Министерство образования и науки Украины

Севастопольский национальный технический университет

РАЗРАБОТКА DLL-БИБЛИОТЕК В MATLAB И ИСПОЛЬЗОВАНИЕ ИХ В ПРИЛОЖЕНИЯХ DELPHI

Методические указания
к выполнению лабораторной работы по дисциплине
"Разработка прикладного программного
обеспечения ЭВМ"
для студентов дневной формы обучения
направления 6.050201 – "Системная инженерия"

Севастополь

2010

УДК 004.4'23

Разработка DLL-библиотек в Matlab и использование их в приложениях Delphi: Методические указания к выполнению лабораторной работы по дисциплине "Разработка прикладного программного обеспечения ЭВМ" для студентов дневной формы обучения направления подготовки 6.50201 – "Системная инженерия" /Разраб. В.А. Карапетьян. – Севастополь: Изд-во СевНТУ, 2010. – 32 с.

В методических указаниях приводится описание лабораторной работы, предназначенной для изучения и практического применения технологии разработки DLL-библиотек в системе Matlab и использования их при программировании Delphi-приложений.

Методические указания предназначены для студентов дневной формы обучения по направлению подготовки 6.050201 – "Системная инженерия".

Методические указания рассмотрены и утверждены

на заседании кафедры Технической кибернетики,

протокол № 1 от " 17 " сентября 2010 г.

Допущено учебно-методическим центром СевНТУ

в качестве методических указаний.

Рецензент Крамарь В.А., проректор СевНТУ.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Изучение и применение возможностей организации динамически компонуемых библиотек (DLL-библиотек) в системе Matlab с целью их использования в приложениях, разработанных в Delphi.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Matlab – мощная многофункциональная интегрированная система автоматизации научно-технических расчетов и моделирования динамических систем и процессов [1, 2]. В состав системы Matlab, помимо собственно математического ядра, специализированных программных пакетов (toolboxes) и пакета визуального моделирования Simulink со своими расширениями (blockset), входят также несколько предметно-ориентированных утилит с графическим интерфейсом пользователя (GUI-утилиты) и специальные инструментальные программные средства для организации взаимодействия Matlab с различными системами программирования. Одним из таких инструментальных средств является Matlab Compiler – специальный компилятор, позволяющий, в частности, создавать из m-файлов автономные консольные приложения и динамически компонуемые библиотеки (DLL-библиотеки, библиотеки совместного использования shared library) [3 – 6].

DLL-библиотеки являются неотъемлемой частью ОС Windows, которая документирует и поддерживает механизмы обращения пользовательских приложений к ресурсам этих библиотек. Все современные системы программирования, работающие под ОС Windows, имеют средства создания DLL-библиотек. Библиотека обычно создаётся как отдельный проект в системе программирования и в соответствии с оригинальным правилам программирования функций этих библиотек на основном языке системы программирования.

Создание собственных m-функций в системе Matlab, с одной стороны, не представляет большой сложности для прикладного программиста, с другой стороны – программист имеет возможность использовать всю мощь математики ядра Matlab и практически всех специализированных пакетов (toolboxes). Организация нескольких m-функций в стандартную DLL-библиотеку с помощью инструментальной службы Matlab Compiler позволяет применять эти библиотеки обычным, для прикладных Windows-приложений, образом. Т.е., появляется еще одна возможность [8] использовать мощные математические возможности системы Matlab в программах прикладного назначения.

2.1. Основные сведения о компиляторе Matlab Compiler

Практически каждая новая версия системы Matlab содержит и новую версию Компилятора. Иногда переход к очередной версии Компилятора сопровождается значительными изменениями в работе, как по форме команд, так и по содержанию создаваемых программных объектов [4, 6]. Последние значительные изменения Компилятора произошли при переходе к версиям Matlab 7.x. Начиная с версии Matlab R14, поставляется Matlab Compiler 4. В дальнейшем изменения были не столь существенными, однако и здесь при переходе к более старшей версии требуется перекомпиляция старых версий DLL-библиотек. В настоящее время в версию Matlab 2010a включен Компилятор Matlab Compiler 4.13 [7].

