Конечно-элементный решатель Надра-3D

Максим Белоус

Институт кибернетики имени В.М. Глушкова НАН Украины, просп. Академика Глушкова 40, Киев, Украина

maksbilous@ukr.net

Аннотация. *В Институте кибернетики имени В.М. Глушкова НАН Украины под руководством академика НАН Украины В.С. Дейнеки разработана информационная технология Надра-3D моделирования пространственных процессов в многокомпонентных средах. Основу ее математического аппарата составляют математические модели пространственных процессов в виде систем дифференциальных уравнений в частных производных в трехмерной постановке. Программная реализация выполнена в виде программного комплекса, состоящего из пре/пост-процессора и параллельного конечно-элементного решателя, функционирующего на многопроцессорных вычислительных комплексах семейства СКИТ Института кибернетики им. В.М. Глушкова и позволяющего выполнять расчеты для конечно-элементных сеток с миллионами узлов. В данной работе проводится обзор некоторых особенностей архитектуры этого конечно-элементного решателя.*

Ключевые слова

Пространственные процессы, метод конечных элементов, параллельные вычисления

1 Введение

Математическое моделирование пространственных процессов в многокомпонентных средах сегодня является важной составляющей разнообразных процессов проектирования инженерных сооружений и механизмов, анализа взаимного влияния частей проектируемых объектов, взаимодействия объектов с окружающей средой. Для решения дифференциальных уравнений, описывающих моделируемую систему, используется программное обеспечение, реализующее специализированные конечно-разностные или конечно-элементные вычислительные схемы.

Процесс моделирования состоит из следующих этапов: выбор математической модели исследуемого процесса, построение модели геометрии исследуемого объекта, задание параметров математической модели, построение расчетной сетки, построение и решение систем алгебраических уравнений, представление полученных результатов в подходящей для анализа форме. При этом инструменты построения моделей геометрии, расчетной сетки, анализа результатов часто отделены от расчетных модулей и составляют модули пре/пост-процессора, которые могут представлять собой вообще разные программные системы, использующие один и тот же решатель.

Возможности программы-решателя определяют спектр задач, которые способен решать программный комплекс в целом. При этом значительную роль играют используемые алгоритмы построения систем алгебраических уравнений, к которым сводится исходная система дифференциальных уравнений, используемые алгоритмы нахождения решений этих систем, доступные схемы размещения данных в оперативной памяти, возможность использования параллельных вычислений.

При проведении расчетов для реальных объектов часто возникает ситуация, когда для некоторой части исследуемой области неизвестны значения параметров используемой математической модели, хотя известен диапазон возможных значений этих параметров. Аналогичным образом, при моделировании динамики процессов с изменяющимися во времени параметрами, часто бывает необходимым оценить поведение процесса для различных законов изменения этих параметров. При этом геометрическая конфигурация исследуемого объекта остается неизменной. Т.е. речь идет о многовариантном моделировании. Следует отметить, что задачи моделирования процесса для различных наборов параметров часто являются независимыми друг от друга (т.е. имеется параллелизм на уровне задач), и при наличии вычислительных ресурсов возможен одновременный их

просчет. Тем не менее, проведение расчетов даже для одной такой задачи может потребовать существенных вычислительных и временных ресурсов, поскольку расчетные сетки могут содержать десятки и сотни миллионов узлов, что приводит к необходимости решать системы алгебраических уравнений с соответствующим количеством неизвестных. Таким образом, существенную роль в уменьшении времени расчетов играют используемые параллельные алгоритмы решения систем линейных алгебраических уравнений (СЛАУ) больших размерностей.

В данной работе рассматриваются некоторые особенности архитектуры конечно-элементного решателя Надра-3D, разработанного в Институте кибернетики имени В.М. Глушкова НАН Украины. Программа предназначена для моделирования методом конечных элементов (МКЭ) трехмерных пространственных процессов фильтрации, теплопереноса, изменения напряженно-деформированного состояния в многокомпонентных средах. Основной идеей при проектировании решателя было максимальное разделение подсистем формирования и заполнения СЛАУ МКЭ, подсистем хранения данных СЛАУ в оперативной памяти и подсистем решения СЛАУ с тем, чтобы предоставить возможность гибкого расширения программы и добавления новых возможностей для каждой из этих подсистем.

