МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М. В. ЛОМОНОСОВА

Научно-исследовательский вычислительный центр

О. Б. Арушанян, Н. И. Волченскова

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ НА РАСПРЕДЕЛЕННОЙ ПАМЯТИ НА ОСНОВЕ ПАКЕТА ScaLAPACK

Учебное пособие

Москва, 2014

О.Б. Арушанян, Н.И. Волченскова

Решение систем линейных алгебраических уравнений на распределенной памяти на основе пакета ScaLAPACK.

(Учебное пособие)

ОГЛАВЛЕНИЕ

1. Введение.

Уровень и сложность решаемых в настоящее время научных и производственных задач таков, что, как правило, требуется использование высокопроизводительных суперкомпьютеров, в частности с распределенной памятью. Подготовка специалистов в этой области в МГУ им. М.В. Ломоносова ведется на базе Суперкомпьютерного центра (СКЦ МГУ), состоящего в настоящее время из двух высокопроизводительных суперкомпьютеров: СКИФ МГУ "Чебышев" (60 Тфлоп/с) и "Ломоносов" (1,7 Пфлоп/с).

Описываемый в настоящем учебном пособии комплекс программ был реализован с использованием суперкомпьютера СКИФ МГУ "Чебышев".

Большое число численных задач включает в себя или сводится к решению систем линейных алгебраических уравнений большого размера (далее систем уравнений или просто линейных систем). Математическая запись такой задачи имеет вид

$$\mathrm{A} * \mathrm{X} = \mathrm{B}\,,$$

где

A — невырожденная квадратная матрица,

B — вектор правой части системы,

X — вектор неизвестных.

Высокоэффективное (в смысле затрат машинного времени и оперативной памяти) решение таких систем может быть выполнено с использованием специализированных пакетов программ, разработанных высококвалифицированными специалистами в этой области. Одним из таких наиболее широко известных пакетов является пакет ScaLAPACK (Scalable Linear Algebra PACKage) [13, 14], позволяющий решать линейные системы на распределенной памяти с использованием технологии MPI (Massage Passing Interface) [1, 2].

В НИВЦ МГУ создан русскоязычный аналог этого пакета, получивший название "Комплекс параллельных программ PARALG". В его основе лежат те же принципы реализации параллельных алгоритмов, которые приняты в ScaLAPACKe, и при реализации целевых программ комплекса PARALG использовались базовые модули пакета ScaLAPACK.

В комплексе PARALG содержатся и задокументированы (т.е. снабжены инструкциями по использованию) именно готовые целевые программы решения задач линейной алгебры (прежде всего для решения систем линейных алгебраических уравнений). Комплекс содержит большее количество целевых программ, чем в пакете ScaLAPACK; полный их список приводится в п. 3.5 настоящего

пособия. Базовые же модули (как составные части целевых программ) вызываются непосредственно из библиотеки ScaLAPACKa.

Общие правила и принципы, приводимые в теоретической части предлагаемого учебного пособия (п. 2), в одинаковой мере справедливы как для пакета ScaLAPACK, так и для комплекса PARALG. Использование готовых целевых программ комплекса PARALG значительно облегчает эффективное решение указанных задач.

Тем не менее, у неспециалистов в данной области (как показывает опыт) это может вызвать определенные трудности, так как требует знакомства с недостаточно известными действиями и понятиями пакета ScaLAPACK, связанными с обеспечением эффективности параллельных вычислений. Сюда относятся такие, например, особенности параллельной реализации алгоритмов как:

— организация параллельных процессов в так называемую (виртуальную) решетку процессов и сопоставление с выбранной решеткой определенного численного значения параметра, который однозначно характеризует выбранную решетку и называется указателем контекста или просто контекстом;

— специальное распределение блоков матриц по параллельным процессам решетки;

— понятие и правильное определение так называемых дескрипторов матриц и связанное с этим представление о локальных и глобальных параметрах целевых программ комплекса PARALG.

Кроме того, требуется овладеть правильным использованием подпрограмм пакета BLACS (Basic Linear Algebra Communication Subprograms) [18, 19], входящего в состав пакета ScaLAPACK и выполняющего на более высоком уровне функции примитивов MPI. Указанные особенности обсуждаются в п. 2 данного учебного пособия. Кроме того, в п. 2 приводится краткое описание математических аспектов решения систем: перечисляются типы матриц линейных систем, которые охватываются комплексом PARLAG, а также указываются используемые методы при решении систем с разными типами матриц.

В п. 3 разбираются примеры программирования различных случаев при решении систем линейных алгебраических уравнений с помощью программ комплекса PARALG. Кроме того, описывается запуск соответствующих задач в ОС Linux на суперкомпьютере СКИФ МГУ “Чебышев”.

2. Основные правила решения линейных систем с помощью комплекса PARALG.

2.1. Общее описание организации и структуры комплекса.

Комплекс программ PARALG относится к разряду "параллельных предметных библиотек", при обращении к которым пользователю не приходится явно применять какие-либо конструкции специальных языков параллельного программирования или интерфейсов, поддерживающих взаимодействие параллельных процессов. Все необходимые действия по совместной работе параллельных процессов при решении задач линейной алгебры выполняются подпрограммами из пакетов PBLAS (Parallel Basic Linear Algebra Subprograms) [15, 16] и BLACS, являющихся составными частями пакета ScaLAPACK.

Пакет BLACS является набором подпрограмм, предназначенных для использования методов распараллеливания при решении задач линейной алгебры. Этот пакет разработан для компьютеров с распределенной памятью и обеспечивает запуск параллельных процессов и их взаимодействие посредством обмена сообщениями. При этом BLACS освобождает пользователей от изучения предназначенных для этих целей комплексов стандартных примитивов, включенных в MPI.

Работа подпрограмм пакета BLACS опирается на представление о так называемой "решетке процессов" (process grid).

Предполагается, что параллельные процессы организованы в двумерную (или одномерную) решетку процессов, в которой каждый из процессов идентифицируется своими координатами в решетке и хранит определенные части обрабатываемых матриц и векторов. Для работы с параллельными процессами в состав пакета BLACS включены подпрограммы, реализующие следующие основные функции:

- построение, изменение и получение сведений о решетке процессов;
- обмен сообщениями (передача матриц или их частей) между двумя процессами;
- передача сообщений от одного процесса сразу многим процессам;
- выполнение необходимых итоговых действий после получения параллельными процессами частичных результатов (например, вычисление итоговых сумм или максимумов или минимумов по всем процессам).

Другим пакетом, к которому обращаются программы комплекса PARALG, является пакет PBLAS, содержащий версии для распределенной памяти подпрограмм (первого, второго и третьего уровней) BLAS [26, 27]. В этом пакете

содержатся версии подпрограмм, реализующих базовые операции линейной алгебры (например, действия над векторами и скалярами, умножение матрицы на вектор или перемножение двух матриц и др.).

Разработчики пакета PBLAS реализовали его таким образом, чтобы интерфейс его подпрограмм был максимально похожим на интерфейс подпрограмм пакета BLAS.

Таким образом, целевые программы описываемого комплекса PARALG (т.е. программы, которые имеют самостоятельное значение при решении прикладных задач) могут содержать обращения к подпрограммам указанных выше пакетов BLACS, PBLAS и BLAS, а также к базовым и вспомогательным подпрограммам пакета ScaLAPACK.

К базовым подпрограммам пакета ScaLAPACK относятся подпрограммы, которые не используются независимо от других программ комплекса PARALG. Они всегда выступают в роли вызываемых из других подпрограмм. Например, подпрограмма умножения матрицы общего вида на ортогональную матрицу, полученную в результате QR-разложения матрицы другой подпрограммой.

К вспомогательным подпрограммам относятся подпрограммы, называемые tool routines. Эти подпрограммы выполняют различные вспомогательные функции, например:

- выдача параметров машинной арифметики конкретной ЭВМ;
- выдача диагностических сообщений в стандартной форме;
- подсчет количества строк или столбцов распределенной по параллельным процессам матрицы, расположенных на одном из этих процессов и др.

Целевые программы комплекса PARALG решают задачи из следующих разделов линейной алгебры: решение систем линейных алгебраических уравнений, решение проблемы собственных значений (линейной и обобщенной) и сингулярное разложение матриц. Настоящее пособие предназначено для обучения практическим навыкам использования программ комплекса, предназначенных для решения систем линейных алгебраических уравнений.

2.2. Применение пакета BLACS для организации параллельных процессов.

Подпрограммы пакета BLACS разработаны специально для обеспечения эффективного решения на параллельных компьютерах с распределенной памятью задач линейной алгебры, в которых основным обрабатываемым объектом является матрица.

При распределении таких двумерных массивов по параллельным процессам удобнее мысленно организовать параллельные процессы в виде логической

двумерной решетки процессов. Тогда, например, все части строки распределенной матрицы могут быть найдены в процессах, относящихся к строке решетки процессов, а части столбца — в процессах, относящихся к столбцу решетки процессов.

В принятой в пакете BLACS логической структуре параллельных процессов каждый процесс идентифицируется с помощью пары чисел — координат процесса в решетке процессов.

На рисунке ниже изображена решетка из 8 процессов, состоящая из двух строк и четырех столбцов: 2×4.

	0	1	2	3
0				
1				

Этим процессам ставятся в соответствие координаты решетки: $(0,0)$, $(0,1)$, $(0,2)$, $(0,3)$, $(1,0)$, $(1,1)$, $(1,2)$, $(1,3)$.

В общем случае, если решетка процессов имеет R строк и C столбцов, то любой процесс в решетке обозначается парой $(i,\ j\ )$, где $0 \leqslant i <$ R, $0 \leqslant j <$ C. Общее число процессов равно $\mathrm{R} \times \mathrm{C}$.

В компьютерах с распределенной памятью каждый параллельный процесс работает со своей локальной памятью, а обмен информацией между процессами осуществляется посредством передачи сообщений через связывающую их сеть.

Пакет BLACS, предназначенный для таких компьютеров, опирается на использование широко известных и распространенных интерфейсов обмена сообщениями MPI (Massage Passing Interface) или PVM (Parallel Virtual Machine). Они содержат не только операции обмена между одним посылающим и одним принимающим сообщение процессом, но также и массовые операции обмена, когда, например, один процесс может рассылать сообщение сразу всем другим параллельным процессам, и, наоборот, один из процессов может получать сообщения от всех других.

В пакете BLACS такие операции называются контекстными операциями (scope operations), а про процессы, принимающие в них участие, говорят, что они находятся внутри операционного контекста (operation's scope). Организация процессов в виде логической решетки, позволяет пакету BLACS естественным образом расширить виды контекстов, а именно, объявлять (задавать) в качестве контекста, т.е. области действия массовых операций обмена сообщениями, не только все параллельные процессы, но также и все процессы, принадлежащие только некоторой строке решетки процессов или некоторому ее столбцу. Это обеспечивает программисту более удобные и гибкие возможности распараллеливания.

Использование понятия решеток процессов удобно не только для программирования, но и позволяет повысить эффективность (скорость) вычислительного процесса. Однако такое возможно только в том случае, если в базовой машине параллельные процессы связаны между собой с помощью, по крайней мере, двумерных сетей.

В большинстве современных суперкомпьютеров это условие выполнено, но при использовании сети Ethernet такое преимущество недостижимо.

2.3. Задание параметров при обращении к целевым программам комплекса.

2.3.1. Глобальные и локальные объекты и параметры параллельных программ.

Программы комплекса PARALG составлены на языке ФОРТРАН-77 в стиле SPMD (Single Program Multiple Data). Это означает, что одна и та же исполняемая программа загружается во все параллельные процессы, заказанные для решения задачи в команде запуска на счет. В то же время, каждый процесс занимается обработкой своих данных, составляющих некоторую часть общих данных задачи. Эти данные отдельного процесса (их называют локальными данными) непосредственно недоступны другим процессам и хранятся в отдельной (локальной) памяти каждого процесса. Необходимый обмен данными между разными процессами производится только посредством передачи сообщений (технология MPI [1, 2, 8–11]).

Из сказанного следует, что исходную матрицу A не требуется хранить целиком в оперативной памяти процессора как единый массив. На практике такие матрицы (векторы) в виде единого целого могут храниться во внешних файлах (на дисковой памяти). По этой причине такую матрицу A называют **глобальной** матрицей.

Поскольку перед началом работы целевых параллельных программ комплекса исходные глобальные матрицы (или векторы) разделяются на части и распределяются по параллельным процессам, то в дальнейшем нередко употребляется термин “распределенная глобальная матрица (вектор)”. Часть глобальной матрицы (вектора), представленная в локальной памяти одного из параллельных процессов, называется **локальной** частью матрицы (вектора). Совокупность всех локальных частей на всех используемых при решении задачи параллельных процессах и составляет **распределенную глобальную** матрицу (вектор).

Поскольку каждый процесс работает со своей локальной частью данных (в нашем случае фрагментом глобальной матрицы), то в головной фортранной программе достаточно выделить (заказать) память, необходимую только для максимальной локальной части этой матрицы из всех локальных частей,

обрабатываемых на параллельных процессах.

Локальные части, обрабатываемые на каждом из процессов, могут быть как одинакового, так и разного размера (см. подробнее пп. 2.4, 3.2). Для того чтобы каждый из процессов мог разместить свою локальную часть данных (матрицы), заказываемая область локальной памяти должна быть не меньше максимальной локальной части данных.

Таким образом, в головной фортранной программе, обращающейся к целевой программе комплекса, не требуется резервировать память для всего глобального объекта (матрицы или вектора) целиком, а только для их локальных частей.

Целиком эти объекты существуют, как правило, во внешних файлах. Для сохранения информации о конкретном способе разбиения глобального объекта на блоки и размещении блоков в локальной памяти всех процессов вводится объект, называемый **дескриптором** глобального массива. Дескриптор представляет собой одномерный массив, состоящий из 9 или 7 элементов (в зависимости от типа дескриптора). В дескрипторе хранится информация о размерах глобального массива (число строк и столбцов матрицы), о размерах блоков, на которые она разбивается, о номере процесса, в память которого распределяется первый блок глобальной матрицы (левый верхний угол), а также некоторая другая информация. Подробнее о дескрипторах см. в п. 2.3.2.

Учитывая общие для комплекса правила распределения блоков матриц (векторов) по параллельным процессам, по информации в дескрипторах всегда можно определить, в локальной памяти какого процесса и где именно находится элемент глобальной матрицы с глобальными индексами i и j; подробнее см. в п. 2.4.1.

Непосредственное размещение исходных данных в локальной памяти каждого процесса должен выполнять сам пользователь (в головной программе перед обращением к целевой программе комплекса) в соответствии с установленными правилами для матриц разных видов: плотных общего вида или симметричных, ленточных или трехдиагональных (см. п. 3.2).

Организация решетки процессов, обмен информацией между процессами, а также другие действия, связанные с процессами, выполняются с помощью подпрограмм пакета BLACS. Пользователю, однако, в большинстве случаев не требуется знание всех возможностей этого пакета. При решении своей задачи пользователь должен записать в головной программе лишь несколько стандартных обращений к нескольким подпрограммам пакета BLACS.

Наглядные примеры этого можно найти, например, в предлагаемом пользователям описании одной из целевых параллельных подпрограмм комплекса (см. Приложение 1), а также в фортранном тексте тестового примера к этой

подпрограмме (см. Приложение 2). Более подробно описание решеток процессов и правил работы с ними приводится в пп. 2.4, 2.3.2, 3.1.

Из сказанного выше следует, что формальные параметры целевых программ комплекса могут быть глобальными или локальными.

Глобальными называются параметры подпрограмм, которые относятся к указанным выше глобальным объектам. Например, глобальными являются число строк или столбцов исходной матрицы A(M, N) или число наддиагоналей (BWU) в матрице A, если она является ленточной.

Локальными называются параметры, которые характеризуют объекты, определяемые в локальной памяти отдельного процесса. Например, указатель на локальную память, занимаемую локальной частью глобальной матрицы A или глобального вектора B правых частей системы уравнений.

Упоминавшиеся выше дескрипторы массивов характеризуются одновременно и как глобальные, и как локальные параметры подпрограмм, поскольку среди их элементов, определяющих в основном характеристики глобальных массивов, имеется характеристика, описывающая размещение части глобального массива в локальной памяти процесса (ведущая размерность локального массива).

2.3.2. Дескрипторы глобальных массивов.