Компилятор Matlab Compiler устанавливается автоматически в процессе стандартной инсталляции системы Matlab. Если Matlab устанавливается под управлением пользователя, следует явно указать установку компоненты Matlab Compiler.

Для работы Компилятора необходимо иметь инсталлированные под операционной системой компиляторы с языка C или C++, поддерживаемые Matlab. В частности, этими компиляторами могут быть компиляторы систем программирования Microsoft Visual C/C++ версий 6.x и выше. Полный перечень компиляторов, поддерживаемых Matlab 7.10 (в том числе и 64-разрядных) представлен на странице http://www.mathworks.com/support/compilers/R2010a/index.html . Если предполагается создавать только 32-разрядные Windows-приложения или DLL-библиотеки, то достаточно иметь включенный в Matlab компилятор Lcc C (у 64-битных версий Matlab встроенного компилятора с языка C уже нет).

Настройка (конфигурирование) Компилятора Matlab Compiler на конкретный языковый компилятор проводится с помощью консольной утилиты mbuild. Эта утилита служит как для выбора или смены внешнего компилятора с языков C/C++, так и для создания автономных консольных приложений. Если в системе установлено несколько языковых компиляторов, поддерживаемых текущей версией Matlab, то с помощью утилиты mbuild можно в любой момент сменить настройку Matlab Compiler на конкретный языковый компилятор.

Выбор/замена внешнего компилятора с языков C/C++ осуществляется консольной командой **mbuild -setup**

Выполнить данную команду можно любым известным способом исполнения консольной команды операционной системы. Например, в ОС Windows XP следует нажать кнопку главного меню "Пуск" и выбрать пункт меню "Выполнить". Затем, в строке набора команд открывшегося окна "Запуск программы", следует набрать команду mbuild -setup и нажать Enter (или просто нажать кнопку выбора OK).

Утилита mbuild определит все установленные в ОС компиляторы с языков C/C++ с учетом и компилятора Lcc, примет подтверждение (y) от пользователя о его желании установить языковый компилятор и выведет нумерованный список найденных компиляторов в окно консоли. Пользователю следует выбрать (**Select a compiler:**) один из приведенных языковых компиляторов, введя с клавиатуры соответствующее положительное целое число. Отказаться от выбора компилятора можно введя цифру ноль. Результаты настройки сохраняются в специальном файле опций compopts.bat, который находится в пользовательском каталоге

C:\Documents and Settings\UserName\Application Data\MathWorks\MATLAB\R2010a

Работать с Компилятором Matlab Compiler можно в двух режимах – консольном и графическом.

Для вызова консольного режима работы Компилятора Matlab Compiler применяется программа mcc с необходимым набором опций и ключей. Она может быть вызвана обычным для операционной системы образом (Пуск→Выполнить…). Для работы в консольном режиме пользователь естественно должен знать наборы ключей и опций для правильной настройки команды mcc на необходимые действия.

Начиная с версии Matlab 2006b, для работы с компилятором может применяться интерактивная пользовательская графическая утилита Deployment Tool (инструмент развёртывания) [6, 7].

2.2. Среда разработки Deployment Tool

Среда разработки Deployment Tool предназначена для создания консольных приложений или DLL-библиотек и позволяет формировать инсталляционные пакеты для их распространения [6, 7].

Вызывается Среда разработки командой Matlab **deploytool**

Среда разработки открывается как встраиваемое окно справа от командного окна Matlab и в строке меню Matlab добавляется элемент меню Project. В последней версии Matlab 2010a интерфейс утилиты Deployment Tool изменён [7].

Окно Deployment Tool можно отстыковать (кнопка Undock в группе системных кнопок окна Среды) от командного окна Matlab и сделать свободно перемещаемым (рисунок 2.1). В этом случае у свободно перемещаемого окна Среды разработки автоматически появляется строка главного меню, а элемент меню Project удаляется из строки главного меню окна Matlab. Главное меню Среды разработки и инструментальная панель содержат все необходимые опции и кнопки для: создания нового (New Project); открытия существующего (Open Project) или сохранения проекта (Save Project); добавления (Add File) или удаления (Remove) файлов из проекта; построения программного компонента (Build); создания инсталляционного пакета (Package) и некоторые др.

