ПРЕПРОЦЕССОР ДЛЯ РАСЧЕТА ПРОСТРАНСТВЕННЫХ ЗАДАЧ СО СВОБОДНОЙ ПОВЕРХНОСТЬЮ

И. В. Григорьева, А. М. Гудов
Кемеровский государственный универсистет, Россия

In this work a program is presented providing the preparation of initial data for the numerical solution of 3D problems with a free surface using the method of boundary integral equations. All the stages of data preparation have been considered, starting from the generation of the surface grid and imposing on it the boundary conditions to obtaining the physical constants of the problem and the parameters of installation of the solver.

Введение

Традиционно принято подразделять этапы численного эксперимента на три основные части: препроцессор — модуль, который готовит входные данные для расчета; процессор или решатель, — выполняет необходимые вычисления, и постпроцессор — модуль для обработки численных результатов. В данной работе представлен препроцессор, позволяющий подготовить начальные данные для расчета пространственных задач динамики жидкости со свободной поверхностью.

Представленная программа-препроцессор является частью программного комплекса по решению плоских, осесимметричных и пространственных задач методами граничных интегральных уравнений [2], который уже около пяти лет разрабатывается в Центре новых информационных технологий Кемеровского государственного университета. Специфика численных методов, основанных на использовании интегральных уравнений, записанных по границе области решения, состоит в реализации следующих основных этапов: построения сетки граничных элементов, покрывающей только поверхности рассматриваемых объектов; задания на элементах поверхности граничных условий; решения интегральных уравнений, описывающих основные законы механики, совместно с дифференциальными уравнениями, используемыми для задания соотношений в граничных и внутренних точках области. Все они подробно описаны в работах [3–5]. Эти особенности требуют от разработчиков программного обеспечения использования специальных алгоритмов при реализации всех компонентов программного комплекса — от подготовки начальных данных для расчета до обработки результатов численного эксперимента.

В настоящее время известны лишь несколько промышленных пакетов, основанных на методах граничных элементов. Самый мощный из них — BEASY [9], ориентирован на решение задач теории упругости. Кроме того, по мнению авторов, все эти пакеты имеют один существенный недостаток — “универсальность”. Данный препроцессор разрабатывался для уже существующих решателей и освобождает исследователя от необходимости

настраивать "универсальный" пакет под конкретные условия задачи. Простой и понятный пользовательский интерфейс позволяет легко генерировать начальные данные для расчета.

Препроцессор выполнен в качестве приложения для операционной системы Windows 95 / Windows NT, что делает программу легкой в освоении и позволяет использовать широкий спектр возможностей, предоставляемый этими операционными системами.

1. Структура препроцессора

Препроцессор состоит из нескольких функциональных блоков, которые связаны в единое приложение посредством программного интерфейса (рис. 1).

Блок генерации поверхностной сетки позволяет создавать набор элементов, аппроксимирующий поверхности рассматриваемых простанственных объектов, блок геометрических преобразований — визуализировать образ сетки и осуществлять с ним различные манипуляции, такие как изменение масштаба, сдвиг и вращение. Это необходимо для наглядной работы с пространственными объектами. Блок задания параметров задачи дает возможность полностью определить как условия задачи, так и параметры для данной версии программы-решателя. Используя специальные параметры, можно управлять самим процессом вычислений и определять необходимый формат вывода результатов. Обмен данными между блоками происходит посредством файлов данных, формат которых обсуждается ниже.

1.1. Блок генерации поверхностной сетки

Подсистема генерации сетки обеспечивает построение треугольных или четырехугольных элементов, аппроксимирующих достаточно сложные пространственные объекты. Идея создания сетки основана на разбиении исходной сложной поверхности на отдельные треугольные или четырехугольные зоны. Каждая из них, отображаясь на каноническую область (топологически подобную данной), делится на заданное число элементов. Обратное преобразование позволяет получить требуемую сетку на поверхности объекта. После использования специального алгоритма слияния зоны собираются в единую поверхность.