2 Архитектура конечно-элементного решателя Надра-3D

Моделирование некоторого пространственного процесса (или нескольких взаимосвязанных процессов) с помощью конечно-элементного решателя Надра-3D представляется как последовательность отработки указанных пользователем этапов (шагов) моделирования. Управление процессом порождения и выполнения этих шагов выполняется модулем формирования последовательности шагов моделирования (МФПШМ), который при помощи набора сервисных модулей выполняет подготовительную работу и передает управление модулям формирования и решения СЛАУ МКЭ, в которых реализованы различные вычислительные схемы.

При этом на уровне указанных модулей не делается никаких предположений об использовании конкретных схем размещения данных в оперативной памяти или алгоритмов решения СЛАУ. Они взаимодействуют с модулями промежуточного уровня, которые предоставляют абстрактное описание поведения конечного элемента, абстрактный интерфейс работы с матрицами и векторами и вызова некоторого алгоритма решения СЛАУ. Конкретное наполнение этих операций и алгоритмов осуществляется потомками классов промежуточного уровня.

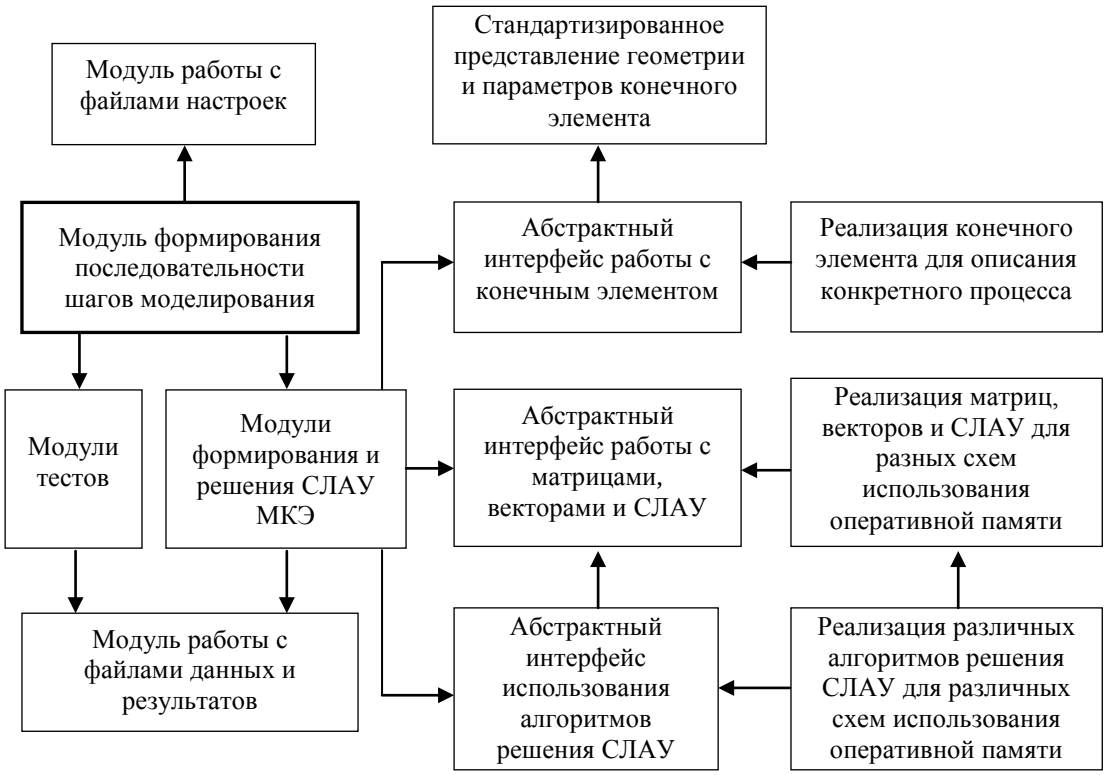

Рис. 1. Схема взаимодействия функциональных модулей конечно-элементного решателя Надра-3D.

Как видно из схемы взаимодействия функциональных модулей конечно-элементного решателя Надра-3D, представленной на рис.1., точкой входа программы является МФПШМ. После запуска программы на

выполнение, этот модуль формирует объект, предназначенный для обработки файла настроек, который выполняет загрузку и разбор настроек из паспорта задачи. После этого МФПШМ имеет доступ к информации о настройках путем вызова соответствующих методов объекта-парсера (модуль работы с файлами настроек, МРФН). Последовательность шагов моделирования записывается пользователем в разделе <sequence> паспорта задачи, а описание настроек отработки каждого шага – в соответствующем разделе паспорта. Получив раздел <sequence> МФПШМ считывает с него название раздела паспорта, описывающего текущий шаг моделирования, получает на основе этого названия от МРФН информацию соответствующего раздела, формирует набор объектов, необходимых для отработки текущего шага, и запускает их на выполнение. После завершения работы этих объектов МФПШМ освобождает память, получает от МРФН настройки следующего шага, формирует и запускает на выполнение набор объектов для его отработки. После отработки всех шагов, записанных пользователем в разделе <sequence> паспорта задачи, МРФН освобождает память и завершает выполнение программы.