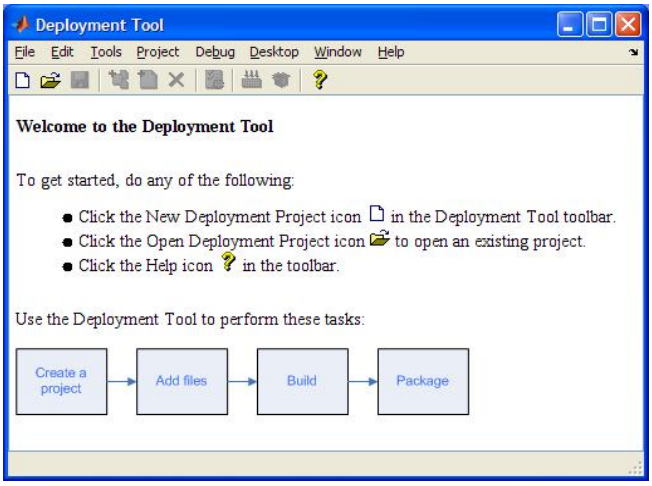


Рисунок 2.1 – Окно Среды разработки Deployment Tool

2.3. Создание DLL-библиотеки в среде Deployment Tool

Рассмотрим иллюстративный пример создания библиотеки совместного использования (DLL-библиотеки) из нескольких Matlab-функций, оформленных в виде m-файлов. Пусть в каталоге D:\Temp имеется десять Matlab-функций в виде стандартных m-файлов. Информация об этих функциях приведена в таблице 2.1.

Таблица 2.1 – Имена, назначения и тексты m-функций для DLL-библиотеки

Имя функции	Назначение функции	Текст функции
Plusvak	Сложение матриц	function a = Plusvak(a1, a2) a = a1 + a2;
Minusvak	Вычитание матриц	function a = Minusvak(a1, a2) a = a1 - a2;
Mtimesvak	Умножение матриц	function a = Mtimesvak(a1, a2) a = a1 * a2;
Detvak	Вычисление определителя матрицы	function d = Detvak(a) d = det(a);
Eigvak	Вычисление собственных чисел и собственных векторов матрицы	function [c,v] = Eigvak(a1) [c,v] = eig(a1);
Invvak	Обращение матрицы	function a = Invvak(a1) a = inv(a1);
Polyvak	Вычисление коэффициентов характеристического полинома матрицы	function d = Polyvak(a) d = poly(a);
Rankvak	Вычисление ранга матрицы	function d = Rankvak(a) d = rank(a);
Sinvak	Вычисление значений синусов от элементов матрицы	function d = Sinvak(a) d = sin(a);

Stepvak	Вычисление и построение единичной переходной характеристики	function o = Stepvak(a)
		sys = ss([0 1; -2 -3],[0;1],eye(2),[0;0]);
		[Y,T] = step(sys);
		plot(T,Y); grid on; o=1;

Создавать DLL-библиотеку будем в Среде разработки Deployment Tool:

