счета на разных процессах из-за неоднородности ЭВМ и из-за неточности засечки времени пришлось вставить коллективную операцию синхронизации выбора номера предобусловливателя – *MPI_Bcast*, в которой ведущим является процесс, заданный в параметрах структуры ***parsol***. Это сделано, чтобы исключить возможность выбора разных методов решения на разных процессах, что недопустимо.

При работе с потоком СЛАУ с неизменным ‘портретом’ будет происходить полная перестройка работающего предобусловливателя при превышении числа итераций на закончившемся шаге значения, заданного в параметрах ***parsol***. В этом же режиме при завершении решения с кодом «*точность не достигнута*» текущий метод попадает под «*подозрение*» и полностью перестраивается. То есть он строится с признаком «*новый портрет матрицы*», и решение повторяется. Если же ситуация повторится, то метод блокируется до следующих испытаний и настраивается следующий, если же решение завершено успешно, то «*подозрение*» снимается.

Начало наступления очередного испытания ускорится, если разница числа итераций на текущем шаге (n_iters) и предыдущем (prev_iters) превысит заданное в параметрах число.

Так же в параметрах можно задать проведение испытаний на каждом шаге для текущего метода. Эта возможность позволяет минимизировать последствия ошибок на испытаниях из-за неравномерной сложности потока СЛАУ.

Наиболее затратная и неэффективная часть программы – это проведение испытаний на заданных методах решения, некоторые из которых могут оказаться неэффективными по времени построения или решения. Поэтому для минимизации потерь времени после выбора оптимального метода решения существует возможность заблокировать на несколько испытаний и для дальнейшего

участия в выборах методы, предполагаемое время решения СЛАУ которых намного больше оптимального. Это позволяет проводить испытания без потери эффективности, что в свою очередь повышает точность определения наиболее эффективного метода. При этом возможна ситуация, когда в процессе счёта число претендентов на роль оптимального может сократиться до одного.

При отключении программы адаптивного выбора (условие задаётся в параметрах) в решении остаётся последний метод, выбранный в качестве наиболее оптимального.

Файл с параметрами («*adapt_preconds.pmlp*»), находящийся в папке, где считается задача, состоит из информационных строк, начинающихся с символа ‘#’ и собственно строк с параметрами. Всего шесть строк.

Ниже приведён образец файла с параметрами:

```
# num_prec in-iter out-iter ignore st_recr print write blck  trace stat mini_tst  coeff_out rcr_prec
  5        10       0          1       200     1     1     1     1     0      1          2.5      0.6
#  parallel preconds
259    272     259  259    261
# inner_preconds
13     0       1       6       7
# ParT
0.001 0       0.01    0.0001 0.01
# dprecond.specify
1       0       0       0       10
# cgc_number
0       0       600     0       100
```

Первая строка задаёт размеры, условия включения/выключения программы, коэффициенты, влияющие на ход адаптивного выбора и отладочные признаки. В этой строке следующие тринадцать параметров:

1. (***num_prec***) Определяет число предобусловливателей для работы, заданных в этом файле. При задании нулевого значения программа выбора не включается в работу.

2. ***(in-iter)*** Задаёт суммарное число итераций решателя СЛАУ на очередном шаге, при превышении которого начинаются испытания заданных методов и включение программы адаптивного выбора. Нулевое значение означает начало испытаний сразу же с первого шага после чтения параметров. Рекомендуемый диапазон значений для этого параметра от 1 до 10. Бесполезно включать адаптивный выбор, если все СЛАУ решаются за минимальное число итераций.

3. (***out-iter***) Задаёт суммарное число итераций на шаге, меньше которого адаптивный выбор отключается до возникновения условий для включения. Нулевое значение означает, что адаптивный выбор не будет отключен до конца задачи.

4. (***ignore***) Задаёт число итераций при решении СЛАУ, при котором данная СЛАУ не учитывается ни при испытаниях, ни при работе программы. Это «*пустые*» СЛАУ, которые, возможно, и решать не пришлось. Такие СЛАУ иногда появляются в конце шага по времени уже с начальным приближением, подходящим в качестве решения. И для них нет ни времени построения, ни времени решения. Рекомендуемое значение для этого параметра = 1.

5. (***st_recr***) Задаёт число шагов, через которое будут происходить испытания методов, заданных в параметрах, за исключением методов временно заблокированных. Значение зависит от скорости изменения сложности потока СЛАУ. Например, в задаче из методики МЕДУЗА [4] этот параметр задавался равным 400, а в задаче из методики МРС [5] – от 3 до 30.