На каждом шаге последовательности моделирования может выполняться или вычислительная схема моделирования некоторого процесса (отработка модулей формирования и решения СЛАУ МКЭ), или выполнение некоторых вспомогательных операций по обработке файлов данных и результатов (отработка модулей тестов).

И модули тестов, и модули формирования СЛАУ МКЭ используют для работы с файлами методы специализированного модуля работы с файлами данных и результатов, который инкапсулирует в себе все операции по созданию/открытию/закрытию этих файлов, проверки их корректности (декодирования заголовков), формированию массива индексов для быстрого доступа к отдельным фрагментам файлов.

На каждом шаге моделирования система использует информацию из файлов данных и записывает результаты расчетов в файлы результатов. Какие именно файлы используются – указывает пользователь в паспорте задачи. Модуль работы с файлами получает от МФПШМ раздел настроек текущего шага, откуда выбирает описание необходимых на этом шаге файлов. Описание файла содержит имя файла в системе, путь к файлу на жестком диске и тип открытия файла (чтение/запись). На основании этой информации модуль работы с файлами создает или открывает необходимые файлы и формирует массив пар «имя_файла_в_системе»-«указатель_на_открытый_файл». Во время отработки объекта вычислительной схемы или теста, они запрашивают нужные для работы файлы у модуля работы с файлами по имени файла в системе и получают необходимый указатель.

Таким образом, имеющимися средствами в паспорте задачи можно записать вычислительную последовательность, файлы результатов на некотором шаге которой будут входными файлами для последующих шагов.

2.1 Файлы данных

Описание исходной задачи выполняется пользователем в виде двух файлов: файла данных о геометрии и конечно-элементном представлении исследуемого объекта и файла описания параметров математической модели процесса и их привязки к конечным элементам. Создание модели геометрии, проведение конечно-элементного разбиения области и генерирование этих файлов осуществляются пользователем при помощи какого-либо внешнего редактора (например, препроцессор программного комплекса Надра-3D).

Схематизация структуры файлов данных о геометрии области и параметрах математической модели представлена на рис.2. Файл описания данных о геометрии и конечно-элементном представлении исследуемого объекта содержит полное описание сетки конечных элементов и состоит из разделов, описывающих конечно-элементное разбиение области, конечно-элементное разбиение границы области, конечно-элементное разбиение поверхностей контакта. При этом схема хранения данных является одинаковой как для трехмерных, так и для двумерных областей. Каждый раздел файла данных состоит из набора фрагментов описания конечно-элементной сетки. Каждый из этих фрагментов содержит полное описание данных – прочитав фрагмент, например, описания трехмерной сетки, из хранящейся в нем информации можно узнать координаты всех узлов, принадлежащих этому фрагменту, геометрию всех тетраэдров этого фрагмента, их локальные и глобальные номера. Таким образом, обеспечивается локальность обработки данных программой-решателем (хотя и за счет дублирования некоторой информации).

Файл данных о параметрах математической модели и их привязке к сетке конечно-элементного разбиения имеет структуру, подобную структуре файла данных о геометрии объекта. Каждому разделу файла описания геометрии соответствует раздел файла описания параметров. Разделы файла описания параметров содержат фрагмент библиотеки параметров, используемый в этом разделе, а также индекс привязки ее элементов к соответствующим данным фрагмента файла описания геометрии. Таким образом, загрузив фрагмент конечно-

элементного разбиения и соответствующий фрагмент описания параметров, программа получает всю необходимую информацию для построения поправок к СЛАУ МКЭ.