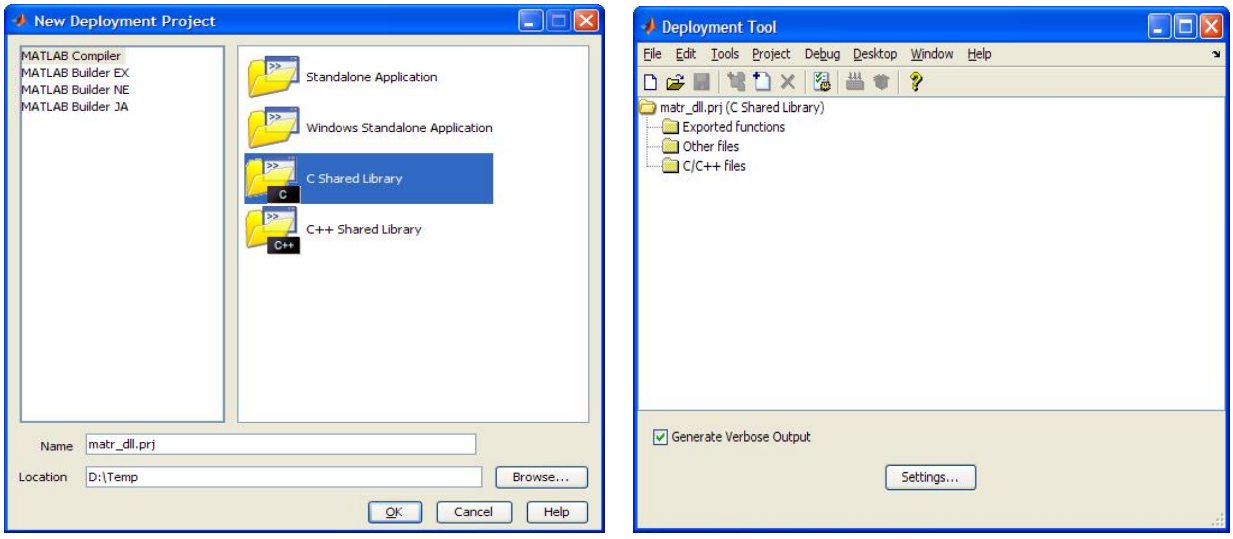
– загрузим Matlab;

– вызовем Среду разработки командой **`>> deploytool`**

– отстыкуем окно Среды от основного окна Matlab (рисунок 2.1).

Для создания новой библиотеки (нового проекта) щёлкнем на кнопке инструментальной панели Create a new deployment project или выберем опцию New Deployment Project в меню File главного меню окна Среды разработки. В результате этих действий откроется окно New Deployment Project (рисунок 2.2 а). Выберем в левой части этого окна инструмент разработки MATLAB Compiler, а в правой – тип создаваемого проекта (C Shared Library). В поле Name введем имя файла нашего проекта matr_dll.prj . В поле Location укажем каталог D:\Temp, в котором будут сохранены результаты всех последующих действий Среды разработки и Компилятора. Напоминаем, что в этот каталог нами помещены и все m-файлы Matlab-функций, которые будут конвертированы в DLL-библиотеку.

После всех действий в окне нового проекта (рисунок 2.2 а) и принятия установок (кнопка OK) происходит возврат в окно Среды разработки (рисунок 2.2 б):



а) б)

Рисунок 2.2 – Окна нового проекта C Shared Library и Среды

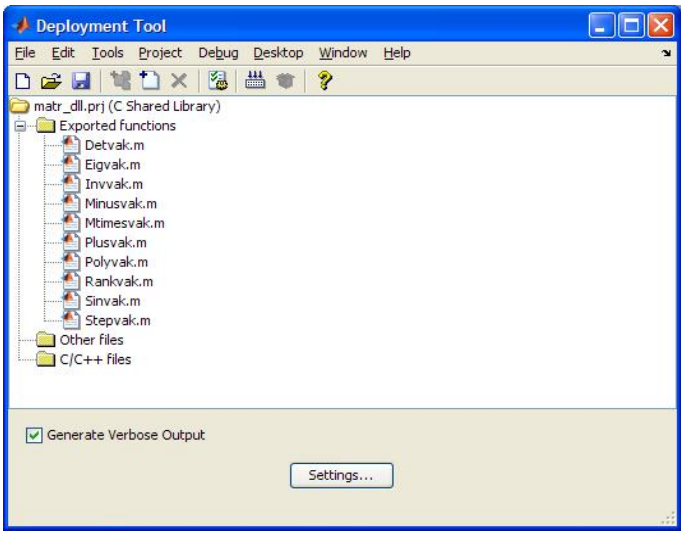


Рисунок 2.3 – Проект с добавленными m-файлами Matlab-функций

Содержание окна Среды разработки изменилось (рисунок 2.2 б). Стали доступными кнопки добавления файлов в проект и изменения настроек проекта .