Как уже упоминалось выше, для каждого глобального массива (матрицы или вектора), который необходимо разбить на блоки (с последующим размещением каждого из них в локальной памяти параллельных процессов, участвующих в решении задачи), вводится специальный объект, называемый **дескриптором** массива.

Дескриптор представляет собой одномерный массив, состоящий из 9 или 7 элементов целого типа (в смысле языка ФОРТРАН). Этот массив предназначен для хранения информации, конкретизирующей способ разбиения глобального массива на блоки и их распределение по всем параллельным процессам.

Дескрипторы являются обязательными параметрами целевых программ комплекса.

Дескрипторы принято обозначать идентификатором DESC с добавлением в конце имени массива (матрицы или вектора). Например, DESCA означает дескриптор матрицы A, DESCB — дескриптор вектора B.

Дескрипторы бывают трех типов, которые обозначаются целыми числами: 1, 501, 502.

2.3.2.1. Дескрипторы плотных матриц. Для плотных глобальных матриц A, которые распределяются по двумерной решетке процессов, используется дескриптор, имеющий тип 1 и состоящий из 9 элементов, за каждым из которых

закреплено свое символическое имя (см. таблицу ниже).

В таблице указываются номер (индекс) каждого элемента в массиве, его символическое имя (которое оканчивается символом подчеркивания "_"), а также его смысл. Идентификатор, стоящий в символическом имени элемента после символа подчеркивания "_", является идентификатором глобального массива, описываемого данным дескриптором. Например, N_A обозначает элемент дескриптора, который содержит число столбцов глобальной матрицы A.

Таблица элементов дескриптора плотной глобальной матрицы A

N	**Имя**		**Смысл**
1.	DTYPE_A	—	тип дескриптора (для плотной матрицы DTYPE_A=1)
2.	CTXT_A	—	обозначение контекста BLACS'a, соответствующего выбранной решетке процессов, по которой распределяется глобальная матрица A; целое значение, устанавливаемое подпрограммой из пакета BLACS; см. п. 3.1
3.	M_A	—	число строк в глобальной матрице A
4.	N_A	—	число столбцов в глобальной матрице A
5.	MB_A	—	число строк в блоках, на которые разбивается глобальная матрица A
6.	NB_A	—	число столбцов в блоках, на которые разбивается глобальная матрица A
7.	RSRC_A	—	номер строки процесса в решетке процессов, куда был распределен первый элемент глобальной матрицы A (левый верхний угол); как правило, удобнее всего распределять первый элемент в процесс с номером $(0,0)$
8.	CSRC_A	—	номер столбца процесса в решетке процессов, куда был распределен первый элемент глобальной матрицы A (левый верхний угол); как правило удобнее всего распределять первый элемент в процесс с номером $(0,0)$
9.	LLD_A	—	ведущая размерность локального массива (LLD_A$\geqslant$MAX$(1,$LOCr(M_A)), см. пояснения ниже

В этой таблице и далее число строк локального массива обозначается LOCr(M_A). Записи DESCA(3) и DESCA(M_A) означают один и тот же элемент дескриптора матрицы A, содержащий число строк этой матрицы.

2.3.2.2. Дескрипторы вектор-столбцов правых частей систем уравнений.

В предыдущем пункте (п. 2.3.2.1) описывалось, как должен быть сформирован дескриптор глобальной плотной матрицы A.

Рассмотрим теперь правила разбиения и распределения по процессам глобального вектора (или матрицы) правых частей B. Прежде всего, этот вектор (матрица) всегда распределяется по той же самой (уже заявленной пользователем) решетке процессов, что и глобальная матрица A. Поскольку матрица A плотная и ее распределение описывается дескриптором типа 1, то распределение вектора (матрицы) B также должно быть описано дескриптором типа 1.

N	Имя		Смысл
1.	DTYPE_B	—	тип дескриптора для вектора (матрицы) правых частей B (соответствующего ленточной или трехдиагональной матрице A), распределенного по решетке процессов блочно-строчным образом, DTYPE_B=502
2.	CTXT_B	—	обозначение контекста BLACS'a, соответствующее выбранной решетке процессов, по которой распределяется глобальный вектор (матрица) B; целое значение, устанавливаемое подпрограммой из пакета BLACS (см. п. 3.4.).
3.	M_B	—	число строк глобального вектора (матрицы) B
4.	MB_B	—	число строк в блоках, на которые разбивается глобальный вектор (матрица) B
5.	CSRC_B	—	номер столбца процесса в решетке процессов, куда была распределена первая строка глобального вектора (матрицы) B
6.	LLD_B	—	ведущая размерность локального массива: LLD_B$\geqslant$MAX(1, LOCr(M_B)); для систем с трехдиагональными матрицами игнорируется (положить равным 0)
7.		—	в текущей версии комплекса не используется (положить равным 0)

Вектор B представляется как матрица вида (M, 1), имеющая M строк и один столбец. Этот вектор-столбец B распределяется по процессам первого столбца двумерной решетки процессов. Процессы из всех остальных столбцов решет-

ки процессов в случае, если B — вектор, не содержат никаких его элементов. Подробнее см. п. 2.4.2; пример приведен в Приложении 1.

Для случая ленточных и трехдиагональных глобальных матриц A, блочно-столбцовое распределение которых по процессам описывается с помощью дескрипторов типа 501 (см. п. 2.3.2.3.), распределение глобального вектора (матрицы) правых частей B должно быть описано с помощью дескриптора типа 502. Этот дескриптор, так же как и дескриптор типа 501, состоит из 7 элементов, имеющих аналогичные закрепленные символические имена. Выше приведена таблица для элементов дескриптора типа 502.

В отличие от глобальной ленточной матрицы A, которая подвергается блочно-столбцовому распределению (описываемому дескриптором типа 501), глобальный вектор (матрица) B подвергается так называемому блочно-строчному распределению. Это означает, что вектор (матрица) B разбивается на блоки по строкам (каждая строка целиком попадает в локальную память одного из процессов). Если B — вектор, то строка вырождается в один элемент, т.е. блокируются несколько соседних элементов вектора B. Подробнее см. в п. 3.4.

Конкретный пример распределения вектора B по процессам можно найти в разделе “Пример использования” описания одной из подпрограмм комплекса для решения систем с ленточной матрицей A

(см. http://num-anal.srcc.msu.ru/par_prog/cat5631.htm).

Инициализацию дескрипторов глобальных матриц A и B необходимо сделать до обращения к подпрограммам комплекса (см. п.3.1.).

2.3.2.3. Дескрипторы ленточных и трехдиагональных матриц.

Для ленточных и трехдиагональных глобальных матриц A, которые распределяются по решетке процессов вида $1 \times P$ (см. пп. 2.4.3 и 2.4.4), используются дескрипторы, имеющие тип, который обозначается целым числом 501. Решетка процессов вида $1 \times P$ означает частный случай двумерной решетки, которая состоит из одной строки и из P столбцов. По существу это — одномерная решетка процессов, однако каждый процесс в ней все равно идентифицируется парой чисел, в которой первое число равно 0. Например, процесс $(0, K)$ означает процесс, находящийся в K-ом столбце решетки процессов (K-ый процесс). При этом матрица A разбивается на блоки по столбцам, т.е. подвергается блочно-столбцовому распределению по решетке процессов. Каждый столбец целиком попадает в локальную память одного из процессов. Дескриптор типа 501 состоит из 7 элементов, за каждым из которых так же закреплено символическое имя, аналогичное именам элементов дескриптора типа 1. Ниже приведена таблица для элементов дескриптора типа 501.

N	Имя		Смысл
1.	DTYPE_A	—	тип дескриптора для ленточной или трехдиагональной матрицы A, распределенной по решетке процессов блочно-столбцовым образом, DTYPE_A = 501
2.	CTXT_A	—	обозначение контекста BLACS'а, соответствующее выбранной решетке процессов, по которой распределяется глобальная матрица A; целое значение, устанавливаемое подпрограммой из пакета BLACS; см. п. 3.1.
3.	N_A	—	число столбцов глобальной матрицы A
4.	NB_A	—	число столбцов в блоках, на которые разбивается глобальная матрица A
5.	CSRC_A	—	номер столбца процесса в решетке, куда был распределен первый столбец глобальной матрицы A
6.	LLD_A	—	ведущая размерность локального массива: $\mathrm{LLD\_A} \geqslant \mathrm{MAX}(1, \mathrm{LOCr}(\mathrm{M\_A}))$; для трехдиагональных матриц игнорируется (положить равной 0)
7.		—	в текущей версии комплекса не используется (положить равным 0)

2.4. Алгоритмы распределения матриц по решетке процессов.

2.4.1. Блочно-циклическое отображение плотных матриц в локальную память процессов.

Как уже говорилось выше, перед обращением к программам комплекса PARALG пользователю необходимо разделить исходную матрицу на блоки и распределить ее по параллельным процессам. Для матриц разных видов приняты свои способы их разбиения и распределения. Здесь мы рассмотрим наиболее общий случай прямоугольной плотной матрицы общего вида.

Для равномерной загрузки параллельных процессов и эффективного использования подпрограмм пакета BLAS при выполнении операций над плотными глобальными матрицами были разработаны специальные способы разбиения таких матриц на прямоугольные блоки ($\mathrm{MB} \times \mathrm{NB}$) и распределения этих блоков по двумерной решетке параллельных процессов. Простой пример конкретного разбиения плотной матрицы на блоки и распределения их по двумерной решетке процессов (это разбиение называется блочно-циклическим распределением) приводится в п. 2.4.2.

Здесь же мы рассмотрим, каким образом блоки, отображенные на один и тот же процесс, располагаются и хранятся в локальной памяти этого процесса. Другими словами, мы выпишем точные формулы, которые связывают элемент

глобальной матрицы, определяемый глобальными индексами I и J (т.е. A(I, J)), с координатами процесса, владеющего этим элементом, в решетке процессов (Pr, Pc) и с расположением этого элемента матрицы в локальной памяти этого процесса.

Рассмотрим сначала, для простоты, эти связи на примере одномерного случая, т.е. размещение одномерного глобального массива длины N, разбитого на блоки длины NB, по одномерной решетке из P процессов, перенумерованных начиная с 0 до (P-1). Элементы самого глобального массива нумеруются от 1 до N. Прежде всего, массив делится на смежные блоки размера NB. Когда N не делится на NB нацело, последний блок массива элементов будет содержать только mod (N, NB) элементов вместо NB элементов. Блоки, на которые разбивается исходный массив, нумеруются (как и процессы) начиная с 0. Они распределяются по процессам подобно тому, как сдается колода карт участникам игры — по кругу (т.е. циклически).

Другими словами, мы предполагаем, что процесс с номером 0 получает первый блок (с номером 0), k-й блок связывается с процессом, имеющим координату (номер) mod (k, P). Блоки, связанные с одним и тем же процессом, хранятся в памяти рядом (в смежных, прилегающих частях). Отображение элемента массива с глобальным индексом I определяется следующим соотношением:

$$\mathrm{I} = \mathrm{k} * \mathrm{NB} + \mathrm{x} = (\mathrm{m} * \mathrm{P} + \mathrm{p}) * \mathrm{NB} + \mathrm{x},$$

где

I	—	глобальный индекс элемента глобального массива;
m	—	локальная координата блока, в котором этот элемент расположен;
p	—	координата процесса, владеющего этим блоком;
x	—	локальная координата элемента внутри того блока, где находится элемент глобального массива с индексом I.

Легко установить соотношения между этими переменными:

$$\begin{aligned} \mathrm{p} &= \big[(\mathrm{I}-1)/\mathrm{NB}\big] \bmod \mathrm{P}, \\ \mathrm{m} &= \big[(\mathrm{I}-1)/(\mathrm{P} * \mathrm{NB})\big], \\ \mathrm{x} &= \mathrm{mod}\,(\mathrm{I}-1,\ \mathrm{NB}) + 1. \end{aligned}$$

Эти уравнения позволяют определить локальную информацию (т.е. локальный индекс m * NB+x), так же как и координату процесса p, соответствующую

глобальному элементу, идентифицируемому его глобальным индексом I, и наоборот.

В таблице ниже показано отображение в локальную память при разбиении массива на блоки, когда P=2, N=16 и NB=8.

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
m*NB+x	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8

Вовсе не обязательно всегда делать распределение блоков начиная с процесса с номером 0. Иногда бывает полезно начать распределение данных с процесса, имеющего отличную от нуля координату SRC. В этом случае соотношения становятся такими:

$$\begin{aligned} \mathrm{p} &= \big(\mathrm{SRC} + [(\mathrm{I}-1)/\mathrm{NB}]\big) \bmod \mathrm{P}, \\ \mathrm{m} &= \big[(\mathrm{I}-1)/(\mathrm{P} * \mathrm{NB})\big], \\ \mathrm{x} &= \mathrm{mod}\,(\mathrm{I}-1, \mathrm{NB}) + 1. \end{aligned}$$

Ниже в таблице показано размещение блоков при P=2, SRC=1, NB=3 и N=16.

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0
m	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2
x	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1
m*NB+x	1	2	3	1	2	3	4	5	6	4	5	6	7	8	9	7

В двумерном случае предположим, что матрица разделена на $\mathrm{MB} \times \mathrm{NB}$ блоки и что первый блок передан процессу с координатами $(\mathrm{RSRC}, \mathrm{CSRC})$. Формула, приведенная выше для одномерного случая, должна использоваться повторно независимо для каждого измерения решетки процессов $\mathrm{Pr} \times \mathrm{Pc}$. Например, элемент матрицы (I, J) находится в процессе с координатами $(\mathrm{Pr}, \mathrm{Pc})$ внутри локального блока (m, n) в позиции (x, y), задаваемой формулами

$$\begin{aligned} (\mathrm{m}, \mathrm{n}) &= \Big(\big[(\mathrm{I}-1)/(\mathrm{Pr} * \mathrm{MB})\big], \quad \big[(\mathrm{J}-1)/(\mathrm{Pc} * \mathrm{NB})\big]\Big), \\ (\mathrm{Pr}, \mathrm{Pc}) &= \Big(\big(\mathrm{RSRC} + [(\mathrm{I}-1)/\mathrm{MB}]\big) \bmod \mathrm{Pr}, \quad \big(\mathrm{CSRC} + [(\mathrm{J}-1)/\mathrm{NB}]\big) \bmod \mathrm{Pc}\Big), \\ (\mathrm{x}, \mathrm{y}) &= \big(\mathrm{mod}\,(\mathrm{I}-1, \mathrm{MB}) + 1, \quad \mathrm{mod}\,(\mathrm{J}-1, \mathrm{NB}) + 1\big). \end{aligned}$$

Эти формулы показывают, как матрица A размера M_A $\times$ N_A отображается и хранится на решетке процессов. Матрица сначала разбивается на MB_A $\times$ NB_A блоки начиная с ее верхнего левого угла. Эти блоки затем равномерно распределяются по решетке процессов циклическим образом.

Каждый процесс владеет набором блоков, которые хранятся рядом (смежно) по столбцам в двумерном массиве, расположенном в памяти по "столбцам".

Это соглашение о локальном размещении позволяет эффективно использовать иерархию локальной памяти посредством вызова пакета BLAS для подмассивов, которые могут быть больше, чем один блок размера MB_A $\times$ NB_A.

На рисунке ниже представлено отображение матрицы 5×5, разделенной на блоки размером 2×2 на решетку процессов размером 2×2 (т.е. M_A=N_A=5, Pr=Pc=2 и MB_A=NB_A=2). Локальные элементы каждого столбца матрицы хранятся рядом в памяти каждого процесса.

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

	0			1	
0	a_{11}	a_{12}	a_{15}	a_{13}	a_{14}
	a_{21}	a_{22}	a_{25}	a_{23}	a_{24}
	a_{51}	a_{52}	a_{55}	a_{53}	a_{54}
1	a_{31}	a_{32}	a_{35}	a_{33}	a_{34}
	a_{41}	a_{42}	a_{45}	a_{43}	a_{44}

Цифры 0 и 1 в левой и верхней части последней таблицы означают номера строк и столбцов решетки процессов соответственно. Важной задачей для пользователя является определение числа строк и столбцов плотной глобальной матрицы, которое получает каждый конкретный процесс. Для выполнения этой функции существует служебная подпрограмма NUMROC.