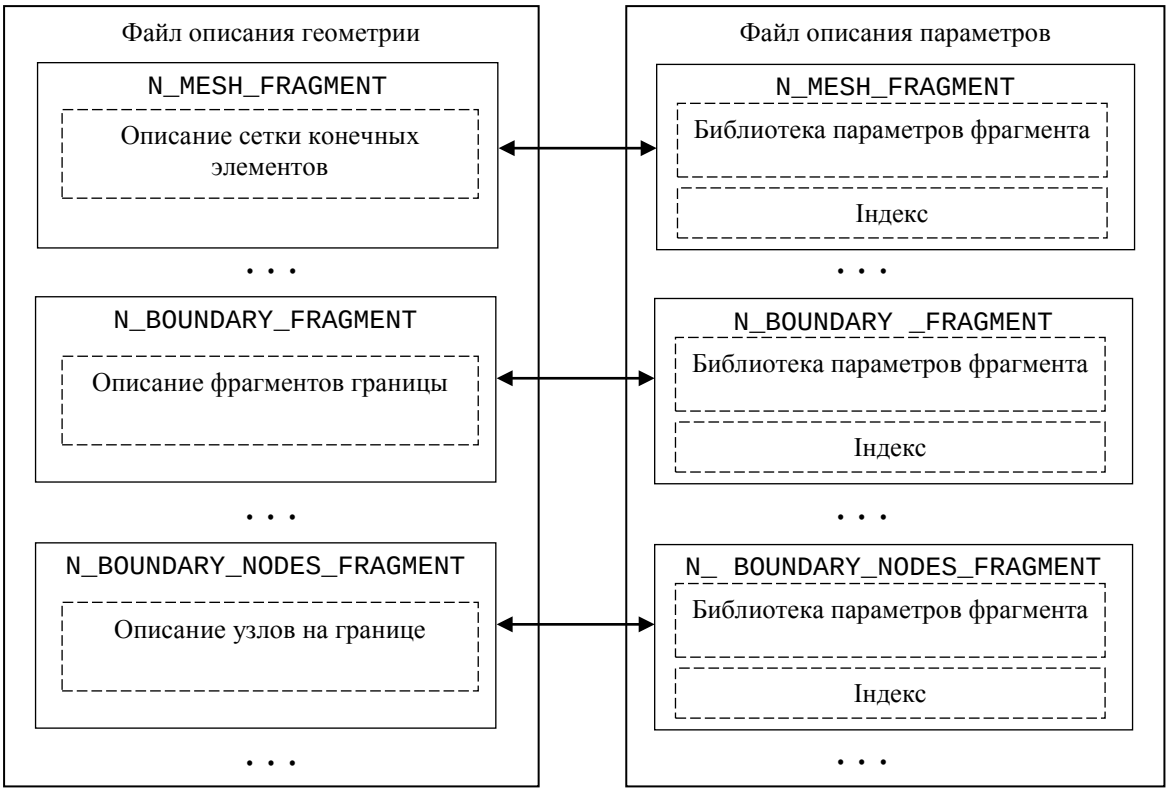

Рис. 2. Структура файлов данных.

Описание фрагмента конечно-элементного представления состоит из массива узлов nVertex3* V, принадлежащих сетке конечных элементов этого фрагмента, и последовательно записанного массива int *FE локальных номеров (в пределах фрагмента) узлов, принадлежащих конечным элементам этого фрагмента. Количество узлов N, принадлежащих одному элементу, определяется значением кода геометрии элемента. Таким образом, j-й элемент содержит узлы с номерами FE[j*N]… FE[(j+1)*N-1].

Для описания параметров математической модели используется набор массивов - int *pSize, int *indexP, int *P – для хранения кодов параметров, iSize, int *indexI, int *I – для хранения целочисленных значений для всех наборов параметров, и массивы int *dSize, int *indexD, double *D – для хранения значений с плавающей точкой для всех наборов параметров. Взаимосвязь каждой тройки массивов показана на рис.3.

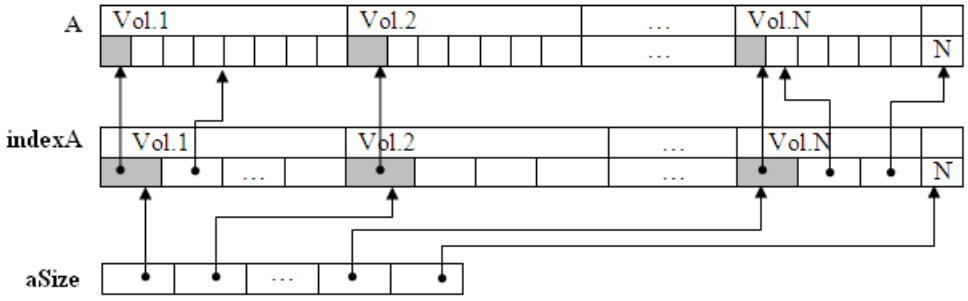

Рис. 3. Структура библиотеки параметров математической модели.