Теперь следует включить m-файлы функций в проект. С помощью кнопки Add File добавим все десять m-файлов из каталога D:\Temp в проект. Как следует из содержимого окна Среды к проекту можно также присоединить файлы и других типов. В результате последней операции окно Среды примет вид, представленный на рисунке 2.3.

После добавления файлов в проект DLL-библиотеки стала доступной кнопка построения проекта библиотеки – Build the project . При нажатии на эту кнопку открывается окно вывода протокола Deployment Tool Output и начинается процесс построения DLL-библиотеки. Длительность процесса может достигать нескольких минут в зависимости от объема проекта, версии языкового компилятора и Компилятора Matlab, мощности компьютера. В окно выхода Output выводится довольно объёмный текстовый протокол процесса построения библиотеки. Последние фразы протокола приведены в окне вывода на рисунке 2.4.

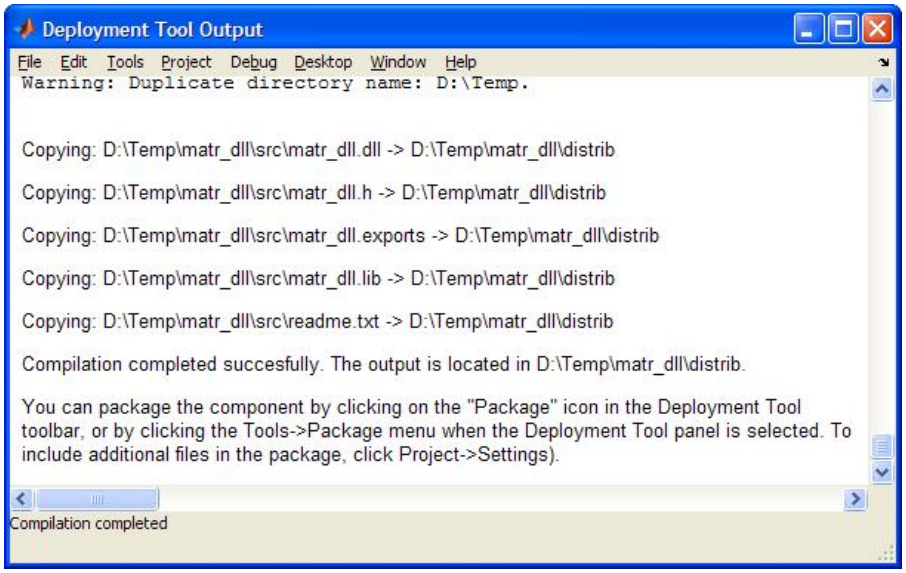


Рисунок 2.4 – Окно вывода протокола построения библиотеки

В результате построения библиотеки в каталоге D:\Temp организуются файл проекта matr_dll.prj и папка [matr_dll]. В папке [matr_dll] находятся две дочерние папки – [distrib] и [src]. В каждой из них, в том числе, присутствует файл DLL-библиотеки matr_dll.dll с нашими десятью функциями. Размер файла в Matlab 2008a более трёх мегабайт. Для версии Matlab 2010a размер dll-файла этой же библиотеки составляет уже около пяти мегабайт.

Если требуется подготовить программный пакет для установки DLL-библиотеки на компьютере, где Matlab не установлен, то следуя последней фразе текста в окне на рисунке 2.4, необходимо создать инсталляционный пакет с помощью кнопки Package . В результате этих действий в папку [distrib] будут добавлены ещё два файла – matr_dll_pkg.exe и _install.bat. Первый файл – архив, содержащий файлы оригинальной папки [distrib]. Второй файл – командный файл для проверки и установки Среды выполнения компонент Компилятора Matlab – Matlab MCR (Matlab Component Runtime).

2.4. Среда выполнения компонентов Matlab – MCR

Более ранние версии Компилятора Matlab использовали тот факт, что все математические функции ядра Matlab находились в виде обычных DLL-библиотек в нескольких отдельных файлах (математическая библиотека C/C++ Matlab) [4, 9]. Доступ к функциям этих библиотек осуществлялся на основе стандартного механизма, поддерживаемого операционной системой Windows. Это либо статическая компоновка DLL-библиотеки на этапе компиляции приложения, либо динамическая

компоновка уже в процессе работы приложения. В работе [9] приведен пример работы с DLL-библиотеками системы Matlab 6.5 в указанном выше контексте.