6. (***print***) Признак разрешения выдачи статистики по работе программы в файл, что происходит при каждом отключении работы программы и в конце задачи.

7. (***write***) Признак разрешения записи в информационный файл «*Adapt_search.log*» последовательности результатов работы программы выбора оптимального метода. Анализ этого файла дает информацию о выдаваемом задачей потоке СЛАУ и о качестве включённых в работу методов.

8. (***blck***) Параметр задаёт количество испытаний адаптивного механизма, на которое можно заблокировать какой-либо метод, показавший себя недостаточно эффективным для продолжения

счёта в текущий момент. Этот параметр позволяет немного сократить время проведения испытаний за счёт уменьшения числа участников. Рекомендуемое значение от 1 до 3.

9. (***trace***) Отладочный параметр задаёт включение трассировки работы отдельных частей программы. Служит для контроля правильности её работы и разбора нестандартных ситуаций разработчиками.

10. (***stat***) Параметр предназначен для управления сбором статистики.

11. (***mini_tst***) Значение этого параметра задаёт частоту проведения испытаний 'на лету' текущего метода решения. Например, значение, равное 1, означает, что на каждом шаге после решения первой СЛАУ будут обновлены значения времени построения предобусловливателя T_{full} или T_{mini} и цена одной итерации – ***t_one_iter***. При задании значения, равным 2, эти вычисления будут происходить на каждом втором шаге. Рекомендуемое значение 1 или 2.

12. (***coeff_out***) Коэффициент для включения временной отбраковки метода. Если после очередного выбора оптимального метода максимальное '*приведённое*' время какого-либо метода из списка окажется в ***coeff_out*** раз больше минимального времени, то этот метод на ***blck*** циклов испытаний блокируется для работы. Рекомендуемое значение этого параметра от 1,7 до 2,0.

13. (***rcr_prec***) Коэффициент для задания момента перестроения предобусловливателя, работающего с потоком СЛАУ на одном и том же 'портрете'. Полное перестроение происходит, когда суммарное время работы итерационной схемы начинает превышать время полного построения (как с новым 'портретом') предобусловливателя, умноженного на этот коэффициент. Рекомендуемое значение 0,6.

Остальные строки файла задают параметры предобусловливателей адаптивного выбора [1].

Для программы адаптивного выбора решателя задействованы только самые основные параметры. Остальные параметры для построения конкретного предобусловливателя содержатся в структуре ***parsol***.

Для возможности использования адаптивного метода решения в прикладной программе в начале каждого шага по времени необходимо вызвать функцию ***dls_nonlinearity_start(hDls)*** [1], которая даёт сигнал библиотеке о начале очередного шага по времени, поскольку только в этом методе производится анализ времён и числа сделанных итераций на предыдущем шаге для выбора оптимального метода решения на следующем. Возможен её вызов перед каждым решением СЛАУ (что, конечно, не рекомендуется), но это приведёт к тому, что описываемый метод выбора становится похожим на предыдущий вариант, за исключением этапа испытаний.

Поскольку анализ результатов счёта может производиться только после завершения счёта на очередном шаге, то это накладывает ограничение на использование программы адаптивного выбора. Она не может быть использована с гарантированным выбором оптимального метода на всех участках для задач, имеющих резко отличающийся по сложности решения поток СЛАУ. При этом выбор оптимального метода может быть осуществлён не точно, а с задержкой. Но для таких задач затруднительно подобрать единственный наилучший метод решения, даже используя для тестирования СЛАУ с разных участков счёта задачи. Наиболее выгодно использовать описываемую программу для задач с постепенными изменениями сложности решения потока СЛАУ.

Для нескольких разнотипных потоков СЛАУ, как, например, из программы ЛОГОС Тепло, имеется возможность задания разных параметров адаптивного метода для каждого типа СЛАУ при условии, что все потоки СЛАУ представлены отдельными объектами линейной системы.

Тестирование программы осуществлялось на наборах СЛАУ и задачах из разных методик ИТМФ РФЯЦ-ВНИИЭФ. В задачах демонстрируется работа механизма адаптивного выбора метода решения СЛАУ и приводится оценка быстродействия в расчётах с использованием и без использования этого механизма.