Для локального числа строк используется обозначение LOCr ( ), а для локального числа столбцов — LOCc ( ). Значения LOCr ( ) и LOCc ( ), получаемые подпрограммой NUMROC, являются результатом точных вычислений.

Однако если требуется понять общую идею вычисления размера локального массива, то можно проделать следующее грубое вычисление верхней границы этой величины:

а) верхняя граница для LOCr ( ) оценивается по формуле

$$\mathrm{LOCr}(\ ) = \frac{\dfrac{\mathrm{M\_A} + \mathrm{MB\_A} - 1}{\mathrm{MB\_A}} + \mathrm{Pr} - 1}{\mathrm{Pr}} * \mathrm{MB\_A},$$

или

$$\mathrm{LOCr}(\ ) = \big[[\mathrm{M\_A}/\mathrm{MB\_A}]/\mathrm{Pr}\big] * \mathrm{MB\_A};$$

б) величина $\mathrm{LOCc}(\ )$ оценивается по формуле

$$\mathrm{LOCc}(\ ) = \frac{\dfrac{\mathrm{N\_A} + \mathrm{NB\_A} - 1}{\mathrm{NB\_A}} + \mathrm{Pc} - 1}{\mathrm{Pc}} * \mathrm{NB\_A},$$

или

$$\mathrm{LOCc}(\ ) = \big[[\mathrm{N\_A}/\mathrm{NB\_A}]/\mathrm{Pc}\big] * \mathrm{NB\_A}.$$

Заметим, что эти вычисления могут привести к очень большой переоценке величины действительно требующегося пространства.

2.4.2. Пример блочно-циклического распределения плотной матрицы по решетке процессов.

Приводимый в настоящем разделе пример является простой иллюстрацией блочно-циклического распределения плотной глобальной матрицы A по двумерной решетке процессов.

В соответствии с принятой схемой, сначала плотная матрица A размера $\mathrm{M} \times \mathrm{N}$ разделяется на блоки размера $\mathrm{MB} \times \mathrm{NB}$ начиная с левого верхнего угла этой матрицы. Эти блоки затем равномерно распределяются по каждому измерению решетки процессов. Математические соотношения, соответствующие такой схеме распределения, приводятся в п. 2.4.1.

Таким образом каждый процесс владеет набором блоков, которые расположены в его локальной памяти рядом в двумерном массиве, хранящемся по столбцам (см. ниже).

Ниже на рисунке показано разбиение матрицы A размером 9×9 на блоки размером 2×2.

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}	a_{49}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}	a_{57}	a_{58}	a_{59}
a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	a_{66}	a_{67}	a_{68}	a_{69}
a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	a_{77}	a_{78}	a_{79}
a_{81}	a_{82}	a_{83}	a_{84}	a_{85}	a_{86}	a_{87}	a_{88}	a_{89}
a_{91}	a_{92}	a_{93}	a_{94}	a_{95}	a_{96}	a_{97}	a_{98}	a_{99}

Далее показано отображение полученных блоков на решетку процессов размером 2×3 (двумерное блочно-циклическое распределение данных).

Чтобы понять, как распределятся изображенные выше блоки по процессам решетки, сначала напишем на каждой из клеток (блоков) рисунка координаты процессов, куда они должны быть распределены в соответствии с установленным блочно–циклическим распределением, описанным в п. 2.4.1.

(0,0)	(0,1)	(0,2)	(0,0)	(0,1)
(1,0)	(1,1)	(1,2)	(1,0)	(1,1)
(0,0)	(0,1)	(0,2)	(0,0)	(0,1)
(1,0)	(1,1)	(1,2)	(1,0)	(1,1)
(0,0)	(0,1)	(0,2)	(0,0)	(0,1)

Пусть первый блок распределяется в процесс $(0, 0)$. Тогда все блоки, расположенные с ним в той же строке будут распределяться в ту же строку решетки процессов, т.е. первая координата в этих клетках будет равна 0.

Вторая же координата (столбца решетки) будет циклически изменяться: $0, 1, 2, 0, 1$, поскольку столбцов в решетке только 3 (с координатами $0, 1, 2$).

Все блоки, расположенные с первым блоком в том же самом столбце, будут распределяться в тот же самый столбец решетки процессов. Иными словами, вторая координата в этих клетках будет равна 0. Первая же координата (строки решетки) будет циклически изменяться: $0, 1, 0, 1, 0$, поскольку число строк в решетке только 2 (с координатами 0 и 1).

Другими словами, каждая координата независимо от другой (в строках — слева направо, а в столбцах — сверху вниз) циклически изменяется.

Если теперь мы соединим вместе (пристыкуем) все блоки, имеющие одинаковые координаты процесса, то получим массив элементов, который и должен расположиться в локальной памяти процесса с такими координатами.

Сборка клеток (блоков) с одинаковыми координатами делается так:

- из этих клеток выбирается расположенная левее и выше всех;
- к ней пристыковываются снизу клетки, расположенные в этом же столбце, тем самым получается первый столбец блоков;
- затем берется клетка, расположенная правее в той же строке клеток, что и самая первая; к ней пристыковываются все расположенные ниже в том же столбце, тем самым получаем следующий столбец клеток;
- после того как собраны все столбцы клеток с одинаковыми координатами, пристыковываем их по-очереди справа к первому столбцу с такими же координатами.

Тем самым получаем единый двумерный массив, расположенный в локальной памяти процесса с указанными в клетках координатами.

Ниже представлено расположение элементов исходной матрицы во всех процессах решетки после завершения процесса блочно–циклического распределения. Вверху указаны номера столбцов решетки процессов, слева — номера строк решетки процессов.

	0				1			2	
0	a_{11}	a_{12}	a_{17}	a_{18}	a_{13}	a_{14}	a_{19}	a_{15}	a_{16}
	a_{21}	a_{22}	a_{27}	a_{28}	a_{23}	a_{24}	a_{29}	a_{25}	a_{26}
	a_{51}	a_{52}	a_{57}	a_{58}	a_{53}	a_{54}	a_{59}	a_{55}	a_{56}
	a_{61}	a_{62}	a_{67}	a_{68}	a_{63}	a_{64}	a_{69}	a_{65}	a_{66}
	a_{91}	a_{92}	a_{97}	a_{98}	a_{93}	a_{94}	a_{99}	a_{95}	a_{96}
1	a_{31}	a_{32}	a_{37}	a_{38}	a_{33}	a_{34}	a_{39}	a_{35}	a_{36}
	a_{41}	a_{42}	a_{47}	a_{48}	a_{43}	a_{44}	a_{49}	a_{45}	a_{46}
	a_{71}	a_{72}	a_{77}	a_{78}	a_{73}	a_{74}	a_{79}	a_{75}	a_{76}
	a_{81}	a_{82}	a_{87}	a_{88}	a_{83}	a_{84}	a_{89}	a_{85}	a_{86}

Ниже в таблице указаны характеристики локальных массивов для каждого из процессов решетки.

Координаты процесса	LLD_A	LOCr (M_A)	LOCc (N_A)
(0,0)	5	5	4
(0,1)	5	5	3
(0,2)	5	5	2
(1,0)	4	4	4
(1,1)	4	4	3
(1,2)	4	4	2

Число строк LOCr и число столбцов LOCc матрицы A, которыми владеет конкретный процесс, могут отличаться у разных процессов в решетке. Подобно этому, для каждого процесса в решетке процессов существует ведущая локальная размерность LLD. Ее величина может быть различной для разных процессов в решетке процессов. Например, как мы можем видеть на рисунке выше, локальный массив, хранящийся в строке решетки процессов с номером 0, должен иметь ведущую локальную размерность LLD не меньше 5, а хранящийся в строке с номером 1 — не меньше 4.

2.4.3 Блочно-столбцовое разбиение и схема размещения в локальной памяти ленточных матриц.

Как уже говорилось в п. 2.3.2.3, при работе с ленточными глобальными матрицами A размера M на N используется блочно-столбцовое разбиение их на блоки. Это означает, что в блок объединяются несколько соседних столбцов матрицы. Все блоки имеют одинаковый размер, т.е. содержат одно и то же число столбцов (за исключением, может быть, последнего блока). Последний блок может содержать меньшее число столбцов, чем остальные.

Размер блока выбирается исходя из числа столбцов N глобальной матрицы A и числа процессоров P, используемых для решения задачи. При этом используется одномерная решетка процессов размером 1×P. Каждый процесс должен получить один из блоков матрицы. Таким образом, размер блока NB определяется из неравенства $NB \geqslant [N/P]$. Если N делится на P нацело, то все блоки, распределяемые на все процессы, имеют одинаковый размер. В противном случае последний блок, распределяемый на последний процесс, имеет размер, меньший NB. При этом K-й столбец матрицы A попадает в память процесса с номером [K/NB]. Как уже говорилось, процессы нумеруются от 0 до (P - 1). Координаты первого процесса в такой решетке суть (0,0), а последнего — (0, P-1). На рисунке условно показано разбиение некоторой глобальной матрицы A размера M×N на 3 блока по столбцам для распределения ее по трем процессам (координаты процессов в решетке указаны под соответствующими блоками).

	1		N
1			
M			
	(0,0)	(0,1)	(0,2)

Такой способ распределения матрицы A позволяет использовать в подпрограммах высокоэффективные алгоритмы, называемые “разделяй и властвуй”.

Рассмотрим схему размещения элементов глобальной ленточной матрицы A в локальной памяти параллельных процессов на конкретных примерах.

Пусть матрица A имеет размеры 7×7 и ширину ленты, равную 2:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & 0 \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} \end{pmatrix}$$

Пусть матрица распределяется по решетке процессов 1×3 с размером блоков NB_A = 3.

Рассмотрим несколько случаев распределения по решетке процессов несимметричных и симметричных положительно определенных ленточных матриц.

2.4.3.1. Случай несимметричной ленточной матрицы, при факторизации которой используются алгоритмы без выбора ведущего элемента.

Пусть число поддиагоналей BWL матрицы A равно 2, а число наддиагоналей BWU равно 2.

Ниже на рисунке представлено распределение такой матрицы по процессам. Символом "$*$" отмечены элементы в двумерном локальном массиве, которые в дальнейших вычислениях не используются.

Процессы	(0,0)	(0,1)	(0,2)
	$*$ $*$ a_{13}	a_{24} a_{35} a_{46}	a_{57}
	$*$ a_{12} a_{23}	a_{34} a_{45} a_{56}	a_{67}
	a_{11} a_{22} a_{33}	a_{44} a_{55} a_{66}	a_{77}
	a_{21} a_{32} a_{43}	a_{54} a_{65} a_{76}	$*$
	a_{31} a_{42} a_{53}	a_{64} a_{75} $*$	$*$

Из этого рисунка видно, что ведущая размерность этих локальных массивов (т.е. число строк в блоках глобальной матрицы A, которые помещаются в каждый процессор) должна быть равна BWL+1+BWU (поскольку элементы изображенных массивов заносятся в локальную память по столбцам). О ведущей размерности локальных массивов см. п. 2.3.2.1.

2.4.3.2. Случай несимметричной ленточной матрицы, при факторизации которой используются алгоритмы с выбором ведущего элемента по столбцам.

В этом случае, в отличие от предыдущего, необходимо дополнительное пространство для хранения информации о производимых в процессе факторизации перестановках. А именно, в каждом локальном массиве количество элементов в начале каждого столбца должно быть увеличено на величину, равную сумме числа поддиагоналей и числа наддиагоналей, т.е. BWL+BWU.

В рассматриваемом нами примере имеем BWL=2 и BWU=2. Ниже на рисунке представлено распределение глобальной ленточной матрицы по процессам с выделением необходимого дополнительного пространства. Дополнительно резервируемые элементы в каждом локальном массиве обозначены буквами "F".

Процессы	(0,0)			(0,1)			(0,2)
	F	F	F	F	F	F	F
	F	F	F	F	F	F	F
	F	F	F	F	F	F	F
	F	F	F	F	F	F	F
	$*$	$*$	a_{13}	a_{24}	a_{35}	a_{46}	a_{57}
	$*$	a_{12}	a_{23}	a_{34}	a_{45}	a_{56}	a_{67}
	a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}
	a_{21}	a_{32}	a_{43}	a_{54}	a_{65}	a_{76}	$*$
	a_{31}	a_{42}	a_{53}	a_{64}	a_{75}	$*$	$*$

Из этого рисунка видно, что ведущая размерность такого локального массива должна быть равна 2$*$(BWL+BWU)+1.

2.4.3.3. Случай симметричной положительно определенной ленточной матрицы, у которой в памяти сохраняется только нижний треугольник (параметр UPLO в подпрограммах обработки таких матриц полагается равным "L").

Пусть ширина ленты такой матрицы BW=2. Ниже на рисунке представлено распределение такой матрицы (7×7) по той же решетке процессов (1×3).

Процессы	(0,0)			(0,1)			(0,2)
	a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}
	a_{21}	a_{32}	a_{43}	a_{54}	a_{65}	a_{76}	$*$
	a_{31}	a_{42}	a_{53}	a_{64}	a_{75}	$*$	$*$

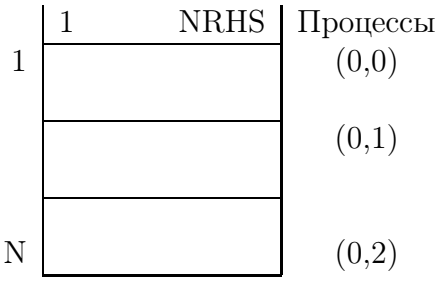
2.4.3.4. Случай симметричной положительно определенной ленточной матрицы, у которой в памяти сохраняется только верхний треугольник (параметр UPLO в подпрограммах обработки таких матриц полагается равным "U").

Ниже на рисунке представлено распределение такой матрицы по той же решетке процессов (1×3).

Процессы	(0,0)	(0,1)	(0,2)
	$*\quad *\quad a_{31}$	$a_{42}\ a_{53}\ a_{64}$	a_{75}
	$*\quad a_{21}\ a_{32}$	$a_{43}\ a_{54}\ a_{65}$	a_{76}
	$a_{11}\ a_{22}\ a_{33}$	$a_{44}\ a_{55}\ a_{66}$	a_{77}

Из двух последних рисунков видно, что ведущая размерность такого локального массива равна BW+1.

Кроме того, при решении систем линейных уравнений предполагается, что вектор (матрица) правых частей B (размером N×NRHS) распределяется по той же решетке процессов, что и описанные выше матрицы A, но только не блочно-столбцовым, а блочно-строчным образом. Это означает, что в блоки объединяются соседние строки матрицы B (или просто соседние элементы, если B — вектор). Ниже на рисунке условно представлено блочно-строчное разбиение матрицы B для распределения ее по той же решетке процессов (1×3), что и соответствующая ленточная (или трехдиагональная) матрица A.



Распределение элементов вектора B по этим процессам будет следующим:

(0,0)	(0,1)	(0,2)
b_1	b_4	b_7
b_2	b_5	$*$
b_3	b_6	$*$

Конкретные примеры распределения матрицы A и вектора B по процессам можно найти в разделе “Пример использования” описаний подпрограмм в разделе Интернет-ресурса по численному анализу “Комплекс программ по линейной алгебре для вычислений на распределенной памяти” (см."http://num-anal.srcc.msu.ru/par_prog/cat563n.htm").

2.4.4. Схема размещения в локальной памяти трехдиагональных матриц.

Рассмотрим пример трехдиагональной глобальной матрицы A размером 7×7 (N_A=7):

$$\begin{matrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & 0 \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & 0 & a_{76} & a_{77} \end{matrix}$$

Как и ленточные матрицы, трехдиагональные матрицы следует распределять по одномерной решетке процессов. Пусть нам необходимо распределить указанную матрицу по решетке процессов 1×3.

2.4.4.1. Если матрица A несимметричная, то она представляется в памяти в виде трех отдельных векторов (D, DL, DU), каждый из которых содержит элементы одной из диагоналей.

Все три вектора должны иметь одинаковую длину, равную длине вектора главной диагонали (D), т.е. быть выровнены. В нашем примере длина всех векторов должна быть равна 7. При этом элементы нижней диагонали должны быть сдвинуты к концу вектора DL, т.е. первый элемент нижней диагонали должен занимать положение второго элемента вектора DL. И, наоборот, элементы верхней диагонали должны быть сдвинуты к началу вектора DU, т.е. ее последний элемент должен занимать положение предпоследнего элемента вектора DU.