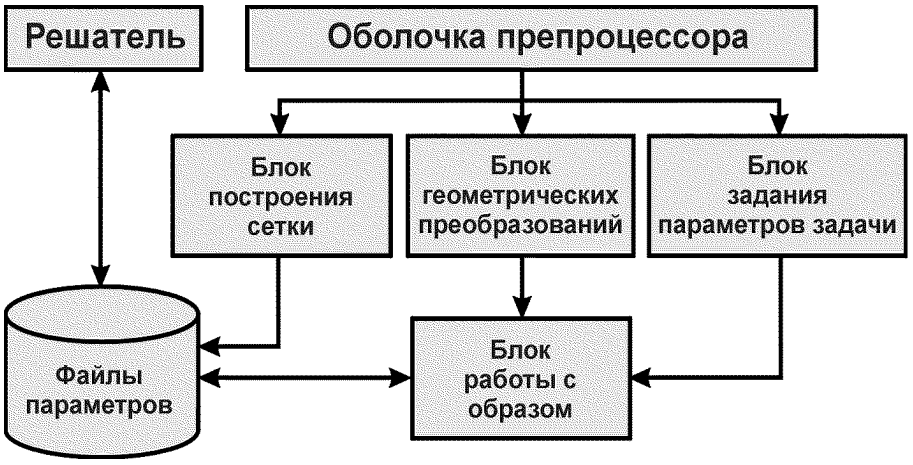

Рис. 1. Взаимодействие блоков и обмен данными в препроцессоре.

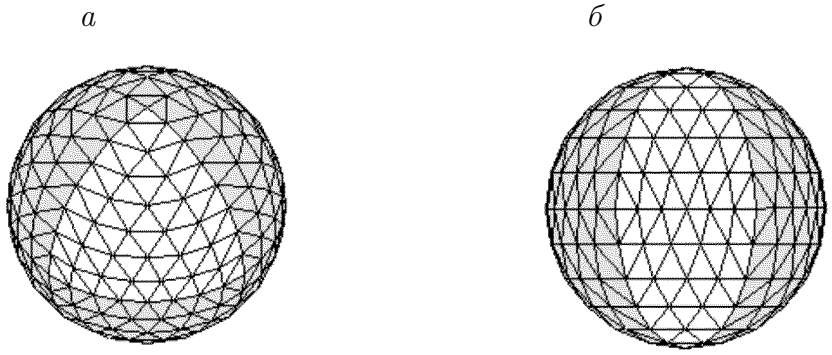

Рис. 2. Сегменты на поверхности сферы. *а* — простой сегмент, состоящий из одной опорной зоны, *б* — сложный сегмент, состоящий из двух опорных зон.

Таким образом, "вырезая" на поверхности плоские или криволинейные сегменты и разбивая отдельно каждый из них, можно покрывать сеткой практически любые объекты. Простой сегмент может представлять собой треугольную или четырехугольную зону, которая непосредственно разбивается на элементы. Сложные сегменты поверхности можно разделить, в свою очередь, на более простые опорные зоны (рис. 2).

Для разбиения опорных зон используется следующий алгоритм. На треугольной зоне вводится множество локальных L-координат. Их особенность состоит в том, что они принимают постоянные значения на линиях, параллельных сторонам зоны. По известной линейной зависимости между L-координатами и декартовыми, а также заданному количеству делений каждой стороны треугольной зоны легко построить плоскую сетку. В случае, когда зона имеет криволинейные границы, используется механизм определения специальных функций формы на треугольном элементе. В работе [10] представлен аналогичный метод для разбиения четырехугольных областей, где вместо L-координат используются базисные функции для четырехугольных элементов.

Недостатком описанного подхода является необходимость "вырезания" большого количества опорных зон для сильно искривленных поверхностей. Однако для канонических областей, таких как сфера, эллипсоид и т. п., данный подход легко модифицировать, используя криволинейные координаты, хорошо аппроксимирующие такого рода поверхности. Для этого можно ввести полярные "долготу" и "широту", которые и будут играть роль L-координат. При разбиении более сложных объектов можно использовать комбинацию криволинейных и L-координат в зависимости от вида вырезаемой зоны. В этом случае каждая зона разбивается в соответствии с одним из подходов, а для связи зон используется алгоритм слияния.

В процессе разбиения допускается произвольная нумерация узлов зоны. Алгоритм обеспечивает нахождение совпадающих узлов на границах прилегающих зон и их перенумерацию в определенном для всей области порядке. Результатом работы блока генерации сетки являются три массива, полностью определяющие геометрию области: массив координат узлов поверхностной сетки, массив согласования узлов для каждого граничного элемента, которые перечисляются в соответствии с выбранным направлением обхода, массив связности каждого узла и окружающих его элементов (сам узел, в свою очередь, также является частью этих элементов). Такое описание сетки является избыточным, так как по одному из массивов связности можно построить другой, но эта избыточность позволяет существенно ускорить работу с сеткой как препроцессора, так и самого решателя,

что в условиях использования большого количества ресурсов компьютера при решении трехмерной задачи представляется очень важным.