Результаты расчётов

Задача 1. При использовании для тестирования СЛАУ из методики МИМОЗА [3], записанных на разных по сложности участках, программа адаптивного выбора подтвердила, что наиболее эффективным методом на сложных участках является *AMG_mpi*. А в другом запуске метод

AMG_mpi на нескольких простых участках уступает блочному предобусловливателю Якоби с двумя итерациями *ILU0*(*JacobiBlock_mpi(ILU0)*) и использованием грубосеточного корректора.

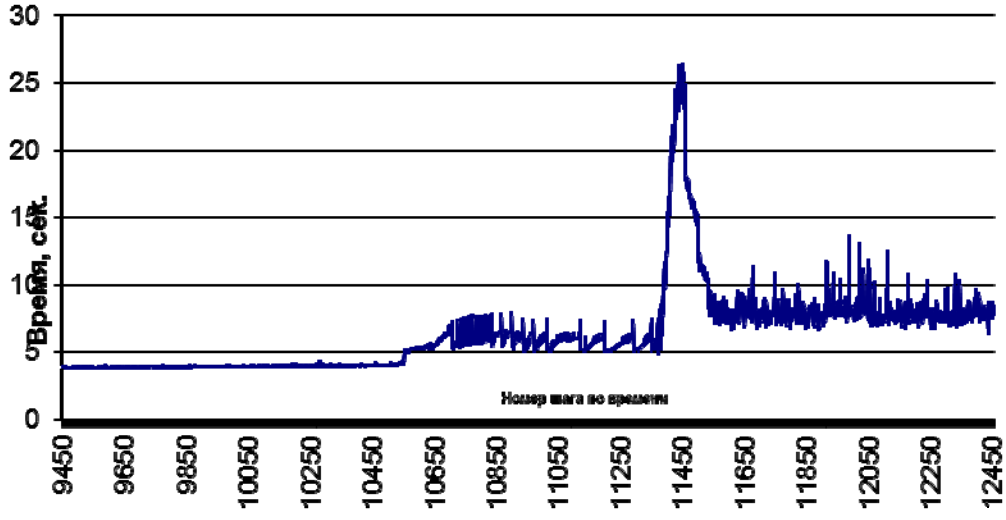

График зависимости времени решения СЛАУ на шаге по времени от номера шага

На рисунке показан график изменения сложности решения задачи. В качестве решателя задавался метод *BiCGStab* с ***DLS_ConvergenceResRhs***[*] критерием сходимости и точностью решения 1e-08. При решении первой СЛАУ на шаге по времени точность понижалась на три порядка.

При счёте задачи на сложных участках с включённым адаптивным выбором метод *AMG_mpi* не оставил другим методам никаких шансов. Программа уверенно выбирала его в качестве наиболее оптимального даже при достаточно частом включении испытаний для исключения ошибки. На первом простом этапе счёта задачи работал метод *ILU0*, затем метод *AMG_mpi* с кусочно-постоянным ‘портретом’ предобусловливающих матриц. Без адаптивной подстройки, например, при использовании метода *AMG_mpi* без использования известного ‘портрета’, время решения СЛАУ значительно возросло бы, судя по пикам.

На завершающем этапе счёта задачи с теми же параметрами наблюдалось отключение программы на нескольких участках на 200 – 400 шагах по времени (примерно 1000 СЛАУ) в соответствии с заданными параметрами.

На первых шагах задачи целесообразно включить в состав адаптивного метода самые простые методы Якоби (*Jacobi_mpi*) или *ILUp3_mpi* (*ILU0* трехдиагональной части матрицы), сокращающих примерно на 2 % время решения СЛАУ.

Задача 2. Для исследования поведения программы адаптивного выбора метода решения на задаче из двумерной методики МЕДУЗА [4] задавались методы:

1. *Schwartz_mpi (ILUt)*;
2. *JacobiBlock_mpi (ILUt)* с грубосеточным корректором;
3. *AMG_mpi*;
4. *Schwartz_mpi (ILUt) с* грубосеточным корректором.

Адаптивный метод включался после достижения на шаге по времени суммарного числа итераций больше четырёх. Блокировка методов была отключена, игнорировались результаты решения СЛАУ с числом итераций меньше двух, испытания происходили каждые 400 шагов.

Запуски производились на 64-х процессах. В качестве решателя использовался BiCGStab с ***DLS_ConvergenceResRhs*** критерием сходимости, описанным выше, и точностью решения 1e-06. Для оценки эффективности методов использовалось число шагов по времени, выполненных задачей в течение одного часа.

[*] Критерий сходимости ***DLS_ConvergenceResRhs*** ($||A*x^{(k)} - b|| \le atol+rtol*||b||$) приведён в [1].