Для работы с библиотекой параметров в описанном выше представлении системой генерируется объект класса nParamLib, инкапсулирующий в себе указанные структуры хранения данных и реализующий методы загрузки/сохранения данных и доступа к ним.

По номеру раздела библиотеки параметров (int volume) и номеру записи в нем (int record) объект nParamLib возвращает для этой записи количество параметров (int pCnt), количество целочисленных значений (int iCnt), количество значений с плавающей запятой (int dCnt) и указатели на ячейки массивов, начиная с которых в библиотеке последовательно хранятся эти значения (int *P, int *I, double *D соответственно). Таким образом, схема хранения данных о параметрах математической модели является одинаковой для разных типов конечных элементов и разных типов процессов.

Для привязки параметров к конечным элементам используется целочисленный массив-индекс int *bindIndex, в котором для каждого конечного элемента последовательно записаны значения [fe_code][vol1]…[volN], где fe_code – код конечного элемента, vol1 – номер записи в разделе 1 библиотеки параметров, volN – номер записи в разделе N библиотеки параметров.

2.2 Описание конечных элементов

Генерирование данных для заполнения СЛАУ МКЭ выполняется объектами, описывающими конечные элементы. Классы всех объектов программного комплекса, описывающих функциональность различных конечных элементов, являются потомками базового класса nFiniteElement, объявляющего абстрактный интерфейс для выполнения таких операций, как формирования элементарных матриц жесткости, масс, элементарных векторов правой части, стандартизированной загрузки описания геометрии и параметров конечного элемента. Конкретные реализации этих операций выполняются в потомках nFiniteElement.

Для порождения системой этих объектов в процессе выполнения вычислительной схемы используется объект класса nFEFactory, который по коду конкретного конечного элемента генерирует объект соответствующего класса и возвращает системе указатель на этот объект, как на объект класса nFiniteElement. Таким образом, из модулей формирования СЛАУ МКЭ все объекты, описывающие конечные элементы, видны как объекты класса nFiniteElement и взаимодействие с ними осуществляется только посредством операций, объявленных как методы nFiniteElement.