1.2. Блок работы с образом

Неотъемлемой частью препроцессора является блок геометрических преобразований объектов. Здесь использованы алгоритмы машинной графики и специальные методы геометрических построений [8], что позволило не занимать больших вычислительных ресурсов, с одной стороны, и дать в руки пользователя понятный и удобный интерфейс манипулирования пространственными объектами почти в реальном времени — с другой.

При манипуляциях с объектом в системе мировых координат используются переход к однородным координатам и обобщенная матрица преобразований в пространстве [7] с последующей обратной нормализацией, что обеспечивает выполнение комплекса операций сдвига, частичного изменения масштаба, вращения, зеркального отображения, переноса, а также изменения масштаба изображения в целом. Для получения изображения объекта на экране строится его ортогональная проекция, т. е. осуществляется переход к видовым координатам [1]. На основе набора видовых координат после несложных преобразований получается набор экранных координат отображаемого объекта.

Визуализация объекта осуществляется на основе простых, но тем не менее достаточно эффективных алгоритмов векторной графики [6]. Алгоритм удаления невидимых линий выполнен на основе известного метода Робертса [6], в котором отображаемая сцена должна быть разбита на выпуклые тела, после чего производится анализ видимости элементов и ребер каждого тела, а также анализ перекрываемости всех тел сцены.

После проведения всех необходимых преобразований на экране отображаются узлы и ребра видимых элементов поверхности. Однако работа пользователя с “проволочным” каркасом объекта не всегда удобна для восприятия. Поэтому в интерфейсе программы предусмотрено изображение объекта с использованием модели освещения. Для создания этой модели применяется простой несглаживающий алгоритм, в котором интенсивность цветового тона заливки зависит только от взаимного расположения нормальных векторов элемента и плоскости проекции сцены [6].

1.3. Блок задания параметров

Этот блок препроцессора позволяет определить характерные физические константы задачи. Кроме того, здесь можно задавать некоторые специфические параметры, указывающие процессору на различные способы интерпретации данных в момент проведения вычислений. Все вводимые параметры разделяются на три логические группы — физические, временные и вычислительные, а также дополнительные параметры, влияющие на способ отображения результатов в момент счета задачи. Кроме того, блок задания параметров позволяет на готовом “каркасе” из граничных элементов задать краевые условия — Неймана или Дирихле.

2. Программная реализация

Основная часть программного кода реализована в среде визуального программирования Delphi 3.0, подсистема построния сетки написана в среде Microsoft Fortran Power Station 4.0 на основе стандарта Fortran 90.

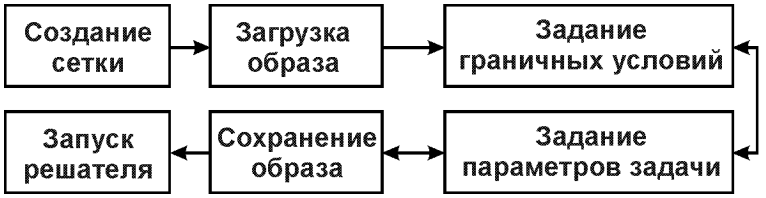

Рис. 3. Последовательность действий при работе в препроцессоре.

Диаграмма, изображенная на рис. 3, поясняет последовательность действий по подготовке данных в препроцессоре. Работу необходимо начать с создания сетки, для чего нужно выбрать заранее подготовленный текстовый файл с координатами опорных зон и запустить из программы-препроцессора подсистему генерации сетки. После этого образ сетки можно визуализировать. Чтобы полностью определить условия задачи и необходимые параметры программы-решателя, необходимо обратиться к блокноту задания параметров. В режиме задания граничных условий необходимо определить их во всех узлах сетки. После сохранения файла определения задачи можно запустить решатель либо из оболочки программы-препроцессора, либо как самостоятельную программу, указав ей этот файл в качестве параметра.