Применение первого, второго и четвёртого методов из приведённого выше списка без использования программы адаптивного выбора приводило к тому, что решение останавливалось из-за того, что даже при достижении максимального числа итераций заданная точность не была получена. Были опробованы и другие методы, но они либо не справлялись, либо работали слишком медленно.

Предобусловливатель *AMG_mpi* без использования программы адаптивного выбора решил все СЛАУ задачи, показав следующие результаты за час счёта:

– число шагов по времени = **2479**;

– доля времени, занятого решателем = **40 %**.

С использованием программы адаптивного выбора метода решения СЛАУ были получены следующие результаты:

– число шагов по времени = **3186**;

– доля времени, занятого решателем = **30 %**.

Таким образом, использование программы адаптивного выбора метода решения на этой задаче привело к повышению надёжности и скорости счёта. Задача досчитана до конца с использованием адаптивного механизма. Ускорение от использования адаптивности составило **1,28** раз.

Задача 3. На задаче из методики МИМОЗА [3] использовался решатель BiCGStab с ***DLS_ConvergenceResRhs*** критерием сходимости с заданной точностью решения СЛАУ 1e-08. При решении первой СЛАУ на шаге по времени точность понижалась на два порядка. Время запуска равнялось 30 минутам.

Для исследования поведения программы задавалось 5 методов:

1. *ILUp3_mpi*;
2. *JacobiBlock_mpi(ILUt);*
3. *AMG_mpi;*
4. *Schwartz_mpi (ILUt);*
5. *Schwartz_omp(ILUt)*.

Адаптивный метод включался после достижения на шаге по времени суммарного числа итераций больше четырёх. Испытания происходили каждые 200 шагов.

На первом этапе счёта наилучшие времена показывал метод 5), затем 2). Остальные участвовали только в испытаниях. Можно было их исключить из списка параметров для адаптивного выбора, но неизвестно было, как поведёт себя задача на других участках. За 30 минут в таком режиме задача сделала 1223 шага по времени, доля времени решателей составила 23,6 %.

Задача 4. На другой задаче из методики МИМОЗА [3] для решения в составе адаптивного выбора метода решения потока СЛАУ использовались следующие методы:

1. *Jacobi_mpi*;
2. *JacobiBlock_mpi(ILUt);*
3. *AMG_mpi*;
4. *BoomerAMG_mpi.*

Результаты работы методов в составе программы адаптивного выбора метода решения СЛАУ показаны в табл. 1.

Таблица 1

Результаты работы по методам на примере задачи МИМОЗА [3]

Метод решения	Число решённых СЛАУ	Время решения, сек.	Доля времени работы, %
Jacobi_mpi	28248	455	5
JacobiBlock_mpi	8573	542	5
AMG_mpi	3249	676	7
BoomerAMG_mpi	32639	8207	83

За 7,5 часов счета сосчитаны десятки тысяч шагов по времени, доля времени решателей составила 36 %.

Задача 5. Запуски на задаче МРС[5] решатели СЛАУ работали на 192-х процессах с использованием потоков.

Для этой задачи задавался решатель – *CG* с описанным выше критерием сходимости и точностью 1e-08. Максимальное число итераций равно 400. Время запуска равнялось 3 часам. Число шагов по времени задавалось 240.

Для исследования поведения программы задавалось 4 метода:

1. *ILUp3_mpi*;
2. *BlockJacobi_mpi(c Schwartz_omp(IC0, ...));*
3. *Schwartz_mpi(c Schwartz_omp(IC0, ...));*
4. *Schwartz_omp(IC0)*;

Адаптивный метод включался после достижения на шаге по времени суммарного числа итераций больше двух. Результаты решения взяты из статистики библиотеки из файла '*pmlp_dls.log'*:

Общее время работы решателей = 3924.7 сек.
Время на итерациях = 3482,4 сек.
Общее число итераций = 26268
Общее число решённых СЛАУ = 1035
Доля времени, занятая решателями = 51,5 %

Результаты работы методов в составе программы адаптивного выбора метода решения СЛАУ показаны в табл. 2.

Таблица 2

Результаты работы по методам на примере задачи МРС [5]

Метод решения	Число решённых СЛАУ	Время решения	Доля времени работы, %
ILUp3_mpi	88	457,2	13,13
BlockJacobi_mpi	496	1573,3	45,18
Schwartz_mpi	76	344,2	09,88
Schwartz_omp	375	1107,7	31,81

Из таблицы видно, что в процессе решения СЛАУ принимали участие всего лишь два метода: *BlockJacobi_mpi* и *Schwartz_mpi*. Остальные принимали участие только в испытаниях. Включение их в состав адаптивного набора обусловлено тем, что неизвестно какие методы окажутся наиболее эффективными для генерируемого задачей потока СЛАУ. Не исключено, что на других участках задачи они будут выбраны как наиболее подходящие для счёта.