Начиная с седьмой версии Matlab, механизм работы с DLL-библиотеками был принципиально изменен. Математическая библиотека C/C++ Matlab была заменена на *Среду выполнения компонентов Matlab* – MCR (Matlab Component Runtime). Среда MCR представляет собой специальный набор библиотек, служб и сервисов, обеспечивающих штатную работу компонентов Matlab.

MCR поддерживает более широкий круг задач – она обеспечивает работу не только с библиотеками общего доступа (DLL), но и исполнение компонентов Matlab для использования в языках программирования C++, Java, VBA для Excel, C# и других допустимых языках платформы .Net Framework [6].

Для установки Среды MCR служит специальная утилита MCRInstaller.exe. Этот исполняемый файл находится в каталоге

```
C:\Program Files\MATLAB\R2008a\toolbox\compiler\deploy\win32\
```

Размер файла для Matlab R2008a составляет около 240 Мб, а для версии Matlab R2010a – 173 Мб. Среда MCR инсталлируется в операционной системе и файлы Среды размещается в каталоге

```
C:\Program Files\MATLAB\MATLAB Compiler Runtime\v78\
```

Имя каталога v78 формируется из номера последнего семейства версий Matlab (Matlab **7**.x) и номера последней версии Компилятора Matlab 4.**8**. Для версии Matlab 2010a соответствующая папка имеет имя v713. Объём установленной среды MCR составляет около 430 Мб.

Таким образом, для установки DLL-библиотеки на другой компьютер необходимо выполнить следующие операции [6]:

– запустить MCRInstaller.exe для установки Среды MCR, необходимой для работы DLL-библиотеки без Matlab. Среда MCR используется всеми установленными компонентами, созданными Компилятором Matlab;

– распаковать инсталляционный пакет с созданным программным обеспечением (в нашем случае это matr_dll_pkg.exe) в выбранный каталог (например, в C:\My_MatlabDLL).

После установки MCR все её DLL-библиотеки и службы регистрируются в операционной системе и становятся доступными приложениям.

2.5. Создание компонентов Matlab командой mcc

Библиотеку совместного использования (dll) можно создать и в командном режиме без использования Среды разработки Deployment Tool. Для этого предназначена утилита mcc. Например, наша библиотека matr_dll.dll из десяти m-файлов может быть сформирована следующей командой

```
mcc -B csharedlib:matr_dll plusvak  minusvak  mtimesvak
    detvak invvak eigvak sinvak rankvak polyvak stepvak
```

В результате работы этой команды будет создан файл DLL-библиотеки matr_dll.dll, а также несколько вспомогательных файлов (заголовочный, файл экспорта, библиотечный, файл протокола) и файл проекта matr_dll.prj. Файл проекта

может быть использован в дальнейшем для модификации библиотеки в среде разработки Deployment Tool.

Утилита mcc может выполнять широкий спектр функций. Настройка mcc на исполнение конкретной функции задается соответствующим набором опций и ключей [4 – 7].

Приведем пример создания простого консольного приложения. Пусть необходимо вычислять квадрат нормы вектора произвольной размерности. Соответствующая m-функция может иметь следующий вид