Если решатель реализован для Windows, то препроцессор откроет под него специальное окно, куда будет организован вывод графической информации. Если решатель реализован без использования графического интерфейса (например, под управлением ОС DOS или в качестве консольного приложения), то препроцессор запустит его на выполнение в фоновом режиме. Такая организация препроцессора позволяет легко приспособить его под архитектуру клиент-сервер: готовить данные на рабочей станции под управлением операционной системы Windows, а решать задачу на специальном удаленном вычислительном сервере.

2.1. Информационные структуры

Одной из главных функций препроцессора является обеспечение удобного интерфейса для работы с графическим объектом. Вот почему продуманная организация структуры классов, представляющих как саму границу, так и ее графический образ, может существенным образом увеличить скорость геометрических построений и определить четкую логическую структуру всего приложения.

Основными классами, представляющими саму границу и ее графический образ, являются `TSolid` и `TSolImage`, на основе полей и методов которых строится механизм визуализации самого образа и последующих с ним манипуляций (см. таблицу). Объект класса `TSolid` описывает физическую границу области и ее свойства, а объект класса `TSolImage` — графический образ сетки и методы его визуализации. Объекты этих классов содержат в себе структуры, состоящие из объектов более простых классов или классов нижнего уровня, а именно `TRib` и `TElem`. Оба класса нижнего уровня, в свою очередь, являются потомками класса `TObject`, который включает в себя указатели на предыдущий и последующий элементы двусвязного списка объектов, что позволяет построить двусвязные списки ребер и элементов, включаемые в объект описания сетки `TSolid`. Необходимо отметить, что классы `TSolid` и `TSolImage` также являются потомками класса `TObject` и при необходимости также могут образовывать двусвязные списки.

Интерфейс препроцессора (см. рис. 2) имеет стандартный для Windows-приложений

вид и состоит из следующих элементов управления:

— строка заголовка программы;

— главное меню программы;

— панель кнопок быстрого доступа;

Т а б л и ц а

Структура классов описания сетки и ее образа

Класс	Структура
`TObject`	Указатели на предыдущий и последующий элементы двусвязного списка объектов
`TRib`	Номера узлов — концов ребра Логическая переменная, определяющая видимость ребра при визуализации
`TElem`	Номера вершин элемента Номера ребер элемента Логические переменные видимости элемента Коэффициенты уравнения плоскости, которой принадлежит элемент
`TSolid`	Массив физических координат узлов сетки Первое и последнее ребра двусвязного списка ребер Первый и последний элементы двусвязного списка элементов Количество точек, элементов и ребер сетки Массив типов граничных условий в узлах Массив значений граничных условий в узлах сетки.
`TSolImage`	Массивы мировых, видовых и экранных координат узлов сетки Координаты точки наблюдения Массив логических переменных видимости ребер Массив логических переменных “помеченности” узлов Логические переменные наличия некоторых элементов изображения (невидимых линий, видимых и невидимых узлов, осей координат, номеров элементов, идентификаторов граничных условий в узлах, модели освещения) в визуализируемом образе **Процедура** определения невидимых линий **Процедура** определения модели освещения **Процедура** определения экранных координат **Процедура** рисования образа

— панель кнопок управления образом;

— строка состояния;

— рабочие и диалоговые окна программы.

Рабочие окна препроцессора:

— окно визуализации образа (основное окно приложения);

— окно подпрограммы создания сетки;

— окно-блокнот задания параметров.

2.2. Главное окно программы

Важным моментом, обеспечивающим удобство работы пользователя, является реализация функций манипулирования с объектом. Это достигается следующим образом.

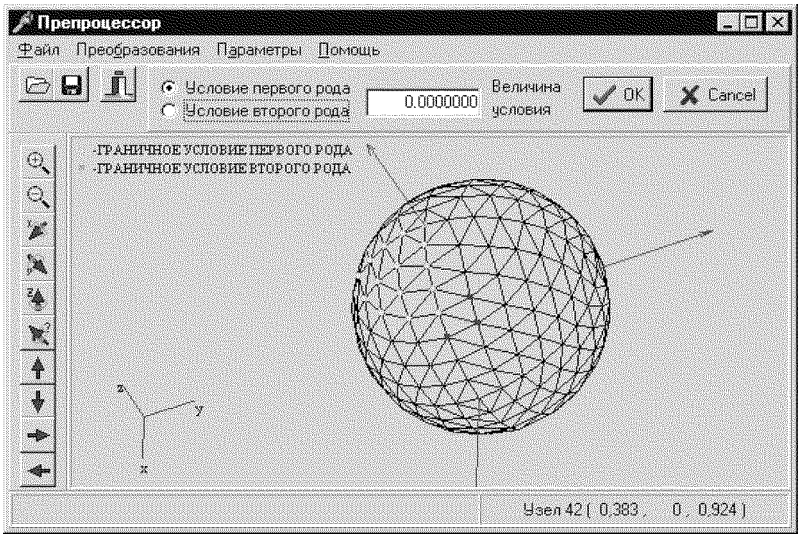

Рис. 4. Основное окно программы препроцессора.