Сказанное проиллюстрировано на рисунке ниже. На нем также показано, как эти векторы должны быть разделены на блоки (если величина блока NB_A выбрана равной 3). Как и в случае ленточных матриц, здесь действует аналогичное правило: в локальную память каждого из процессов должен быть распределен один блок главной диагонали матрицы A, а также по одному блоку двух других диагоналей.

Процессы

	(0,0)	(0,1)	(0,2)
DL	$*\ a_{21}\ a_{32}$	$a_{43}\ a_{54}\ a_{65}$	a_{76}
D	$a_{11}\ a_{22}\ a_{33}$	$a_{44}\ a_{55}\ a_{66}$	a_{77}
DU	$a_{12}\ a_{23}\ a_{34}$	$a_{45}\ a_{56}\ a_{67}$	$*$

Символом “$*$” отмечены элементы, которые не используются при вычислениях.

2.4.4.2. Если трехдиагональная матрица A является симметричной положительно определенной, то она должна быть представлена в памяти в виде двух векторов, один из которых (D), содержит элементы главной диагонали, а другой (E) — элементы одной из кодиагоналей.

Если предполагается, что в памяти сохраняется нижняя диагональ матрицы A (параметр UPLO в соответствующей подпрограмме полагается равным "L"), то вектор E содержит элементы нижней диагонали.

Если предполагается, что в памяти сохраняется верхняя диагональ матрицы A (параметр UPLO полагается равным "U"), то вектор E содержит элементы верхней диагонали. Оба вектора D и E должны иметь одинаковую длину, равную длине вектора D, т.е. быть выровнены. При этом элементы верхней или нижней диагонали сдвигаются к началу вектора E, т.е. последний элемент кодиагонали становится предпоследним элементом вектора E.

Сказанное проиллюстрировано на двух рисунках ниже (первый рисунок для параметра UPLO="L", второй — для UPLO="U"). На них также показано распределение векторов, разбитых на блоки длиной NB_A=3, в локальную память каждого из трех процессов.

Процессы

	(0,0)	(0,1)	(0,2)
D	$a_{11}\ a_{22}\ a_{33}$	$a_{44}\ a_{55}\ a_{66}$	a_{77}
E	$a_{21}\ a_{32}\ a_{43}$	$a_{54}\ a_{65}\ a_{76}$	

Процессы

	(0,0)	(0,1)	(0,2)
D	$a_{11}\ a_{22}\ a_{33}$	$a_{44}\ a_{55}\ a_{66}$	a_{77}
E	$a_{12}\ a_{23}\ a_{34}$	$a_{45}\ a_{56}\ a_{67}$	

Правила распределения по решетке процессов вектора (или матрицы) правых частей B, соответствующего ленточной или трехдиагональной матрице коэффициентов системы A, можно найти в конце п. 2.4.3.

2.5. Алгоритмы решения систем линейных алгебраических уравнений.

Для решения систем линейных алгебраических уравнений используются модифицированные варианты известных алгоритмов.

Пусть имеется линейная система

$$AX = B, \tag{1}$$

где

A — невырожденная квадратная матрица;

B — либо вектор правой части, если решается единственная система, либо матрица, состоящая из столбцов правых частей, если решается несколько систем с одной и той же матрицей A;

X — вектор неизвестных, если B — вектор, или матрица, состоящая из векторов неизвестных, если B — матрица.

Под векторами понимаются векторы-столбцы.

Для решения указанной выше системы применяются специальные блочные реализации (блочные алгоритмы) для прямого и обратного хода метода Гаусса. Решение системы разбито на два этапа:

— факторизация матрицы системы;

— решение системы с использованием результатов, полученных на первом этапе.

Каждый из этапов осуществляется посредством обращения к специальным базовым подпрограммам комплекса (см. п. 3.5).

1. Перечислим виды факторизации первого этапа, зависящие от свойств матрицы A.

1.1. Для плотных матриц общего вида применяется LU-факторизация методом Гаусса с выбором ведущего элемента по столбцам.

1.2. Для симметричных или эрмитовых положительно определенных матриц применяется разложение (факторизация) Холецкого (известное также как метод квадратного корня).

1.3. Для ленточных матриц общего вида применяется LU-факторизация методом Гаусса с выбором ведущего элемента по столбцам.

1.4. Для ленточных матриц общего вида с диагональным преобладанием, включая трехдиагональные матрицы общего вида, применяется LU-факторизация Гаусса без выбора ведущего элемента.

1.5. Для симметричных и эрмитовых положительно определенных ленточных матриц применяется разложение Холецкого.

1.6. Для симметричных и эрмитовых положительно определенных трехдиагональных матриц применяется LDL^T-факторизация (или U^TDU-факторизация), где U и L — двухдиагональные верхняя и нижняя матрицы соответственно.

Указанные факторизации выполняются подпрограммами, указанными в п. 3.5.

2. После этого следует второй этап — непосредственное решение системы с использованием полученной на предыдущем этапе факторизации матрицы A при помощи подпрограммам, указанным в п. 3.5..

3. Практические сведения по использованию параллельных программ.

3.1. Предварительные действия, необходимые для обращения к программам.

С учетом сказанного в п. 2.2, прежде всего необходимо начать с определения конкретной решетки процессов. В программах комплекса PARALG используются следующие обозначения:

NPROW — число строк решетки процессов,

NPCOL — число столбцов решетки процессов,

NP=NPROW*NPCOL — общее число процессов решетки.

Для идентификации каждого процесса используется пара (MYROW, MYCOL):

MYROW — номер строки процесса в решетке,

MYCOL — номер столбца процесса в решетке,

где

$$0 \leqslant \text{MYROW} < \text{NPROW},$$

$$0 \leqslant \text{MYCOL} < \text{NPCOL}.$$

Еще одна процедура, которую необходимо проделать пользователю перед обращением к одной из целевых программ комплекса, это — разделить исходную(ые) матрицу(ы) на блоки и распределить эти блоки по определенным правилам по всем процессам решетки.

Сказанное необходимо проделать по той причине, что программы ориентированы на более эффективную блочную реализацию операций с матрицами. Поэтому при обращении к подпрограммам комплекса необходимо задавать, в частности, величину указанных блоков, т.е.

MB — число строк матрицы в блоке;

NB — число столбцов матрицы в блоке.

Описание конкретных правил распределения матриц по параллельным процессам приводится в п. 3.2, а также в пп. 2.4.3, 2.4.4.

Проделав указанные действия, пользователь может вызвать одну из программ комплекса, передав ей через параметры всю информацию об исходных матрицах, а также о том, как распределены их элементы в локальную память каждого из процессов решетки.

После выполнения требуемых вычислений и получения результатов пользователю необходимо освободить использованную решетку процессов и осуществить выход из пакета BLACS.

Ниже подробнее описываются общие обязательные действия программиста при обращении к любой целевой программе комплекса.

3.1.1. Инициализация решетки процессов.

После присвоения конкретных значений параметрам решетки NPROW и NPCOL инициализация решетки процессов проще всего производится посредством обращения к подпрограмме SL_INIT из разряда служебных (tool routines). Эта подпрограмма состоит из обращений к необходимым подпрограммам пакета BLACS.

Подпрограмма SL_INIT инициализирует выбранную решетку процессов, упорядочивая процессы по строкам, а также присваивает служебному параметру ICTXT некоторое целое значение. Это значение закрепляет за выбранной пользователем решеткой процессов статус конкретной области действия массовых операций передачи сообщений. Параметр ICTXT называется указателем контекста, системным контекстом, контекстом BLACS'а или просто контекстом.

Пользователь, работая с выбранной решеткой, не должен производить никаких действий с параметром ICTXT. Необходимо только передавать его в качестве входного параметра при обращении к другим подпрограммам пакета BLACS, а также к подпрограммам комплекса, куда передаются сведения об обрабатываемых матрицах.

Обращение к подпрограмме SL_INIT выглядит так:

CALL SL_INIT (ICTXT, NPROW, NPCOL),

где

ICTXT — специфицирует контекст BLACS'а (глобальный выходной параметр, тип: целый);

NPROW — число строк в создаваемой решетке процессов (глобальный входной параметр, тип: целый);

NPCOL — число столбцов в создаваемой решетке процессов (глобальный входной параметр, тип: целый).

Программы комплекса написаны с расчетом на использование стиля SPMD (Single Program Multiple Data). Это означает, что одна и та же программа пользователя будет загружена и одновременно выполняться всеми процессами решетки. Поэтому при необходимости выполнения некоторых действий только на некоторых из этих процессов, требуется определить координаты процесса, на котором выполняется данный экземпляр программы. Как уже указывалось, они обычно обозначаются идентификаторами (MYROW, MYCOL). Например, пусть необходимо произвести распечатку входных параметров, одинаковых для всех процессов. Чаще всего это делают с процесса с координатами (0, 0).

Для определения координат процесса используют подпрограмму с именем BLACS_GRIDINFO пакета BLACS:

CALL BLACS_GRIDINFO (ICTXT, NPROW, NPCOL, MYROW, MYCOL),

где

MYROW — номер строки данного процесса в решетке процессов, $0 \leqslant$ MYROW $<$ NPROW (глобальный выходной параметр);

MYCOL — номер столбца данного процесса в решетке процессов, $0 \leqslant$ MYCOL $<$ NPCOL (глобальный выходной параметр).

Три первых параметра описаны выше, с той только разницей, что в данном случае параметр ICTXT — входной.

3.1.2. Распределение матриц по решетке процессов.

Поскольку каждый процесс производит операции над своей частью исходной глобальной матрицы, до обращения к программам комплекса необходимо распределить по определенным правилам матрицу A по решетке процессов. Это является обязанностью пользователя (подробности этого процесса описаны в п. 2.4, а также в п 3.2).

Каждой глобальной матрице, распределенной по решетке процессов, должен быть приписан дескриптор, содержащий информацию о том, как именно было произведено это распределение. Дескриптор представляет собой одномерный массив из 9 или 7 элементов целого типа. Подробное описание дескрипторов можно найти в п. 2.3.2. Здесь мы покажем, как инициализировать дескрипторы глобальной матрицы A (DESCA) и вектора правых частей системы уравнений B (DESCB).

Если матрица A — плотная, то проще всего инициализировать дескриптор посредством обращения к служебной подпрограмме DESCINIT.

Ниже приводится обращение к этой подпрограмме при инициализации дескриптора DESCA. Подобные обращения можно найти, например, в текстах тестовых примеров к программам TPDGESV.f и TPDPOSV.f решения систем уравнений с плотными матрицами.

CALL DESCINIT(DESCA, M, N, MB, NB, RSRC, CSRC, ICTXT, MXLLDA, INFO).

Этот вызов подпрограммы DESCINIT эквивалентен следующим операторам присваивания:

DESCA(1) = 1
DESCA(2) = ICTXT
DESCA(3) = M
DESCA(4) = N
DESCA(5) = MB
DESCA(6) = NB
DESCA(7) = RSRC
DESCA(8) = CSRC
DESCA(9) = MXLLDA

Здесь первый элемент обозначает тип дескриптора для плотной матрицы, второй — обсуждавшийся выше служебный параметр подпрограммы пакета BLACS (контекст).

Другие параметры указанного вызова имеют следующий смысл:

M, N — число строк и столбцов глобальной матрицы A(M,N);

MB, NB — число строк и столбцов в блоках, на которые разбивается матрица A;

RSRC, CSRC — координаты процесса в решетке, куда размещается пользователем первый элемент распределяемой глобальной матрицы;

MXLLDA — ведущая локальная размерность матрицы A;

INFO — целая переменная, служащая для сообщения о результате работы программы (глобальный выходной параметр).

Простой пример отображения глобальной матрицы на решетку процессов можно найти, например, в тестовом примере к подпрограмме решения систем с плотной матрицей (см. TPDGESV.f). В нем эти действия выполняются в подпрограмме с именем MATINIT (или PDMATINIT), которая является частью тестового примера.

Приведем здесь также обращение к подпрограмме DESCINIT при инициализации дескриптора DESCB для матрицы (вектора) правых частей системы уравнений B с плотной матрицей системы A. Подобные обращения можно найти в текстах тестовых примеров к программам TPDGESV.f и TPDPOSV.f решения системы уравнений с плотными матрицами.

CALL DESCINIT
(DESCB, N, NRHS, NB, NBRHS, RSRC, CSRC, ICTXT, MXLLDA, INFO)

Этот вызов подпрограммы DESCINIT эквивалентен следующим операторам присваивания:

DESCB(1) = 1
DESCB(2) = ICTXT
DESCB(3) = N
DESCB(4) = NRHS
DESCB(5) = NB
DESCB(6) = NBRHS
DESCB(7) = RSRC
DESCB(8) = CSRC
DESCB(9) = MXLLDB

Здесь первый элемент обозначает тип дескриптора, второй — служебный параметр подпрограммы пакета BLACS (контекст).

Другие параметры указанного вызова имеют следующий смысл:

N, NRHS — число строк и столбцов матрицы B (N×NRHS); если B — вектор, то NRHS = 1;

NB, NBRHS — число строк и столбцов блоков, на которые разбивается матрица B; если B — вектор, то NBRHS=1;

RSRC, CSRC — координаты процесса, куда размещается пользователем первый элемент глобальной матрицы (вектора);

MXLLDB — ведущая локальная размерность матрицы (вектора) B;

INFO — целая переменная, служащая для сообщения о результате работы программы (глобальный выходной параметр).

Примеры инициализации дескрипторов DESCA и DESCB, имеющих типы 501 и 502, можно найти в разделе “Пример использования” описания одной из подпрограмм для решения систем с ленточной матрицей A (http://numanal.srcc.msu.ru/par_prog/cat563n.htm).

3.1.3. Освобождение решетки процессов.

После того как необходимые вычисления на решетке процессов были завершены, требуется освободить решетку процессов посредством вызова подпрограммы
BLACS_GRIDEXIT. Когда завершены все вычисления, выход из пакета BLACS должен быть осуществлен посредством обращения к подпрограмме BLACS_EXIT.

Типичный пример фрагмента программы, включающий в себя эти действия:

```
CALL BLACS_GRIDEXIT(ICTXT)
CALL BLACS_EXIT(0).
```

3.2. Программирование распределения матриц по решетке процессов.

В настоящем учебном пособии рассматриваются три способа реализации на языке ФОРТРАН рассылки элементов исходных глобальных матриц по параллельным процессам, поскольку каждый из них удобнее использовать в трех разных ситуациях.

1. Первый способ удобен только для небольших матриц и требует от пользователя самостоятельного размещения элементов исходных матриц в локальной памяти параллельных процессов в соответствии с общими правилами, описанными в п. 2.4. При этом используются обычные операторы присваивания языка ФОРТРАН. Такой способ достаточно трудоемок и может использоваться для целей обучения и усвоения общих правил распределения матриц (см. п. 2.4). Фортранная подпрограмма такого способа распределения приводится в конце Приложения 1.

2. Второй способ удобнее как для небольших матриц, так и для матриц среднего размера с регулярной структурой или матриц любого размера, элементы которых задаются математической формулой.

Этот способ гораздо проще для пользователя, поскольку не требует самостоятельного определения местоположения элементов матриц в локальной памяти каждого из параллельных процессов. Он опирается на использование служебной подпрограммы PDELSET, которая сама (в соответствии с упомянутыми общими правилами, см. п. 2.4.1) заносит элемент с индексами I, J исходной матрицы A(I,J) в определенное место локальной памяти того или другого из параллельных процессов.

На ФОРТРАНе это выливается в написание некоторого числа циклов, внутри которых стоит обращение к подпрограмме PDELSET.

Фортранная подпрограмма с реализацией такого способа распределения приводится в конце Приложения 2.

Еще раз подчеркнем, что хотя в конце Приложения 1 и Приложения 2 приводятся два разных способа распределения одной и той же исходной матрицы, результат распределения будет одним и тем же. Подробный разбор приводимого в этих приложениях примера дается в п. 3.4.

3. Третий способ может быть использован для обработки больших исходных матриц, численное значение элементов которых хранится в виде файлов на внешней памяти (наиболее частый на практике случай).

Тогда можно воспользоваться специальной служебной подпрограммой, последовательно считывающей из внешнего файла элементы исходной матрицы и распределяющей их по установленным правилам в локальную память того или другого из параллельных процессов.