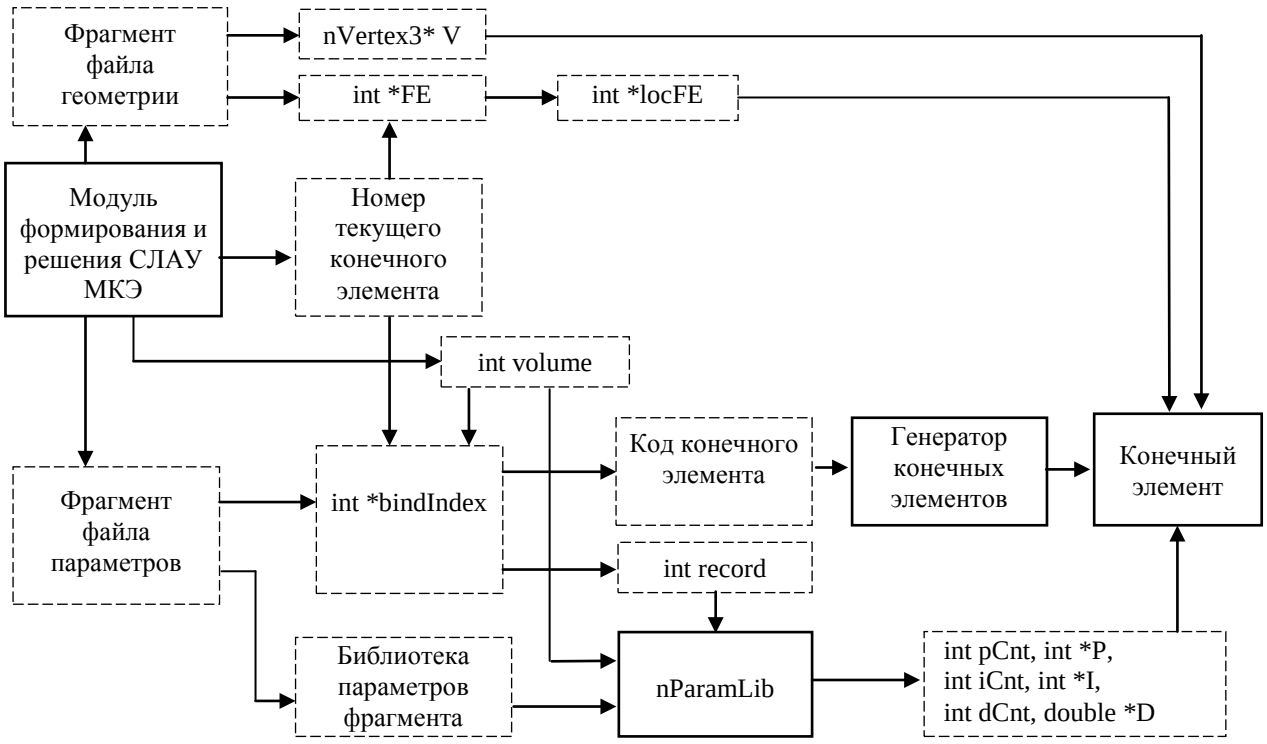

Рис. 4. Схема порождения объектов, описывающих конечные элементы

Для использования объекта, описывающего конечный элемент, его необходимо инициализировать данными о его геометрии и параметрах математической модели. Реализация загрузки геометрии и параметров выполнена в виде методов этого объекта, при этом на их вход подается стандартизированное представление геометрии и параметров, а интерпретация этих данных уже относится к реализации конкретного конечного элемента.

Схема процесса генерирования и инициализации объекта, описывающего конечный элемент, приведена на рис. 4. Сплошными линиями обозначены объекты, пунктирными – блоки информации, стрелками – направления передачи информации.

Как видно из схемы, модулям формирования СЛАУ МКЭ ничего неизвестно о конкретном содержании конечного элемента – порождение объекта конечного элемента осуществляется специализированным классом-генератором, выборка значений из библиотеки параметров – посредством интерфейса специализированного объекта-библиотеки, выборка значений из массивов геометрии – по алгоритму, общему для всех конечных элементов, инициализация конечного элемента – посредством общего для всех конечных элементов интерфейса загрузки геометрии и параметров.

После инициализации конечный элемент может быть использован модулями формирования СЛАУ. При этом от них, по-прежнему, остаются скрытыми детали формирования элементарных матриц и векторов конечным элементом. Они только отправляют запросы на их формирование и получают результаты в виде массива int *N, содержащего номера рассылки компонент элементарных матрицы или вектора в глобальные матрицу или вектор СЛАУ, и массива double *data, содержащего сгенерированные элементарные матрицу или вектор.

2.3 Описание матриц, векторов, и алгоритмов решения СЛАУ

Алгоритм построения матрицы и вектора СЛАУ МКЭ зависит от вычислительной схемы и реализуется соответствующим модулем формирования и решения СЛАУ МКЭ, но в нем можно выделить следующие составляющие:

- инициализация конечного элемента данными о геометрии и математической модели;
- формирование конечным элементом элементарных матриц и векторов;
- формирование на их основе поправок к элементам матрицы и вектора СЛАУ;
- размещение вычисленных поправок в матрице и векторе СЛАУ.

Формирование поправок к матрице и вектору глобальной СЛАУ выполняется текущим модулем формирования и решения СЛАУ МКЭ на основании сгенерированных конечным элементом элементарных матрицы и вектора и зависит от моделируемого процесса (стационарный/нестационарный, линейный/нелинейный) и выполняемого этапа моделирования (обработка конечно-элементного представления области, обработка конечно-элементного представления границы, учет главных краевых условий и т.п.).

Размещение вычисленных поправок в глобальной матрице и векторе СЛАУ зависит от схемы использования оперативной памяти. Выбор этой схемы определяется размерностью задачи, структурой матрицы, реализованными алгоритмами решения СЛАУ и имеющимися вычислительными ресурсами. При этом, одна вычислительная схема при различных постановках задачи может использовать различные схемы размещения данных в оперативной памяти. Для учета этой особенности заполнения матриц и векторов СЛАУ выборка/размещение их элементов и операции матричной алгебры, доступны модулям формирования СЛАУ МКЭ через интерфейс абстрактных классов nSLAEMatrix и nSLAEVector. Причем, подобно работе с описанием конечных элементов, только через интерфейс этих классов.

Конкретная схема размещения СЛАУ в оперативной памяти реализуется в потомках этих классов, а за порождение объектов, описывающих конкретные реализации матриц и векторов, отвечает специализированный класс-генератор, который на основании набора параметров создает объекты соответствующих классов и возвращает запрашивающему модулю указатели на них как на nSLAEMatrix или nSLAEVector. Таким образом, обеспечивается возможность использование разных схем использования оперативной памяти без изменения или дублирования кода модулей формирования СЛАУ МКЭ.