Основное окно программы препроцессора предназначено для работы с визуализированной сеткой или так называемым образом (рис. 4). В левой части основного окна приложения расположена панель кнопок управления образом. Пока образ не определен пользователем, кнопки панели остаются недоступными. Для загрузки образа необходимо указать текстовый файл описания, на основе которого строится ортогональная проекция сетки. Как уже упоминалось, препроцессор предусматривает различные режимы визуализации. К исходному "проволочному" каркасу сетки с удаленными невидимыми гранями можно добавить невидимые ребра, видимые и невидимые узлы, оси координат, номера элементов, а также типы граничных условий (если таковые уже заданы) или построить модель освещения. В левом нижнем углу главного окна выводятся оси координат, поясняющие положение тела в пространстве. Кроме того, направив указатель "мышки" в любой узел сетки, пользователь может видеть в строке состояния основного окна номер узла и его координаты. Кнопки управления реализуют стандартный набор средств управления графическим объектом, который позволяет пользователю поворачивать образ около любой координатной или произвольной оси, осуществлять сдвиг, приближать или удалять образ.

2.3. Задание параметров задачи

Для задания параметров задачи пользователю предоставляется диалоговое окно-блокнот "Параметры" (рис. 5). Все параметры, задаваемые в нем, разделяются на три раздела: Физические, Временные и вычислительные и Дополнительные. Первая страница блокнота — Физические параметры — позволяет задать такие константы задачи, как характерное и внутреннее давление, плотность, наличие свободной границы или твердой стенки, расстояние до твердой стенки или свободной поверхности, неограничную или ограниченную всюду область, а также коэффициенты поля скоростей. Раздел Временные и вычислительные параметры включает две группы вводимых констант. Первая позволяет задать временные параметры для нестационарной задачи: начальное, конечное время и шаг по времени, вторая — влиять на процесс счета решателя, задавая допустимую погрешность вычислений, максимальное количество итераций и т. д. Третья страница блокнота — Дополнительные параметры — дает возможность определить исходные данные, влияющие на процесс отображения информации в момент решения задачи самим решателем.

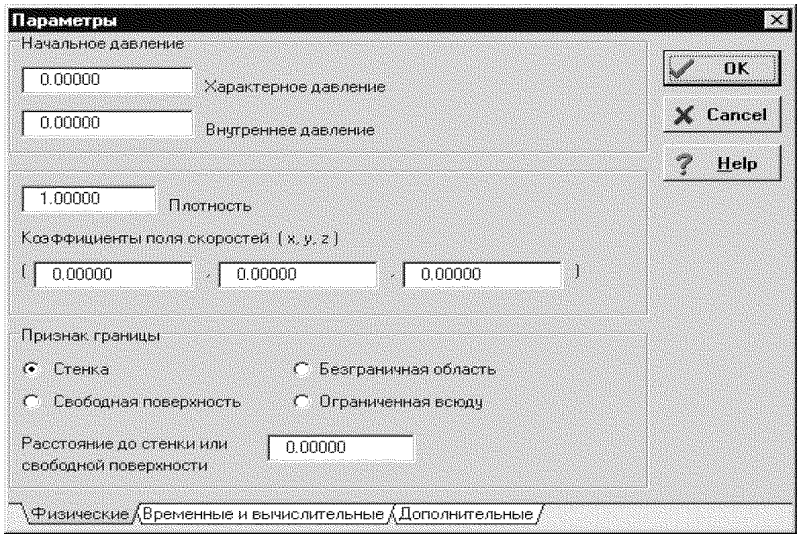

Рис. 5. Блокнот задания параметров.

2.4. Определение граничных условий

Режим задания граничных условий дает возможность пользователю выделить одну или целую группу точек для задания в них типа и значения граничных условий (первого или второго рода). Узлы с разными типами граничных условий отображаются на экране разными цветами. Уже установленные граничные условия отображаются на соответствующей панели после помещения указателя "мышки" в интересующий узел сетки.