Пример обращения к такой служебной фортранной подпрограмме PDLAREAD

приводится в Приложении 3. Однако пользователь может написать свою аналогичную подпрограмму, если какие-либо детали размещения исходной матрицы в файле отличаются от приводимого примера.

В этом же примере содержится обращение к служебной подпрограмме с именем PDLAWRITE записи полученных результатов в файл. Тексты этих подпрограмм можно найти, перейдя по гиперссылкам в конце Интернет-страницы с адресом

http://num-anal.srcc.msu.ru/par_prog/instr.htm.

Приводимые в Приложениях 1 и 2 примеры программирования распределения матриц по решетке процессов относятся к плотным исходным матрицам.

Аналогичные два первых способа программирования при распределении по процессам ленточных и трехдиагональных матриц можно найти в разделах "Пример использования" в описаниях целевых программ комплекса для решения линейных систем с ленточными и трехдиагональными матрицами. Для этого необходимо перейти на соответствующие страницы "Систематического каталога целевых программ" по адресам

http://num-anal.srcc.msu.ru/par_prog/cat5631.htm
или

http://num-anal.srcc.msu.ru/par_prog/cat5632.htm.

Приведем здесь лишь один пример фортранного фрагмента, реализующего второй способ распределения несимметричной трехдиагональной матрицы A вида

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

с помощью служебной подпрограммы PDELSET.

Как было сказано в п. 2.4.4.1, такая матрица представляется в памяти в виде трех отдельных векторов (D, DL, DU), каждый из которых содержит элементы одной из диагоналей.

Фортранный фрагмент имеет следующий вид

```
T = 2.0D0
M0 = -1.0D0
Z = 0.0D0
```

```
      DO 10 J = 1, N-1
       CALL PDELSET( D, 1, J, DESCD, T )
10 CONTINUE

      DO 30 J = 2, N
      CALL PDELSET( DL, 1, J, DESCD, 1 )
30 CONTINUE

      DO 20 J = 1, N-1
      CALL PDELSET( DU, 1, J, DESCD, MO )
20 CONTINUE

      CALL PDELSET( D, 1, N, DESCD, 1.DO )
```

3.3. О документировании и примерах использования параллельных программ.

3.3.1. Правила наименований подпрограмм.

Каждая подпрограмма комплекса PARALG имеет имя, начинающееся с буквы P. Допущено нарушение стандарта языка Фортран-77, так как имена подпрограмм могут быть длиной более шести символов; кроме того, в именах служебных подпрограмм (tool routines) допускается использование символа подчеркивания “ _ ”.

Все целевые и базовые подпрограммы имеют имена в виде последовательности шести или семи символов: PXYYZZ или PXYYZZZ, где второй символ X может принимать следующие значения, указывающие на тип обрабатываемых данных:

S	REAL
D	DOUBLE PRECISION
C	COMPLEX
Z	DOUBLE COMPLEX

Следующие две буквы YY указывают на то, какого вида матрица обрабатывается подпрограммой (или вид самой главной из участвующих матриц). Большинство из этих двухсимвольных сочетаний относятся одновременно как к вещественным, так и к комплексным матрицам, некоторые же относятся конкретно к одному из этих типов матриц. Приняты следующие мнемонические правила для указания вида обрабатываемых матриц:

DB	(general band)	—	ленточная матрица общего вида, для которой не требуется выбор ведущего элемента
DT	(general tridiagonal)	—	трехдиагональная матрица общего вида, для которой не требуется выбор ведущего элемента
GB	(general band)	—	ленточная матрица общего вида
GE	(general)	—	матрица общего вида (т.е. не симметричная, иногда прямоугольная)
GG	(general matrices, generalized problem)	—	матрица общего вида, обобщенная проблема собственных значений (т.е. пара матриц общего вида)
HE	(Hermitian)	—	эрмитова матрица
OR	(real orthogonal)	—	вещественная ортогональная матрица
PB	(symmetric or Hermitian positive definite band)	—	симметричная или эрмитова ленточная положительно определенная матрица
PO	(symmetric or Hermitian positive definite)	—	симметричная или эрмитова положительно определенная матрица
PT	(symmetric or Hermitian positive definite tridiagonal)	—	симметричная или эрмитова трехдиагональная положительно определенная матрица
ST	(real symmetric tridiagonal)	—	вещественная симметричная трехдиагональная матрица
SY	(symmetric)	—	симметричная матрица
TR	(triangular or in some cases quasi-triangular)	—	трехдиагональная или в некоторых случаях почти трехдиагональная матрица
TZ	(trapezoidal)	—	трапециевидная матрица
UN	(complex unitary)	—	комплексная унитарная матрица

Последние два символа ZZ или три символа ZZZ в имени подпрограммы указывают ее назначение. Например, окончание TRF означает подпрограммы, которые выполняют факторизацию матриц.

Имена вспомогательных подпрограмм подчиняются тем же правилам, за исключением того, что третьим и четвертым символом YY обычно являются символы LA (например, PDLASCL или PZLARFG).

3.3.2. Описания программ и примеры их использования.

Документация параллельных программ по линейной алгебре включает в себя

- информацию общего характера (об организации и структуре комплекса и об общих правилах использования программ и размещения данных);

- систематический каталог целевых программ;

- описания целевых программ (т.е. инструкции по их использованию).

Документация общего характера приводится как в рамках настоящего учебного пособия, так и в соответствующих подразделах раздела "Комплекс программ по линейной алгебре для вычислений на распределенной памяти" Научно-образовательного Интернет-ресурса НИВЦ МГУ по численному анализу (http://num–anal.srcc.msu.ru/). Там же приводится систематический каталог целевых программ в виде таблиц, содержащих, в частности, гиперссылки на описания программ.

Описания программ составлялись во многом по аналогии с тем, как это было ранее разработано и реализовано в Библиотеке фортранных программ по численному анализу НИВЦ МГУ [3], которая состоит из обычных последовательных алгоритмов.

Такая четкая форма документации, состоящая из нескольких обязательных смысловых разделов, заслужила широкое одобрение среди многочисленных пользователей Библиотеки численного анализа, в течение многих лет эксплуатировавших ее в разных организациях и на разных вычислительных системах.

Одним из самых важных разделов в описании каждой целевой программы является его последний раздел под названием "Пример использования". Содержание этого раздела на примере матриц небольшого размера наглядно демонстрирует пользователю, как будут распределены их элементы по параллельным процессам и как такое распределение можно реализовать средствами языка ФОРТРАН и служебных подпрограмм. А именно, приводится фрагмент головной фортранной программы (теста) с подпрограммой рассылки элементов конкретной матрицы по параллельным процессам одним из трех способов, рассматриваемых в настоящем пособии (см. п. 3.2). Указанные подпрограммы рассылки имеют в примерах либо имя MATINIT, либо PDMATINIT.

3.3.3. Обозначения в примерах по использованию программ Комплекса.

В тестовых примерах и примерах по использованию программ комплекса приняты следующие единообразные обозначения величин, задающих параметры исходной задачи и выбранный способ распараллеливания (указаны в алфавитном порядке).

Имя		**Определение**
CSRC	—	Номер столбца решетки процессов, в который распределяется первый столбец исходной глобальной матрицы
DESCA	—	Дескриптор исходной глобальной матрицы A
DESCB	—	Дескриптор исходной глобальной матрицы (вектора) B (или правых частей при решении систем линейных уравнений, или при решении обобщенной проблемы собственных значений)
ICTXT	—	Контекст BLACS'а, связанный с выбранной решеткой процессов (внутренний параметр комплекса; служит для обеспечения согласованной работы подпрограмм комплекса и устанавливается пользователем обращением к соответствующей подпрограмме BLACS'а в начале решения задачи, см. пп. 2.2, 3.1)
M	—	Число строк исходной глобальной матрицы A
MB	—	Число строк в блоке, на которые делится исходная матрица A
MXLLDA	—	Максимальная ведущая локальная размерность локальной части матрицы A среди всех локальных частей на всех процессах решетки
MXLLDB	—	Максимальная ведущая локальная размерность локальной части матрицы (вектора) B среди всех локальных частей на всех процессах решетки
MXLOCC	—	Максимальное число столбцов в локальных частях матрицы A, которые принадлежат процессам в столбцах решетки процессов
MXLOCR	—	Максимальное число строк в локальных частях матрицы A, которые принадлежат процессам в строках решетки процессов
MXRHSC	—	Максимальное число столбцов в локальных частях матрицы (вектора) правых частей B, которые принадлежат процессам в столбцах решетки процессов (если B — вектор, то MXRHSC=1)
MYCOL	—	Координата (номер) столбца вызываемого процесса в решетке процессов

MYROW — Координата (номер) строки вызываемого процесса в решетке процессов

N — Число столбцов исходной глобальной матрицы A (и число строк глобальной матрицы (вектора) правых частей B при решении систем уравнений)

NB — Число столбцов в блоке, на которые делится исходная матрица A

NBRHS — Число столбцов в блоке, на которые делится глобальная матрица (вектор) правых частей системы уравнений B (если B — вектор, то NBRHS=1)

NPCOL — Число столбцов в решетке процессов

NPROW — Число строк в решетке процессов

NRHS — Число столбцов исходной глобальной матрицы (вектора) правых частей системы уравнений B (если B — вектор, то NBRHS=1)

RSRC — Номер строки решетки процессов, в которую распределяется первая строка исходной глобальной матрицы

3.3.4. Упрощения, принятые в примерах по использованию программ.

Опишем далее некоторые упрощения и ограничения, которые были сделаны при подготовке примеров использования подпрограмм комплекса, для того чтобы упростить пользователю освоение правил работы с этими подпрограммами.

1. В примерах предполагается, что RSRC=CSRC=0; это означает, что исходные матрицы A (и B) распределяются по решетке процессов начиная с процесса $(0, 0)$. На самом деле, любой из процессов решетки может быть выбран в качестве начального при распределении матриц.

2. В примерах предполагается, что задача решается при использовании всей исходной распределенной матрицы A (хотя подпрограммы допускают, чтобы задачи решались с матрицей, являющейся подматрицей sub (A) исходной матрицы A). При указанном предположении параметры, задающие первый элемент подматрицы A, полагаются равными 1, т.е. IA=JA=IB=JB=1.

3. При распределении исходных матриц A по решетке процессов используется подпрограмма MATINIT (см. конец Приложения 1), которая заносит с помощью операторов присваивания в соответствующие локальные массивы на каждом из процессов решетки распределенные на него элементы исходных матриц. Упрощает эту процедуру использование служебной подпрограммы

PDELSET (см. п. 3.2 и конец Приложения 2).

4. В примерах предполагается, что ведущая локальная размерность локальных частей матрицы A и ведущая локальная размерность локальных частей матрицы (вектора) B являются теми же самыми для всех процессов, принадлежащих к одним и тем же строкам решетки процессов (т.е. имеющих одинаковые координаты строк решетки). Переменная MXLLDA равна максимальной ведущей локальной размерности для матрицы A (обозначаемой LLD_A) из всех строк решетки процессов. Точно так же переменная MXLLDB равна максимальной локальной ведущей размерности для матрицы B (обозначаемой LLD_B) из всех строк решетки процессов. Однако в общем случае локальные ведущие размерности локальных частей матриц могут отличаться друг от друга для всех процессов решетки.

5. В примерах рассматриваются исходные матрицы A и B небольших размеров. При этом размеры блоков, на которые разбиваются эти матрицы, выбираются небольшими. Например, в тесте к подпрограмме, PDGESV.f, решающей систему уравнений с плотной матрицей, предполагается, что MB=NB=2. Однако это не является самым лучшим размером блоков для практических задач. Для достижения лучшей производительности более подходящим размером блоков будет MB=NB=32 или MB=NB=64.

3.4. Пример использования программы для плотных матриц.

Рассмотрим подробнее пример по использованию подпрограммы PDGESV, которая предназначена для решения линейной системы алгебраических уравнений с плотной матрицей общего вида A. Данный пример приводится в описании этой целевой программы (см. Приложение 1) в последнем разделе описания с названием “Пример использования”.

Пример демонстрирует решение системы уравнений A∗X=B с матрицей A размера 9×9. Ниже показана матричная форма записи этой системы.

$$\begin{pmatrix} 19 & 3 & 1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & 3 & 1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & 1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & -12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & -12 & -1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & -12 & -1 & -16 & 1 & 3 & 11 \\ -19 & 3 & -1 & -12 & -1 & -16 & -1 & 3 & 11 \\ -19 & -3 & -1 & -12 & -1 & -16 & -1 & -3 & 11 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Для простоты решается система с одной правой частью (NRHS=1), т.е. правая часть B системы — вектор (матрица из одного столбца).

Сначала мы должны задать решетку процессов и распределить по ней исходные глобальные матрицы A и B.

Предположим, что матрица A разделена на блоки размером MB×NB, где MB=NB=2. Предположим также, что выбрана решетка процессов размером 2×3, т.е. NPROW=2 и NPCOL=3. Таким образом, имеет место случай распределения матрицы A, приведенный в п. 2.4.2. После подстановки конкретных значений элементов матрицы A получаем следующую картину распределения ее элементов по процессам (вверху указаны номера столбцов решетки процессов, слева — номера строк решетки процессов).

	0				1			2	
0	19	3	1	3	1	12	11	1	16
	−19	3	1	3	1	12	11	1	16
	−19	−3	1	3	−1	−12	11	1	16
	−19	−3	1	3	−1	−12	11	−1	16
	−19	−3	−1	−3	−1	−12	11	−1	−16
1	−19	−3	1	3	1	12	11	1	16
	−19	−3	1	3	−1	12	11	1	16
	−19	−3	1	3	−1	−12	11	−1	−16
	−19	3	−1	3	−1	−12	11	−1	−16

Из последнего рисунка видно, что процесс с координатами $(0, 0)$ содержит локальный массив (часть матрицы A) размера $(5, 4)$.

На следующем рисунке показано разбиение на блоки и распределение по той же решетке процессов исходного вектора правых частей B.

	0	1	2
0	b_1=0		
	b_2=0		
	b_5=0		
	b_6=0		
	b_9=0		
1	b_3=1		
	b_4=0		
	b_7=0		
	b_8=0		

Как видим, блоки состоят из двух соседних элементов вектора B и распределяются только в столбце решетки процессов с координатой 0. Процессы в других столбцах решетки не содержат никаких частей исходного глобального вектора B.

После вызова подпрограммы PDGESV и выполнения всех необходимых вычислений процесс $(0,0)$ будет содержать на месте локальной части вектора B локальную часть глобального вектора решений X. Ниже изображено, на месте каких элементов локального вектора B (обозначенных как $\tilde{}b_i$) с локальными индексами оказываются расположенными элементы глобального вектора X с глобальными индексами по окончании вычислений. Это расположение соответствует расположению элементов исходного глобального вектора правых частей B, показанному на предыдущем рисунке. Ниже показаны также вычисленные значения элементов вектора X:

x_1	$\rightarrow$	$\tilde{}b_1$		0
x_2	$\rightarrow$	$\tilde{}b_2$		$-1/6$
x_5	$\rightarrow$	$\tilde{}b_3$	$=$	0
x_6	$\rightarrow$	$\tilde{}b_4$		0
x_9	$\rightarrow$	$\tilde{}b_5$		0

Аналогичное соответствие и полученные результаты изображены далее для процесса с координатами $(1,0)$:

x_3	$\rightarrow$	$\tilde{}b_1$		$1/2$
x_4	$\rightarrow$	$\tilde{}b_2$		0
x_7	$\rightarrow$	$\tilde{}b_3$	$=$	$-1/2$
x_8	$\rightarrow$	$\tilde{}b_4$		$1/6$

Таким образом, глобальный выходной вектор решений X имеет следующий вид:

x_1		0
x_2		$-1/6$
x_3		$1/2$
x_4		0
x_5	$=$	0
x_6		0
x_7		$-1/2$
x_8		$1/6$
x_9		0

Кроме того, выполняется проверка точности полученных результатов вычислением нормализованной невязки по формуле

$$\frac{||\mathrm{A} * x - b||}{\big(||x|| * ||\mathrm{A}|| * \mathrm{eps} * \mathrm{N}\big)}.$$

Здесь eps означает машинное эпсилон, которое вычисляется с помощью вспомогательной подпрограммы PDLAMCH.

Как уже говорилось, фрагмент фортранного текста с реализацией этого примера приведен в разделе “Пример использования” в Приложении 1. В этом примере показан первый из рассмотренных в п. 3.2 способов программирования на ФОРТРАНе процесса распределения исходных матриц в локальную память каждого из параллельных процессов.