Описание совокупности матрицы и вектора именно как СЛАУ в системе осуществляется базовым классом nSLAE. Решение о создании отдельного класса для описания СЛАУ вызвано тем, что в алгоритмах решения СЛАУ может использоваться специализированная схема размещения данных в оперативной памяти, предполагающая размещение матрицы и вектора в одном массиве данных. Интерфейсы классов nSLAEMatrix и nSLAEVector предполагают возможность их использования также и для хранения дополнительных матриц и векторов, если это предусмотрено при реализации вычислительной схемы. Т.е. необходимости размещения пары матрица-вектор в общем массиве данных нет, тем более, что объекты этих классов не обязательно будут рассматриваться как СЛАУ. А объект класса nSLAE предназначен именно для описания пары матрица-вектор,

которая будет использоваться как входные данные для объекта-решателя СЛАУ. При этом конкретные схемы использования пары матрица-вектор как СЛАУ реализуются в потомках nSLAE, внутренняя реализация которых скрыта от модулей формирования СЛАУ МКЭ, а работа с матрицей и вектором, хранящимися в них, осуществляется посредством интерфейса nSLAEMatrix и nSLAEVector.

Абстрактный интерфейс инициализации и запуска на выполнение решателя СЛАУ описан классом nSLAESolver, а конкретные алгоритмы нахождения решений реализуются в потомках этого класса. При этом имеет смысл говорить о проектировании набора классов для некоторого алгоритма, поскольку они должны, с одной стороны, предоставлять интерфейс nSLAEMatrix и nSLAEVector, с другой – позволять конкретному потомку nSLAESolver прямой доступ к внутреннему представлению данных. На рис.5 приведена схема взаимодействия классов, описывающих матрицы и векторы.

В созданном программном обеспечении используется блочная строково-циклическая схема размещения матрицы и вектора СЛАУ в памяти процессоров многопроцессорного комплекса и созданная в Институте кибернетики имени В.М. Глушкова параллельная реализация алгоритма LDL^T–разложения для нахождения решений СЛАУ [3, 4]. Соответственно требованиям формата представления данных этой программы и реализованы специализированные классы-потомки классов nSLAEMatrix, nSLAEVector, nSLAESolver.