2.5. Структура файлов входных и выходных данных

Рассмотрим важный момент — обмен данными как между блоками самого препроцессора, так и между препроцессором и решателем.

Входными данными для подпрограммы является текстовый файл, в котором перечислены треугольные и четырехугольные опорные зоны сетки. Опорная зона, в свою очередь, описывается координатами вершин, количеством точек разбиения на сторонах зоны и указанием типа сегмента поверхности (плоскости, эллипсоида или сферы). На основе файла описания опорных зон области блок построения сетки создает двоичный файл описания сетки и его текстовую копию, которая и служит входным файлом для препроцессора. Результатом работы препроцессора является файл описания задачи, который имеет определенный формат и служит входными данными для решателя (совместно с двоичным файлом описания сетки). Кроме того, этот файл может быть считан препроцессором для последующего его изменения.

Файл описания задачи также является текстовым и имеет блочную структуру:

— в блоке [SIZE] описывается количество узлов и элементов сетки;

— в блоке [PHYS] описываются физические параметры задачи;

— блок [TIME] задает временные параметры для нестационарной задачи;

— блок [OTHER] определяет режим расчета задачи, точность вычислений, а также способы интерпретации входных данных и вывода результатов;

— в блоке [COORD] перечисляются координаты узлов, типы и величины граничных условий в узлах сетки.

Все блоки являются обязательными, но в файле могут располагаться и распознаваться препроцессором и решателем в любом порядке. Таким образом, файл описания задачи полностью определяет как саму задачу, так и необходимые параметры для программы-решателя. Двоич-

ный файл описания сетки позволяет решателю создать представление сетки на основе массивов связности и произвести контроль правильности считывания координат узлов.

К основным недостаткам препроцессора можно отнести невозможность задания опорных зон в рамках программы препроцессора, что создает для пользователя ряд неудобств, а именно, необходимость "вручную" править файл описания опорных зон.

3. Заключение

Описанный препроцессор полностью снимает проблему ввода начальных данных для решения задачи. Существующая реализация использовалась в качестве практического приложения к специальному курсу "Методы граничных интегральных уравнений", который читается на математическом факультете КемГУ. Дальнейшее развитие препроцессора предполагает автоматизацию процесса задания входных данных для генерации сетки граничных элементов, совершенствование интерфейса пользователя, создание базы данных начальных параметров для расчета различных задач.

Список литературы

[1] Аммерал Л. *Принципы программирования в машинной графике.* СолСистем, М., 1992.

[2] Афанасьев К. Е., Гудов А. М., Долаев Р. Р., Коротков Г. Г. Пакет прикладных программ в курсе "Методы граничных интегральных уравнений". *Новые инфомационные технологии в университетском образовании: Сб. тр. Изд-во НИИ МИОО НГУ*, Новосибирск, 1998, 23–24.

[3] Афанасьев К. Е., Гудов А. М. Численное моделирование динамики пространственного пузыря методом граничных элементов. *Моделирование в механике: Сб. науч. тр.* Новосибирск, **7**, №1, 1993, 11–19.

[4] Афанасьев К. Е., Гудов А. М., Захаров Ю. Н. Исследование эволюции пространственного газового пузыря методом граничных элементов. В *"Вычислит. технологии"*. ИВТ СО РАН, Новосибирск, **1**, №3, 1992, 158–167.

[5] Гудов А. М. Численное моделирование возмущений свободной поверхности, вызванных коллапсом газового пузыря. *Там же*, **4**, №11, 1993, 92–103.

[6] Роджерс Д. *Алгоритмические основы машинной графики.* Мир, М., 1989.

[7] Роджерс Д. *Математические основы машинной графики.* Мир, М., 1980.

[8] Фоли Дж., вен Дем А. *Основы интерактивной машинной графики.* Мир, М., 1985.

[9] Brebbia C. A., Adey R. A. *Boundary Element Method Educational Package.* Wessex Inst. of Tecnology, Southampton, UK, 1994.

[10] Ghassemi F. Automatic mesh generation scheme. *Computing and Structures*, **15**, No. 6, 1982, 613–626.

Поступила в редакцию 7 декабря 1998 г.,
в переработанном виде 16 марта 1999 г.