Приведем здесь фортранные операторы второго, более простого способа реализации такого же распределения исходных матриц с помощью служебной подпрограммы PDELSET (преимущества которого также рассматривались в п. 3.2):

```
      AIJ = 19.0D0

      DO 20 J = 1, N
         DO 10 I = 1, N
            IF( I.LE.J ) THEN
                CALL PDELSET( A, I, J, DESCA, AIJ )
            ELSE
                CALL PDELSET( A, I, J ,DESCA, -AIJ )
            END IF
 10   CONTINUE
          IF( J.EQ.1 .OR. J.EQ.7 ) AIJ = 3.0D0
          IF( J.EQ.2 .OR. J.EQ.4 .OR. J.EQ.6 ) AIJ = 1.0D0
          IF( J.EQ.3 ) AIJ = 12.0D0
          IF( J.EQ.5 ) AIJ = 16.0D0
          IF( J.EQ.8 ) AIJ = 11.0D0
 20   CONTINUE

      CALL PDELSET( A, 8, 2, DESCA, 3.0D0 )

      BIJ = 0.0D0

      J = 1
      DO 30 I = 1, N
        CALL PDELSET( B, I, J, DESCB, BIJ )
 30   CONTINUE

      CALL PDELSET( B, 3, 1, DESCB, 1.0D0 )
```

Полностью весь фортранный текст рассматриваемого примера приведен в Приложении 2.

Кроме того, в Приложении 3 приводится текст программы при использовании третьего (из рассмотренных в п. 3.2) способа распределения матриц при считывании исходной матрицы из внешнего файла.

3.5. Общий список программ комплекса PARALG.

В п. 2.5 настоящего учебного пособия перечислены алгоритмы, используемые в комплекс PARALG при решении систем линейных алгебраических уравнений.

Здесь мы приведем полный список всех целевых параллельных программ комплекса, а также два списка основных базовых подпрограмм, которые выполняют два, описанных в п. 2.5, этапа решения систем.

Список целевых программ

1. Подраздел “Решение линейных систем с матрицами общего вида”.

PDGESV, PZGESV	—	решение системы $A * X = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу;
PDGESV1, PZGESV1	—	решение системы $A * X = B$ или $A^T * X = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу;
PDGESV2, PZGESV2	—	решение системы $A * X = B$ или $A^T * X = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу и оценка обратного числа обусловленности;
PDGESV3, PZGESV3	—	решение системы $A * X = B$ или $A^T * X = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу и с итерационным уточнением решения и оценкой границ ошибок.

2. Подраздел “Решение линейных систем с симметричными или эрмитовыми матрицами”.

PDPOSV, PZPOSV — решение системы с симметричной (эрмитовой) положительно определенной матрицей методом квадратного корня (методом Холецкого);

PDPOSV1, PZPOSV1 — решение системы с симметричной (эрмитовой) положительно определенной матрицей методом Холецкого и оценка обратного числа обусловленности;

PDPOSV2, PZPOSV2 — Решение системы с симметричной (эрмитовой) положительно определенной матрицей методом Холецкого с итерационным уточнением решения и оценкой границ ошибок.

3. Подраздел “Решение линейных систем с матрицами специального вида”.

PDDBSV, PZDBSV — решение системы с ленточной матрицей общего вида без выбора ведущих элементов;

PDDTSV, PZDTSV — решение системы с трехдиагональной матрицей общего вида без выбора ведущего элемента;

PDGBSV, PZGBSV — решение системы линейных уравнений с ленточной матрицей общего вида с выбором ведущего элемента;

PDPBSV, PZPBSV — решение системы с симметричной (эрмитовой) положительно определенной ленточной матрицей методом Холецкого;

PDPTSV, PZPTSV — решение системы с симметричной (эрмитовой) положительно определенной трехдиагональной матрицей.

Список основных базовых подпрограмм

1. Базовые подпрограммы, выполняющие факторизацию матрицы системы:

- для плотных (вещественных или комплексных) матриц общего вида и симметричных (или эрмитовых) положительно определенных — PDGETRF (PZGETRF) и PDPOTRF (PZPOTRF);

- для ленточных матриц общего вида (вещественных или комплексных) — PDGBTRF (PZGBTRF) (с выбором ведущего элемента по столбцу) и PDDBTRF (PZDBTRF) (без выбора ведущего элемента);

- для симметричных/эрмитовых ленточных положительно определенных матриц — PDPBTRF (PZPBTRF);

- для трехдиагональных матриц общего вида (вещественных или комплексных) — PDDTTRF (PZDTTRF) (без выбора ведущего элемента);
- для симметричных/эрмитовых трехдиагональных положительно определенных матриц — PDPTTRF (PZPTTRF).

2. Базовые подпрограммы, выполняющие решение систем с использованием полученных на первом этапе факторизаций матриц:

- для плотных (вещественных или комплексных) матриц общего вида и симметричных (или эрмитовых) положительно определенных — PDGETRS (PZGETRS) и PDPOTRS (PZPOTRS);
- для ленточных матриц общего вида (вещественных или комплексных) — PDGBTRS (PZGBTRS) и PDDBTRS (PZDBTRS);
- для симметричных/эрмитовых ленточных положительно определенных матриц — PDPBTRS (PZPBTRS);
- для трехдиагональных матриц общего вида (вещественных или комплексных) — PDDTTRS (PZDTTRS);
- для симметричных/эрмитовых трехдиагональных положительно определенных матриц — PDPTTRS (PZPTTRS).

3.6. Запуск программ комплекса в ОС Linux на суперкомпьютере "Чебышев".

Комплекс программ PARLAG доступен пользователям суперкомпьютера СКИФ "Чебышев" в НИВЦ МГУ в виде откомпилированных библиотек. Пакеты (BLAS, BLACS, ScaLAPACK и др.), модули которых используются программами комплекса, входят в состав MKL-библиотеки (Intel Math Kernel Library), установленной на этом суперкомпьютере, и должны быть подсоединены при получении исполнимой программы. Объектные модули самого комплекса берутся из библиотеки libparalg.a, указанной в последней строке приводимого ниже скрипта **bld-tcl**, который предназначен для компиляции и получения исполнимой программы для ее последующего запуска на счет в параллельном режиме на суперкомпьютере "Чебышев".

```
#! /bin/sh
EXEDIR=.
EXE="<имя исполнимой программы>"
OBJ="<список имен объектных модулей>"
F77OPTS="-O2"
```

```
F77=mpif77
#
$F77 -c $F77OPTS <имя исходного фортранного модуля>
#
echo Linking executable $EXE
#
$F77 -o $EXEDIR/$EXE  $OBJ \
       -Wl,--start-group -lmkl_scalapack_lp64 -lmkl_blacs_lp64 \
       -lmkl_intel_lp64 \
       -lmkl_sequential -lmkl_core \
       -Wl,--end-group \
       -lpthread -libparalg.a
```

Здесь:

- <имя исполнимой программы> — указываемое пользователем имя, которое будет присвоено полученному исполнимому модулю (например, tcl_gesv.e);
- <список имен объектных модулей> — список имен всех объектных модулей, которые не вызываются из библиотек (например, tcl_gesv.o);
- $F77 -c $F77OPTS <имя исходного фортранного модуля> — команда компиляции (которых может быть несколько) фортранного модуля (например, $F77 -c $F77OPTS tcl_gesv.f).

В скрипте **bld-tcl** подключаются все пакеты из MKL-библиотеки, содержащие все вызываемые подпрограммы, используемые при работе параллельных подпрограмм комплекса.

Полученная исполнимая программа может быть затем запущена на счет с помощью команды **mpirun** с параметрами.

Например, если требуется запустить программу tcl_gesv.e на шести процессорах, то команда запуска имеет вид

mpirun -np 6 -maxtime 1 ./tcl_gesv.e

В тестах к программам выдача исходных матриц и полученных результатов осуществляется с помощью обращения к подпрограмме PDLAPRNT из пакета ScaLAPACK(TOOLS). Эта подпрограмма распечатывает всю распределенную матрицу (вектор) полностью по столбцам по одному элементу в строке, получая элементы со всех участвующих в вычислениях процессоров.

Обращение к программам комплекса может осуществляться с помощью подпрограмм автоматизации доступа к ним. Описание этих подпрограмм доступно по адресу

http://num-anal.srcc.msu.ru/par_prog/org/avt.htm

Тогда конкретный вид скрипта может быть проиллюстрирован следующим примером для запуска на счет программы PDGESV1:

```
#! /bin/sh
EXEDIR=.
EXE="tcl_gesv.e"
OBJ="tcl_gesv.o  cal_gesv.o  pdgesv1.o  pdlaprnt.o  pdlaread.o  pdlawrite.o"
#
F77OPTS="-O2"
F77=mpif77
#
$F77 -c  $F77OPTS  tcl_gesv.f
#
$F77 -o $EXEDIR/$EXE  $OBJ \
       -Wl,--start-group -lmkl_scalapack_lp64 -lmkl_blacs_lp64 \
       -lmkl_intel_lp64 \
       -lmkl_sequential -lmkl_core \
       -Wl,--end-group \
       -lpthread -libparalg.a
```

Чтобы не указывать все объектные модули подпрограмм комплекса PARALG, которые необходимо вызвать для решения конкретной задачи, достаточно указать в последней строке скрипта объектную библиотеку libparalg.a. Тогда строка с объектными модулями будет содержать только объектный модуль головной программы
OBJ=“tcl_gesv.o”.

Здесь подпрограммы pdlaread.o и pdlawrite.o — входящие в состав комплекса служебные подпрограммы для считывания из файла исходных матриц и распределения их по параллельным процессам и записи в файл полученного вектора решения.

Приложение 1

Описание подпрограммы решения линейной системы с матрицей общего вида

Подпрограмма: PDGESV

Назначение

Решение линейной системы с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу

Математическое описание

Подпрограмма вычисляет решение вещественной системы линейных уравнений

$$\mathrm{sub(A)} * \mathrm{X} = \mathrm{sub(B)},$$

где

- sub(A) = A(IA : IA+N-1, JA : JA+N-1) — квадратная распределенная матрица порядка N,
- X — распределенная матрица размеров N на NRHS,
- sub(B) = B(IB : IB+N-1, JB : JB+NRHS-1) — распределенная матрица размеров N на NRHS.

Используется LU-разложение методом Гаусса с выбором ведущего элемента по столбцу для факторизации матрицы sub(A) в виде

$$\mathrm{sub(A)} = \mathrm{P} * \mathrm{L} * \mathrm{U},$$

где

- P — матрица перестановок,
- L — нижняя треугольная матрица с единичными диагональными элементами,
- U — верхняя треугольная матрица.

Матрицы L и U помещаются в памяти на место матрицы sub(A). Полученное LU-разложение используется для решения системы $\mathrm{sub(A)} * \mathrm{X} = \mathrm{sub(B)}$.
Матрицы A и B должны быть заданы в виде массивов двойной точности. Все вычисления проводятся в режиме DOUBLE PRECISION.

Литература:

http://num-anal.srcc.msu.ru/par_prog/comparp.htm

http://www.netlib.org/scalapack/slug/index.html

Использование

CALL PDGESV (N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB, INFO)

Параметры

N — порядок распределенной подматрицы sub(A) (глобальный входной параметр, тип целый);

NRHS — число правых частей, т.е. число столбцов распределенной подматрицы sub(B) (глобальный входной параметр, тип целый);

A — указатель на локальную память, занимаемую массивом двойной точности размерности (LLD_A, LOCc(JA+N-1));
на входе — это локальные части факторизуемой распределенной матрицы sub(A) порядка N;
на выходе — массив A содержит локальные части множителей L и U, полученные при разложении (факторизации) подматрицы в виде sub(A) = P ∗ L ∗ U; единичные диагональные элементы матрицы L не хранятся в памяти (локальный входной и локальный выходной параметр);

IA — номер строки в глобальном массиве A, указывающий на первую строку подматрицы sub(A) (глобальный входной параметр, тип целый);

JA — номер столбца в глобальном массиве A, указывающий на первый столбец подматрицы sub(A) (глобальный входной параметр, тип целый);

DESCA — дескриптор распределенной матрицы A (одномерный массив длины DLEN_); подробное описание см. в п. 2.3.2 (глобальный и локальный входной параметр, тип целый);

IPIV — одномерный массив длины (LOCr(M_A)+MB_A), содержащий информацию о выборе ведущего элемента на каждом шаге исключения Гаусса;
элемент массива IPIV(i) содержит глобальный номер строки, которая переставляется со строкой, имеющей локальный номер i;
этот массив содержит информацию о перестановках строк распределенной матрицы A (локальный выходной параметр, тип целый);

B — указатель на локальную память, занимаемую массивом двойной точности размерности (LLD_B, LOCc(JB+NRHS-1));
на входе — это распределенная матрица sub(B) правой части;
на выходе (если INFO=0) на месте sub(B) находится распределенная матрица X, столбцы которой образуют NRHS векторов решений исходной системы (локальный входной и локальный выходной параметр);

IB — номер строки в глобальном массиве B, указывающий на первую строку подматрицы sub(B) (глобальный входной параметр, тип целый);

JB — номер столбца в глобальном массиве B, указывающий на первый столбец подматрицы sub(B) (глобальный входной параметр, тип целый);

DESCB — дескриптор распределенной матрицы B (одномерный массив длины DLEN_); подробное описание см. в п. 2.3.2.2 (глобальный и локальный входной параметр, тип целый);

INFO — целая переменная, диагностирующая результат работы подпрограммы (глобальный выходной параметр)
= 0 — успешное завершение работы;
< 0 — если i-й фактический параметр является массивом и его j-й элемент имеет недопустимое значение,
то INFO = −(i ∗ 100 + j),
если i-й фактический параметр является скаляром и имеет недопустимое значение,то INFO=-i;
> 0 — если INFO=K, то элемент U(IA+K-1, JA+K-1) является нулем; разложение завершено, но матрица U вырождена, и поэтому решение системы не вычисляется

Версии

PSGESV, PCGESV, PZGESV — решение системы с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу для вещественных данных одинарной точности, комплексных данных одинарной точности и комплексных данных двойной точности соответственно

Вызываемые подпрограммы

Здесь указаны только базовые подпрограммы (2-го уровня), которые вызываются из целевой подпрограммы (1-го уровня).

PDGETRF, PZGETRF	—	LU-разложение матрицы общего вида методом Гаусса с выбором ведущего элемента по столбцу
PDGETRS, PZGETRS	—	Решение системы AX=B, $A^TX = B$ или $A^HX = B$ на основе LU-разложения, полученного подпрограммой PDGETRF(PZGETRF)

Замечания по использованию

1. В подпрограммах PSGESV, PCGESV, PZGESV параметры A и B имеют тип REAL, COMPLEX и DOUBLE COMPLEX соответственно

2. Используются подпрограммы BLACS_GRIDINFO (из пакета BLACS), PXERBA (из пакета PBLAS), CHK1MAT, PCHK2MAT, INDXG2P (из библиотеки ScaLAPACK_TOOLS)

Пример использования

Решение системы AX=B , где A — матрица порядка 9.
Пусть матрица A системы имеет вид

19	3	1	12	1	16	1	3	11
-19	3	1	12	1	16	1	3	11
-19	-3	1	12	1	16	1	3	11
-19	-3	-1	12	1	16	1	3	11
-19	-3	-1	-12	1	16	1	3	11
-19	-3	-1	-12	-1	16	1	3	11
-19	-3	-1	-12	-1	-16	1	3	11
-19	3	-1	-12	-1	-16	-1	3	11
-19	-3	-1	-12	-1	-16	-1	-3	11

Вектор правых частей B имеет вид:

0
0
1
0
0
0
0
0
0

Матрица A разбивается на блоки размеров 2 на 2. Осуществляется запуск программы на 6 процессах и используется двумерная решетка процессов 2 на 3. Ниже показано, как разбитая на блоки матрица A распределяется (блочно-циклически) по решетке процессов 2×3.