При использовании отдельно без механизма адаптивного выбора этих методов очевидно, что потребуется больше времени для решения такого же количества СЛАУ, если вообще возможно их решение с заданной точностью. Например, на том же участке задачи до внедрения механизма адаптивного выбора задача считалась только с блочным предобусловливателем Якоби.

Заключение

Программа адаптивного выбора решателя была опробована в методиках МЕДУЗА [4], МИМОЗА [3], МРС [5], и ЛОГОС Тепло [6]. Результаты счёта задач с адаптивным выбором и без него показали, что он может дать ускорение счёта задач путем выбора наиболее оптимального метода решения СЛАУ на разных участках счёта, отличающихся по сложности и требующих разные методы для решения. А также может освободить пользователя от необходимости подбора оптимального метода решения произвольной задачи и послужить для повышения надежности решения СЛАУ, возникающих в процессе счёта.

Литература

1. Алейников А. Ю., Барабанов Р. А., Бартенев Ю. Г., Ерзунов В. А., Карпов А. П., Кузнецов В. Ю., Петров Д. А., Резчиков В. Ю., Стаканов А. Н., Щаникова Е. Б. Применение параллельных решателей СЛАУ в пакетах программ инженерного анализа РФЯЦ-ВНИИЭФ // XVI Международная конференция «Супервычисления и математическое моделирование»: сб. науч. тр. / под ред. Р.М. Шагалиева. – Саров: ФГУП «РФЯЦ-ВНИИЭФ», 2017. – С. 28 – 36.

2. Ерзунов В. А., Горбунов А. А. Механизм адаптивного выбора решателя в библиотеке LParSol // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. – 2009. – Вып. 1 – С.55 – 62.

3. Софронов И. Д., Винокуров О. А., Змушко В. В., Плетенев Ф. А., Сараев В. А. Комплекс программ МИМОЗА // Вопросы математического моделирования, вычислительной математики, информатики. – М. – Арзамас-16, 1994. – С. 94 – 96.

4. Горбунов А. А. Метод решения уравнения теплопроводности на нерегулярных сетках в параллельном режиме в методике МЕДУЗА // Вопросы атомной науки и техники». Серия: Математическое моделирование физических процессов. – 2008. – Вып. 3. – С. 32 – 46.

5. Дерюгин Ю. Н., Полищук С. Н., Тихомиров Б. П. Расчет лучистой теплопроводности в методике МРС с использованием неточных методов Ньютона // XV Международная конференция «Супервычисления и математическое моделирование»: сб. науч. тр. / под ред. Р.М. Шагалиева. – Саров: ФГУП «РФЯЦ-ВНИИЭФ», 2015. – 198 – 205.

6. Вишняков А. Ю., Дерюгин Ю. Н., Глазунов В. А., Чистякова И. Н. Пакет программ ЛОГОС. Модуль расчета сопряженных и связанных задач теплопереноса // XIV Международная конференция «Супервычисления и математическое моделирование»: сб. науч. тр. / под ред. Р. М. Шагалиева. – Саров: ФГУП «РФЯЦ-ВНИИЭФ», 2013. – С. 154 – 161.

AN ADAPTIVE PROGRAM TO CHOOSE A SOLVING METHOD FOR SLAE FLOW IN THE PMLP/PARSOL LIBRARY

Yu. G. Bartenev, V. A. Erzunov

Russian Federal Nuclear Center –
All-Russian Research Institute of Experimental Physics, Sarov

The paper describes the operation algorithms and the parameters of the adaptive selection program of the SLAE – flow solution method in PMLP/LParsol library. The research has been carried out for several problems of different methods. The computation results on the problems with adaptive selection and without it have shown that it can provide considerable speedup in the computations by selecting the most optimum method to solve SLAE at different sections of computations that differ in complexity and demand different methods of solution. It can also free the user from the need to select and optimum solution method of a random problem and serve in upgrading the reliability of SLAE solution in the process of computer simulation.

Key words: a system of linear algebraic equations SLAE, sparse matrices, computational systems with distributed memory, computational systems, processors, adaptive selection of a method to solve SLAE.