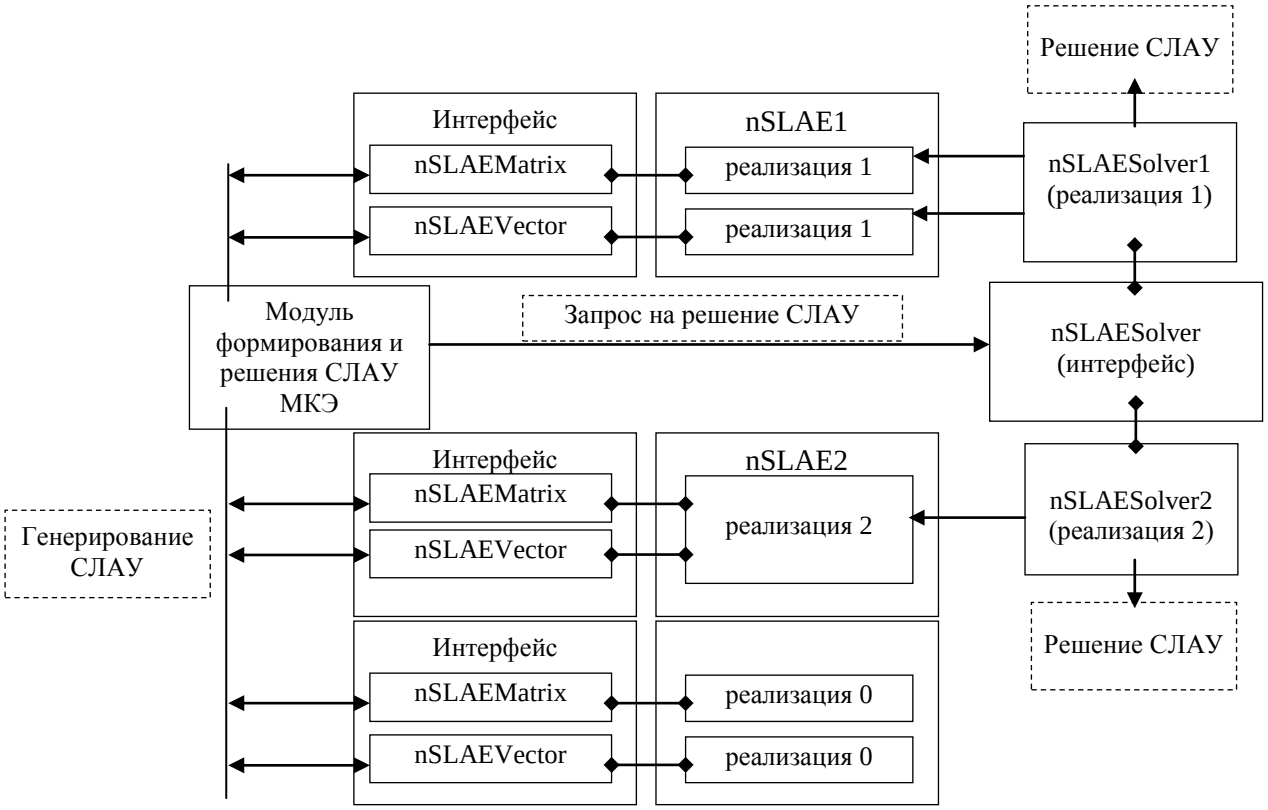

Рис. 5. Схема взаимодействия классов, описывающих матрицы и векторы.

2.4 Вычисления

На данный момент минимальное и оптимальное количество необходимых вычислительных узлов (процессоров), необходимой оперативной памяти и объем времени вычислений определяются параметрами использованной в решателе программы нахождения решений СЛАУ с симметричной матрицей ленточной структуры по методу LDL^T–разложения и зависят от размерности матрицы (количества узлов сетки конечно-элементного разбиения исследуемой области), полуширины ленты матрицы, размера блока строково-циклической схемы размещения данных в оперативной памяти многопроцессорных вычислительных комплексов.

Разработанное программное обеспечение использовалось для решения тестовых задач моделирования процесса теплопроводности и для решения задач моделирования фильтрации подземных вод в трехмерной неоднородной геологической среде Черниговского региона с учетом разветвленной сети поверхностных рек [5].

В таблице 1 приведены значения параметров для некоторых из просчитанных модельных задач, которые позволяют оценить порядок необходимого для выполнения вычислений времени (в таблице приведены значения оценок работы программы для расчета одного варианта параметров математической модели).

Таблица 1. Параметры задач

Размерность СЛАУ	Полуширина ленты матрицы	Оценка необходимого объема оперативной памяти (Gb)	Время расчетов (мин.)	Количество процессоров
227388	2566	4.4	13	12
634428	2901	13.7	30	24
910254	3243	22	49	24
1125798	3814	32	40	20

Следует заметить, что для разных задач оптимальным является использование различного количества процессоров и различных настроек программы нахождения решения СЛАУ, поэтому возможны ситуации, когда увеличение количества задействованных процессоров не приводит к уменьшению времени работы программы, или СЛАУ с большей размерностью решается быстрее на меньшем количестве процессоров, чем СЛАУ с меньшей размерностью на большем количестве процессоров.

3 Выводы

В статье описаны некоторые особенности архитектуры конечно-элементного решателя Надра-3D, предназначенного для конечно-элементного многовариантного моделирования пространственных процессов в многокомпонентных средах с использованием параллельных вычислений. Предполагается расширение созданного программного обеспечения путем реализации классов, описывающих различные типы конечных элементов. Также планируется добавление, в рамках разработанной архитектуры, возможности использования других программ нахождения решений СЛАУ и других схем использования оперативной памяти (в том числе, использование итерационных методов).

4 Благодарности

Разработка последней версии конечно-элементного решателя Надра-3D частично выполнена как составляющая работ в рамках Государственной целевой научно-технической программы внедрения и применения грид-технологий на 2009-2013 гг.

Ссылки

[1] Страуструп Б. Язык программирования С++. – М.: Бином, 1999. – 990 с.
[2] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – С.-П.: Питер, 2010. – 367 с.
[3] Молчанов И.Н. Введение в алгоритмы параллельных вычислений. – Киев: Наук.думка,1990. – 128 с.
[4] Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф. Параллельные алгоритмы решения задач вычислительной математики. – Киев: Наук. думка, 2008. – 247 с.
[5] Дейнека В.С., Білоус М.В. Побудова моделі ґрунтового масиву чернігівського регіону в програмному комплексі НАДРА-3D // Матеріали II науково-технічної конференції «Обчислювальні методи і системи перетворення інформації», присвяченої пам’яті Б.О. Попова, Львів, 4-5 жовтня 2012. – Львів, 2012. – С.78-81.