	0				1			2	
0	19	3	1	3	1	12	11	1	16
	-19	3	1	3	1	12	11	1	16
	19	-3	1	3	-1	-12	11	1	16
	-19	-3	1	3	-1	-12	11	-1	16
	-19	-3	-1	-3	-1	-12	11	-1	-16
1	-19	-3	1	3	1	12	11	1	16
	-19	-3	1	3	-1	12	11	1	16
	-19	-3	1	3	-1	-12	11	-1	-16
	-19	3	-1	3	-1	-12	11	-1	-16

Распределение вектора B (блочно-циклическое) по процессам

	0	1	2
0	0		
	0		
	0		
	0		
	0		
1	1		
	0		
	0		
	0		

Фрагмент фортранного текста вызывающей программы (принятые в тестах обозначения приведены в п. 3.3.3)

```
 PROGRAM TPDGESV
 include 'mpif.h'

 INTEGER     DLEN_, IA, JA, IB, JB, M, N, MB, NB, RSRC,
$                    CSRC, MXLLDA, MXLLDB, NRHS, NBRHS, NOUT,
$                    MXLOCR, MXLOCC, MXRHSC
 PARAMETER   ( DLEN_ = 9, IA = 1, JA = 1, IB = 1, JB = 1,
$                         M = 9, N = 9, MB = 2, NB = 2, RSRC = 0,
$                         CSRC = 0, MXLLDA = 5, MXLLDB = 5, NRHS = 1,
$                         NBRHS = 1, NOUT = 6, MXLOCR = 5, MXLOCC = 4,
$                         MXRHSC = 1 )
 DOUBLE PRECISION   ONE
 PARAMETER   ( ONE = 1.0D+0 )

 INTEGER   ICTXT, INFO, MYCOL, MYROW, NPCOL, NPROW

 INTEGER   DESCA( DLEN_ ), DESCB( DLEN_ ), IPIV( MXLOCR+NB )
 DOUBLE PRECISION   A( MXLLDA, MXLOCC ), B( MXLLDB, MXRHSC ),
$                                        WORK( MXLOCR )

 EXTERNAL   BLACS_EXIT, BLACS_GRIDEXIT, BLACS_GRIDINFO,
$                       DESCINIT, MATINIT, PDGESV, SL_INIT

 INTRINSIC   DBLE
 DATA   NPROW / 2 / , NPCOL / 3 /

 CALL  SL_INIT( ICTXT, NPROW, NPCOL )
 CALL  BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )

 IF( MYROW.EQ.-1 ) GO TO 10

 CALL  DESCINIT( DESCA, M, N, MB, NB, RSRC, CSRC, ICTXT, MXLLDA,
$                               INFO )
 CALL  DESCINIT( DESCB, N, NRHS, NB, NBRHS, RSRC, CSRC, ICTXT,
$                               MXLLDB, INFO )

 CALL  MATINIT( A, DESCA, B, DESCB )
```

```
      CALL  PDGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB,
     $                            INFO )

      CALL  BLACS_GRIDEXIT( ICTXT )
 10 CONTINUE
      CALL  BLACS_EXIT( 0 )
      STOP
      END

      SUBROUTINE  MATINIT( AA, DESCA, B, DESCB )
* MATINIT генерирует и распределяет матрицы  A и B  по решетке процессов 2 x 3

      INTEGER   DESCA( * ), DESCB( * )
      DOUBLE PRECISION   AA( * ), B( * )

      INTEGER   CTXT_, LLD_
      PARAMETER    ( CTXT_ = 2, LLD_ = 9 )

      INTEGER   ICTXT, MXLLDA, MYCOL, MYROW, NPCOL, NPROW
      DOUBLE PRECISION   A, C, K, L, P, S
      EXTERNAL   BLACS_GRIDINFO

      ICTXT = DESCA( CTXT_ )
      CALL  BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )

      S = 19.0D0
      C = 3.0D0
      A = 1.0D0
      L = 12.0D0
      P = 16.0D0
      K = 11.0D0

      MXLLDA = DESCA( LLD_ )

* Локальные части матриц A и B для процесса (0, 0)

      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         AA( 1 ) = S
         AA( 2 ) = -S
```

```
         AA( 3 ) = -S
         AA( 4 ) = -S
         AA( 5 ) = -S
         AA( 1+MXLLDA ) = C
         AA( 2+MXLLDA ) = C
         AA( 3+MXLLDA ) = -C
         AA( 4+MXLLDA ) = -C
         AA( 5+MXLLDA ) = -C
         AA( 1+2*MXLLDA ) = A
         AA( 2+2*MXLLDA ) = A
         AA( 3+2*MXLLDA ) = A
         AA( 4+2*MXLLDA ) = A
         AA( 5+2*MXLLDA ) = -A
         AA( 1+3*MXLLDA ) = C
         AA( 2+3*MXLLDA ) = C
         AA( 3+3*MXLLDA ) = C
         AA( 4+3*MXLLDA ) = C
         AA( 5+3*MXLLDA ) = -C

         B( 1 ) = 0.0D0
         B( 2 ) = 0.0D0
         B( 3 ) = 0.0D0
         B( 4 ) = 0.0D0
         B( 5 ) = 0.0D0

* Локальная часть матрицы A для процесса (0, 1)

      ELSE IF( MYROW.EQ.0 .AND. MYCOL.EQ.1 ) THEN
         AA( 1 ) = A
         AA( 2 ) = A
         AA( 3 ) = -A
         AA( 4 ) = -A
         AA( 5 ) = -A
         AA( 1+MXLLDA ) = L
         AA( 2+MXLLDA ) = L
         AA( 3+MXLLDA ) = -L
         AA( 4+MXLLDA ) = -L
         AA( 5+MXLLDA ) = -L
         AA( 1+2*MXLLDA ) = K
```

```
         AA( 2+2*MXLLDA ) = K
         AA( 3+2*MXLLDA ) = K
         AA( 4+2*MXLLDA ) = K
         AA( 5+2*MXLLDA ) = K

* Локальная часть матрицы A для процесса (0, 2)

      ELSE IF( MYROW.EQ.0 .AND. MYCOL.EQ.2 ) THEN
         AA( 1 ) = A
         AA( 2 ) = A
         AA( 3 ) = A
         AA( 4 ) = -A
         AA( 5 ) = -A
         AA( 1+MXLLDA ) = P
         AA( 2+MXLLDA ) = P
         AA( 3+MXLLDA ) = P
         AA( 4+MXLLDA ) = P
         AA( 5+MXLLDA ) = -P

* Локальные части матриц A и B для процесса (1, 0)

      ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
         AA( 1 ) = -S
         AA( 2 ) = -S
         AA( 3 ) = -S
         AA( 4 ) = -S
         AA( 1+MXLLDA ) = -C
         AA( 2+MXLLDA ) = -C
         AA( 3+MXLLDA ) = -C
         AA( 4+MXLLDA ) = C
         AA( 1+2*MXLLDA ) = A
         AA( 2+2*MXLLDA ) = A
         AA( 3+2*MXLLDA ) = A
         AA( 4+2*MXLLDA ) = -A
         AA( 1+3*MXLLDA ) = C
         AA( 2+3*MXLLDA ) = C
         AA( 3+3*MXLLDA ) = C
         AA( 4+3*MXLLDA ) = C
```

```
         B( 1 ) = 1.0D0
         B( 2 ) = 0.0D0
         B( 3 ) = 0.0D0
         B( 4 ) = 0.0D0

* Локальная часть матрицы A для процесса (1, 1)

      ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.1 ) THEN
         AA( 1 ) = A
         AA( 2 ) = -A
         AA( 3 ) = -A
         AA( 4 ) = -A
         AA( 1+MXLLDA ) = L
         AA( 2+MXLLDA ) = L
         AA( 3+MXLLDA ) = -L
         AA( 4+MXLLDA ) = -L
         AA( 1+2*MXLLDA ) = K
         AA( 2+2*MXLLDA ) = K
         AA( 3+2*MXLLDA ) = K
         AA( 4+2*MXLLDA ) = K

* Локальная часть матрицы A для процесса (1, 2)

      ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.2 ) THEN
         AA( 1 ) = A
         AA( 2 ) = A
         AA( 3 ) = -A
         AA( 4 ) = -A
         AA( 1+MXLLDA ) = P
         AA( 2+MXLLDA ) = P
         AA( 3+MXLLDA ) = -P
         AA( 4+MXLLDA ) = -P
      END IF
      RETURN
      END

     Результаты:
     Решение системы
```

```
X=(0.D0,-0.16666666666667D0,0.5D0,0.D0,0.D0,0.D0,-0.5D0,0.16666666666667D0,0.D0)

 Значение   INFO  =  0
```

Приложение 2

Головная программа для вызова подпрограммы из Приложения 1

```
      PROGRAM TPDGESV
*
*     Тест к подпрограмме PDGESV
*
      include 'mpif.h'
* Именованные константы - параметры задачи
      INTEGER            DLEN_, IA, JA, IB, JB, M, N, MB, NB, RSRC,
     $                   CSRC, MXLLDA, MXLLDB, NRHS, NBRHS, NOUT,
     $                   MXLOCR, MXLOCC, MXRHSC
      PARAMETER    ( DLEN_ = 9, IA = 1, JA = 1, IB = 1, JB = 1,
     $                   M = 9, N = 9, MB = 2, NB = 2, RSRC = 0,
     $                   CSRC = 0, MXLLDA = 5, MXLLDB = 5, NRHS = 1,
     $                   NBRHS = 1, NOUT = 6, MXLOCR = 5, MXLOCC = 4,
     $                   MXRHSC = 1 )
      DOUBLE PRECISION   ONE
      PARAMETER          (ONE = 1.0D+0)
*
* Локальные переменные
      INTEGER            ICTXT, INFO, MYCOL, MYROW, NPCOL, NPROW
      DOUBLE PRECISION   ANORM, BNORM, EPS, RESID, XNORM
*
* Локальные массивы
      INTEGER                 DESCA( DLEN_ ), DESCB( DLEN_ ),
     $                                 IPIV( MXLOCR+NB )
      DOUBLE PRECISION   A( MXLLDA, MXLOCC ), A0( MXLLDA, MXLOCC ),
     $                   B( MXLLDB, MXRHSC ), B0( MXLLDB, MXRHSC ),
     $                   WORK( MXLOCR ), WORK1( MB )
*
* Внешние функции
      DOUBLE PRECISION   PDLAMCH, PDLANGE
      EXTERNAL           PDLAMCH, PDLANGE
*
* Внешние подпрограммы
      EXTERNAL           BLACS_EXIT, BLACS_GRIDEXIT, BLACS_GRIDINFO,
     $                   DESCINIT, PDMATINIT, PDGEMM, PDGESV, PDLACPY,
```

```
     $                   PDLAPRNT, SL_INIT
*
* Внутренние функции
      INTRINSIC          DBLE
*
*  Операторы DATA
      DATA               NPROW / 2 / , NPCOL / 3 /
*
* Выполняемые операторы
*
* Инициализация решетки процессов
*
      CALL SL_INIT( ICTXT, NPROW, NPCOL )
      CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
*
* Если процесс вне решетки процессов, переход на конец программы
*
      IF( MYROW.EQ.-1 ) GO TO 10
*
* Распределение матриц по решетке процессов
*
* Инициализация дескрипторов для матриц A и B
*
      CALL DESCINIT( DESCA, M, N, MB, NB, RSRC, CSRC, ICTXT, MXLLDA,
     $                            INFO )
      CALL DESCINIT( DESCB, N, NRHS, NB, NBRHS, RSRC, CSRC, ICTXT,
     $                            MXLLDB, INFO )
*
* Генерация матриц A и B и распределение их по решетке процессов
*
      CALL PDMATINIT( N, A, IA, JA, DESCA, B, IB, JB, DESCB, INFO )
*
* Копирование A и B для проверки невязки
*
      CALL PDLACPY( 'All', N, N, A, 1, 1, DESCA, A0, 1, 1, DESCA )
      CALL PDLACPY( 'All', N, NRHS, B, 1, 1, DESCB, B0, 1, 1, DESCB )
*
* Печать входных данных
*
```

```
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         WRITE( NOUT, FMT = 99 )
         WRITE( NOUT, FMT = 98 )M, N, MB, NB
         WRITE( NOUT, FMT = 97 )NPROW*NPCOL, NPROW, NPCOL
         WRITE( NOUT, FMT = * ) ' Исходные данные '
         WRITE( NOUT, FMT = * ) '  DESCA'
         WRITE( NOUT, FMT = 85 )DESCA
         WRITE( NOUT, FMT = * ) '  DESCB'
         WRITE( NOUT, FMT = 85 )DESCB
         WRITE( NOUT, FMT = * ) 'Матрица A'
      END IF
 85 FORMAT( / 9I5 /)
*
      CALL PDLAPRNT( M, N, A, IA, JA, DESCA, 0, 0, 'A', NOUT, WORK1 )
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
      WRITE( NOUT, FMT = * ) 'Матрица B'
      END IF
      CALL PDLAPRNT( N, NRHS, B, IB, JB, DESCB, 0, 0, 'B', NOUT, WORK1 )
*
* Вызов подпрограмм комплекса
*
* Решение линейной системы A * X = B
*
      CALL PDGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB,
     $                           INFO )
*
* Печать результатов X
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         WRITE( NOUT, FMT = * ) ' Результаты '
         WRITE( NOUT, FMT = 96 )INFO
         WRITE( NOUT, FMT = * ) 'Решение X'
      END IF
*
      CALL PDLAPRNT( N, NRHS, B, IB, JB, DESCB, 0, 0, 'X', NOUT, WORK1 )
*
* Вычисление невязки ||A * X  - B|| / ( ||X|| * ||A|| * eps * N )
*
```

```
      EPS = PDLAMCH( ICTXT, 'Epsilon' )
      ANORM = PDLANGE( 'I', N, N, A, 1, 1, DESCA, WORK )
      BNORM = PDLANGE( 'I', N, NRHS, B, 1, 1, DESCB, WORK )
      CALL PDGEMM( 'N', 'N', N, NRHS, N, ONE, A0, 1, 1, DESCA, B, 1, 1,
     $                               DESCB, -ONE, B0, 1, 1, DESCB )
      XNORM = PDLANGE( 'I', N, NRHS, B0, 1, 1, DESCB, WORK )
      RESID = XNORM / ( ANORM*BNORM*EPS*DBLE( N ) )
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         IF( RESID.LT.10.0D+0 ) THEN
            WRITE( NOUT, FMT = 95 )
            WRITE( NOUT, FMT = 93 )RESID
         ELSE
            WRITE( NOUT, FMT = 94 )
            WRITE( NOUT, FMT = 93 )RESID
         END IF
      END IF
*
* Освобождение решетки процессов
*
* Освобождение контекста BLACS'а
*
      CALL BLACS_GRIDEXIT( ICTXT )
 10 CONTINUE
*
* Выход из BLACS
*
      CALL BLACS_EXIT( 0 )
*
 99 FORMAT( / ' Тест к подпрограмме PDGESV' )
 98 FORMAT(/' Решение A X = B ,'//' где A - матрица ',I2,' на',I2,',',
    $       /'      разбитая на блоки', I2, ' на ', I2, /)
 97 FORMAT( ' Пропуск на ', I2, ' процессах,',
    $       ' образующих решетку размером ', I2, ' на ', I2 /)
 96 FORMAT( / ' Значение INFO  = ', I6 /)
 95 FORMAT( /' Невязка мала')
 94 FORMAT( /' Невязка велика')
 93 FORMAT( / '||A*x - b|| / ( ||x||*||A||*eps*N ) = ', 1P, E16.8 )
      STOP
```

```
      END
*
      SUBROUTINE PDMATINIT( N, A, IA, JA, DESCA, B, IB, JB, DESCB,
     $                                                  INFO )
*
* PDMATINIT генерирует и распределяет матрицы A и B по решетке процессов 2 x 3
*
      INTEGER            IA, IB, INFO, JA, JB, N
*
      INTEGER                           DESCA( * ), DESCB( * )
      DOUBLE PRECISION   A( * ), B( * )
*
      INTEGER                      CTXT_
      PARAMETER            ( CTXT_ = 2 )
*
      INTEGER                           ICTXT, I, J, MYCOL, MYROW, NPCOL, NPROW
      DOUBLE PRECISION   AIJ
      EXTERNAL           BLACS_GRIDINFO, PDELSET
*
      INFO = 0
      ICTXT = DESCA( CTXT_ )
      CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
*
      IF( IA.NE.1 ) THEN
         INFO = -3
      ELSE IF(JA .NE.1) THEN
         INFO = -4
      END IF
*
      AIJ = 19.0D0
*
      DO 20 J = 1, N
         DO 10 I = 1, N
            IF( I.LE.J ) THEN
                CALL PDELSET( A, I, J, DESCA, AIJ )
            ELSE
                CALL PDELSET( A, I, J ,DESCA, -AIJ )
            END IF
 10 CONTINUE
```

```
         IF( J.EQ.1 .OR. J.EQ.7 ) AIJ = 3.0D0
         IF( J.EQ.2 .OR. J.EQ.4 .OR. J.EQ.6 ) AIJ = 1.0D0
         IF( J.EQ.3 ) AIJ = 12.0D0
         IF( J.EQ.5 ) AIJ = 16.0D0
         IF( J.EQ.8 ) AIJ = 11.0D0
 20 CONTINUE
*
      CALL PDELSET( A, 8, 2, DESCA, 3.0D0 )
*
      BIJ = 0.0D0
      J = 1
      DO 30 I = 1, N
        CALL PDELSET( B, I, J, DESCB, BIJ )
 30 CONTINUE
*
      CALL PDELSET( B, 3, 1, DESCB, 1.0D0 )
      RETURN
      END
```

Приложение 3

Головная программа для чтения матриц из файла и записи результатов в файл

Содержимое файлов с исходными матрицами приведено следом за головной программой.

```
      PROGRAM TDGESV25
*
*     Тест к подпрограмме   PDGESV
*
      include 'mpif.h'
* Именованные константы - параметры задачи
      INTEGER               NOUT, MEMSIZE, NIN, NPR
      PARAMETER          ( MEMSIZE = 20000,
     $                              NOUT =13, NIN = 11, NPR = 6 )
*
* Локальные переменные
      INTEGER            CTXT, I, IAM, INFO, IPA, IPB, IPW,
     $                          IPPIV, MYCOL, MYROW, NP, NQ, NQRHS, NRHS,
     $                          N, NB, NPCOL, NPROCS, NPROW, IA, JA, IB, JB,
     $                          WORKSIZ
*
* Локальные массивы
      INTEGER                        DESCA( 9 ), DESCB( 9 ), DESCX( 9)
      DOUBLE PRECISION   MEM( MEMSIZE )
*
* Внешние функции
      INTEGER                        ICEIL, NUMROC
      DOUBLE PRECISION   PDLAMCH, PDLANGE
      EXTERNAL                   ICEIL, NUMROC, PDLAMCH, PDLANGE
*
* Внешние подпрограммы
      EXTERNAL           BLACS_EXIT, BLACS_GET, BLACS_GRIDEXIT,
     $                            BLACS_GRIDINFO, BLACS_GRIDINIT, BLACS_PINFO,
     $                            BLACS_SETUP, DESCINIT, PDLAPRNT, PDLAREAD,
     $                            PDLAWRITE, PDGESV
*
* Выполняемые операторы
*
```

```
* Постановка задачи
*
      N = 9
      NB = 2
      NPROW = 2
      NPCOL = 2
      IA = 1
      JA = 1
      IB = 1
      JB = 1
      NRHS = 1
*
* Инициализация пакета BLACS
*
      CALL BLACS_PINFO( IAM, NPROCS )
*
        IF( NPROCS.LT.1 ) THEN
           CALL BLACS_SETUP( IAM, NPROW*NPCOL )
        END IF
*
* Инициализация контекста BLACS'а и решетки процессов
*
      CALL BLACS_GET( -1, 0, CTXT )
      CALL BLACS_GRIDINIT( CTXT, 'R', NPROW, NPCOL )
      CALL BLACS_GRIDINFO( CTXT, NPROW, NPCOL, MYROW, MYCOL )
*
* Если процесс не является частью данного контекста,
* переход на конец программы
*
      IF( MYROW.GE.NPROW .OR. MYCOL.GE.NPCOL ) GO TO 20
*
* Вычисление размеров локальных частей матриц A и B на всех процессах
*
      NP = NUMROC( N, NB, MYROW, 0, NPROW )
      NQ = NUMROC( N, NB, MYCOL, 0, NPCOL )
      NQRHS = NUMROC( NRHS, NB, MYCOL, 0, NPCOL )
*
* Инициализация дескрипторов для матриц A, B и X
*
```

```
      CALL DESCINIT( DESCA, N, N, NB, NB, 0, 0, CTXT, MAX(1,NP), INFO)
      CALL DESCINIT(DESCB, N, NRHS, NB,NB, 0,0, CTXT, MAX(1,NP), INFO)
      CALL DESCINIT(DESCX, N, NRHS, NB,NB, 0,0, CTXT, MAX(1,NP), INFO)
*
* Распределение памяти  MEM
*
* Указатель на первый элемент локальной части матрицы  A
      IPA = 1
* Указатель на первый элемент локальной части матрицы(вектора) B
      IPB = IPA + DESCA( 9 )*NQ
* Указатель на первый элемент массива перестановок строк матрицы A
      IPPIV = IPB + DESCB( 9 )*NQRHS
      LIPIV = ICEIL( 4*(NP+NB), 8 )
* Указатель на рабочий массив
      IPW = IPPIV + MAX( NP, LIPIV )
*
* Проверка величины рабочей памяти
*
      WORKSIZ = MAX( NB, NP )
*
      INFO = 0
      IF(IPW + WORKSIZ .GT. MEMSIZE ) THEN
         IF( IAM .EQ. 0 ) WRITE( NOUT, FMT = * )' test'
         INFO = 1
      END IF
*
      CALL IGSUM2D( ICTXT, 'All', ' ', 1, 1, INFO, 1, -1, 0 )
      IF( INFO .GT. 0 ) THEN
         IF( IAM .EQ. 0 )
     $       WRITE( NOUT, FMT = * ) 'BAD MEMORY'
         GO TO 10
      END IF
*
* Чтение из файлов и распределение матриц A и B
*
      CALL PDLAREAD('tdgesv2A.dat', MEM(IPA), DESCA, 0, 0, MEM(IPW))
      CALL PDLAREAD('tdgesv2B.dat', MEM(IPB), DESCB, 0, 0, MEM(IPW))
*
* Печать входных данных
```

```
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         WRITE( NPR, FMT = 92 )
         WRITE( NPR, FMT = 98 )N, N, NB, NB
         WRITE( NPR, FMT = 97 )NPROW*NPCOL, NPROW, NPCOL
         WRITE( NPR, FMT = * ) ' Исходные данные '
         WRITE( NPR, FMT = * )'   IPA,   IPB,   IPPIV,   IPW'
         WRITE( NPR, FMT = 94 )IPA, IPB, IPPIV, IPW
         WRITE( NPR, FMT = * ) '  DESCA'
         WRITE( NPR, FMT = 85 )DESCA
         WRITE( NPR, FMT = * ) '  DESCB'
         WRITE( NPR, FMT = 85 )DESCB
         WRITE( NPR, FMT = * ) '  DESCX'
         WRITE( NPR, FMT = 85 )DESCX
         WRITE( NPR, FMT = * ) ' Матрица A'
      END IF

  94 FORMAT( / 4I7 /)
  85 FORMAT( / 9I5 /)

      CALL PDLAPRNT( N, N, MEM(IPA), IA, JA, DESCA, 0, 0, 'A',
     $                                NPR, MEM( IPW ) )
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         WRITE( NPR, FMT = * ) ' Матрица B'
      END IF
*
      CALL PDLAPRNT( N, NRHS, MEM(IPB), IB, JB, DESCB, 0, 0, 'B',
     $                                NPR, MEM( IPW ) )
*
* Решение линейной системы A * X = B
*
      CALL PDGESV( N, NRHS, MEM(IPA), IA, JA, DESCA, MEM(IPPIV),
     $                          MEM(IPB), IB, JB, DESCB, INFO )
*
* Печать результатов X
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         WRITE( NPR, FMT = * ) ' Результаты '
```

```
         WRITE( NPR, FMT = 96 )INFO
         WRITE( NPR, FMT = * ) ' Вектор X'
      END IF
*
      CALL PDLAPRNT( N, NRHS, MEM(IPB), IB, JB, DESCB, 0, 0, 'X', NPR,
     $                                MEM( IPW ) )
*
* Запись в файл результатов работы подпрограммы PDGESV.f
*
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
         OPEN( NOUT, FILE = 'tdgesv2X.res', STATUS = 'UNKNOWN' )
         WRITE( NOUT, FMT = * ) ' Решение линейной системы A * X = B'
      END IF
*
      CALL PDLAWRITE( 'tdgesv2X.res', N, NRHS, MEM(IPB), 1, 1, DESCB,
     $                                  0, 0, MEM(IPW) )

 10 CONTINUE

      CALL BLACS_GRIDEXIT( CTXT )
 20 CONTINUE

      IF( IAM .EQ. 0 ) THEN
         IF( NOUT .NE. 6 .AND. NOUT .NE. 0 ) CLOSE( NOUT )
      END IF
*
      CALL BLACS_EXIT( 0 )
*
  92 FORMAT( / 'Тест к подпрограмме PDGESV' )
  98 FORMAT(/' Решение A X = B ,'//' где A - матрица ',I2,' на',I2,',',
     $       /'      разбитая на блоки', I2, ' на ', I2, /)
  97 FORMAT( / ' Запуск на ', I2, ' процессах,',
     $       ' образующих решетку размером ', I2, ' на ', I2 /)
  96 FORMAT( / ' Значение INFO  = ', I6 /)
      STOP
      END
```

Файл с исходной матрицей A:

```
    9     9
 0.19000D+02
-0.19000D+02
-0.19000D+02
-0.19000D+02
-0.19000D+02
-0.19000D+02
-0.19000D+02
-0.19000D+02
-0.19000D+02

 0.30000D+01
 0.30000D+01
-0.30000D+01
-0.30000D+01
-0.30000D+01
-0.30000D+01
-0.30000D+01
 0.30000D+01
-0.30000D+01

 0.10000D+01
 0.10000D+01
 0.10000D+01
-0.10000D+01
-0.10000D+01
-0.10000D+01
-0.10000D+01
-0.10000D+01
-0.10000D+01

 0.12000D+02
 0.12000D+02
 0.12000D+02
 0.12000D+02
-0.12000D+02
-0.12000D+02
-0.12000D+02
```

```
-0.12000D+02
-0.12000D+02

 0.10000D+01
 0.10000D+01
 0.10000D+01
 0.10000D+01
 0.10000D+01
-0.10000D+01
-0.10000D+01
-0.10000D+01
-0.10000D+01

 0.16000D+02
 0.16000D+02
 0.16000D+02
 0.16000D+02
 0.16000D+02
 0.16000D+02
-0.16000D+02
-0.16000D+02
-0.16000D+02

 0.10000D+01
 0.10000D+01
 0.10000D+01
 0.10000D+01
 0.10000D+01
 0.10000D+01
 0.10000D+01
-0.10000D+01
-0.10000D+01

 0.30000D+01
 0.30000D+01
 0.30000D+01
 0.30000D+01
 0.30000D+01
 0.30000D+01
```

```
     0.30000D+01
     0.30000D+01
    -0.30000D+01

     0.11000D+02
     0.11000D+02
     0.11000D+02
     0.11000D+02
     0.11000D+02
     0.11000D+02
     0.11000D+02
     0.11000D+02
     0.11000D+02
```

Файл с исходным вектором B

```
        9     1
     0.00000D+00
     0.00000D+00
     0.10000D+01
     0.00000D+00
     0.00000D+00
     0.00000D+00
     0.00000D+00
     0.00000D+00
     0.00000D+00
```

Ниже приводится содержимое файла с результатами работы подпрограммы PDGESV.

```
  Решение линейной системы A * X = B
          9           1
    -0.146081976924362695D-17
    -0.166666666666666657D+00
     0.500000000000000000D+00
     0.000000000000000000D+00
     0.000000000000000000D+00
     0.173472347597680709D-17
    -0.500000000000000000D+00
     0.166666666666666657D+00        0.000000000000000000D+00
```

1. *Антонов А.С.* Параллельное программирование с использованием технологии MPI: Учебное пособие. М.: Изд-во МГУ, 2004. 71 с.
2. *Антонов А.С.* Практический курс MPI. MPI: A Message–Passing Interface Standard (Version 1.1).
3. Библиотека численного анализа НИВЦ МГУ, http://num_anal.srcc.msu.ru/lib_na/
4. *Воеводин В.В.* Математические модели и методы в параллельных процессах. М.: Наука, 1986. 296с.
5. *Воеводин В.В.* Математические основы параллельных вычислений. М.: Изд-во МГУ, 1991. 345с.
6. *Воеводин В.В.* Параллельные структуры алгоритмов и программ. М.: ОВМ АН СССР, 1987. 148с.
7. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. С.-П. "БХВ–Петербург", 2002. 608с.
8. *Дацюк В.Н., Букатов А.А., Жегуло А.И.* Введение в организацию и методы программирования многопроцессорных вычислительных систем (методическое пособие, часть I). Ростов-на-Дону: Изд-во РГУ, 2000. 36с.
9. *Дацюк В.Н., Букатов А.А., Жегуло А.И.* Среда параллельного программирования MPI (методическое пособие, часть II). Ростов-на-Дону: Изд-во РГУ, 2000. 65с.
10. *Немнюгин С.А., Стесик О.Л.* "Параллельное программирование для многопроцессорных вычислительных систем". "БХВ-Петербург", 2002. 400с.
11. *Корнеев В.Д.* "Параллельное программирование в MPI". Новосибирск: Изд-во СО РАН, 2000. 213 с.
12. *E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. Mckenney, S. Ostrouchov, and D. Sorensen* LAPACK Users' Guide, Society for Industrial and Applied Mathematics. Philadelphia, PA, second ed., 1995.
13. *L.S. Blackford, J. Choi, A. Cleary, J. Demmej, I. Dhillon, J.J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D.W. Walker, and R. C. Whaley* ScaLAPACK: A portable linear algebra library for distributed memory computers — design issues and performance, in Proceedings of Supercomputing '96, Sponsored by ACM SIGARCH and IEEE Computer Society, 1996. (ACM Order Number: 415962, IEEE Computer Society Press Order Number: RS00126. http://www.supercomp.org/sc96/proceedings/).
14. http://www.netlib.org/scalapack/index.html/.
15. *J. Choi, J. Dongarra, and D. Walker* PB-BLAS: A Set of Parallel Block Basic

Linear Algebra Subroutines, Practice and Experience, 8 (1996), pp. 517–535.
16. http://www.netlib.org/scalapack/html/pblas_qref.html/.
17. *J. Dongarra, G. Henry, and D. Watkins* A distributed memory implementation of the nonsymmetric QR algorithm, in Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, Philadelphia, PA, 1997, Society for Industrial and Applied Mathematics.
18. *J. Dongarra, R. van de Geun, and R.C. Whaley* Two dimensional basic linear algebra communication subprograms, in Environments and Tools for Parallel Scientific Computing, Advances in Parallel Computing, J. Dongarra and B. Tourancheau, eds., vol. 6, Elsevier Science Publishers B.V., 1993, pp. 31–40.
19. http://www.netlib.org/blacs/
20. *G. Geist and C. Romine* LU factorization algorithms on distributed memory multiprocessor architectures, SIAM J. Sci. Stat Comput., 9 (1988), pp. 639–649.
21. *G. Golub and C.F. Van Loan* Matrix Computations, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.
22. *W.W. Hager* Condition estimators. SIAM J. Sci. Stat. Comput., 5 (1984), pp. 311–316.
23. *N.J. Higham* A survey of condition number estimation for triangular matrices. SIAM Review, 29 (1987), pp. 575–596.
24. height 2pt depth -1.6pt width 23pt, FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation, ACM Trans. Math. Softw., 14 (1988), pp. 381–396.
25. height 2pt depth -1.6pt width 23pt, Experience with a matrix norm estimator, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 804–809.
26. *C.L. Lawson, R.J. Hanson, D. Kincaid, and F.T. Krog* Basic linear algebra subprograms for Fortran usage, ACM Trans. Math. Soft., 5 (1979), pp. 308–323.
27. http://www.netlib.org/blas/
28. *W. Lichtenstein and S.L. Johnsson* Block–cyclic dense linear algebra, SIAM J. Sci. Stat. Comput. 14 (1993), pp. 1259–1288.
29. *L. Prylli and B. Tourancheau* Efficient block cyclic data redistribution, in EUROPAR'96, vol. 1 of Lecture Notes in Computer Science, Springer–Verlag, 1996, pp. 155–165.