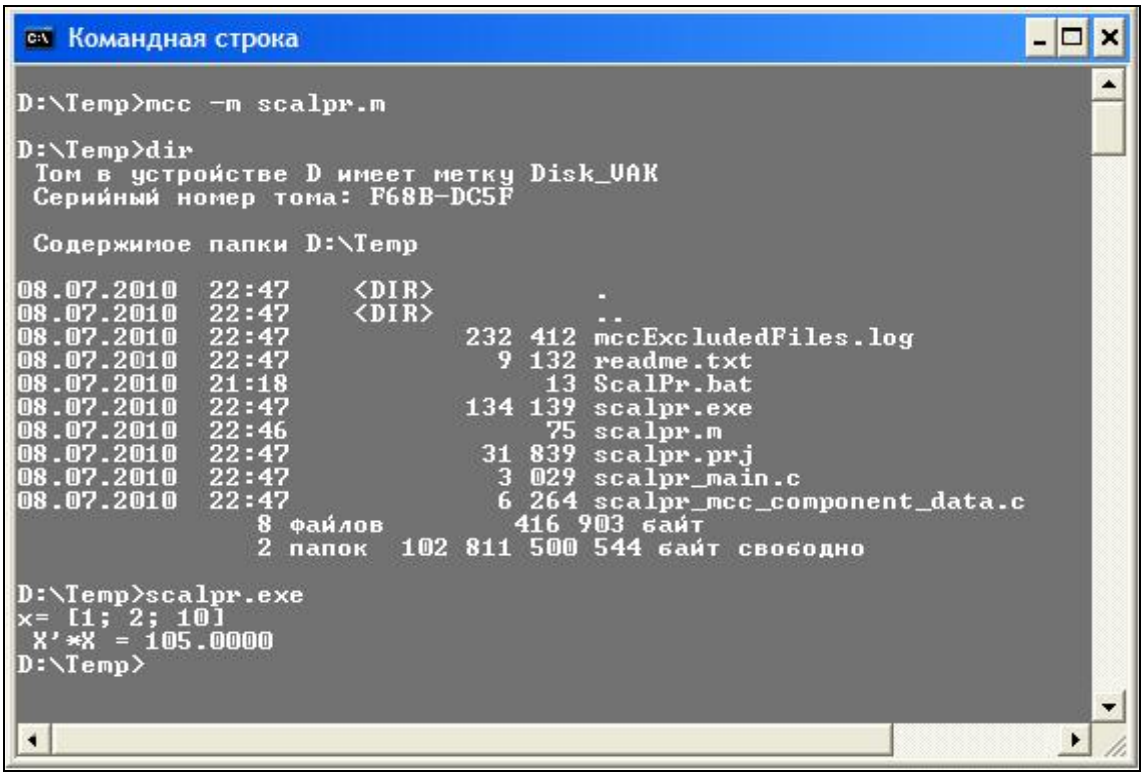
```
function y=scalpr
x=input('x= ');
y=x'*x;
fprintf(1,' X''*X = %1.4f',y);
```

Текст этой m-функции сохраним в файле scalpr.m. При запуске данной функции в Matlab пользователь должен задать (ввести) вектор-столбец в контексте языка Matlab, т.е., например, в виде [1; 2;10]. После вычислений функция выведет результат в виде строки текста: **X'*X = 105.0000** .

Чтобы получить исполняемый exe-файл с данной программой необходимо вызвать утилиту mcc со следующим набором опций

```
mcc -m scalpr
```

На рисунке 2.5 приведен текстовый протокол операций создания и выполнения консольного приложения:



D:\Temp>mcc -m scalpr.m	– создание консольной программы;
D:\Temp>dir	– распечатка содержимого каталога D:\Temp;
D:\Temp>scalpr.exe	– запуск консольного приложения;
x= [1; 2; 10]	– ввод данных в ответ на приглашение;
X'*X = 105.0000	– вывод программой результатов счета.

Рисунок 2.5 – Протокол работы с консольным приложением

2.6. Особенности работы с DLL-библиотекой

Прокомментируем здесь следующие особенности работы с DLL-библиотекой, созданной Компилятором Matlab:

а) соглашения о составе библиотеки и именах функций;

б) состав и структура формальных параметров функций библиотеки;

в) представление заголовков функций библиотеки в нотации Delphi;

г) предварительные операции перед первым вызовом функций Matlab.

а) Соглашения о составе библиотеки и именах функций.

Имена всех функций, экспортируемых DLL-библиотекой, состоят из префикса _mlf и собственно имени m-функции (точнее – имени m-файла, в котором находится текст m-функции). В состав библиотеки автоматически включаются и две специальные функции для выполнения операций инициализации библиотеки – **`_matr_dllInitialize`** и **`_matr_dllTerminate`**. Как видим имена этих функций формируются из имени библиотеки _matr_dll и слов Initialize и Terminate. Соответствующие заголовки функций представлены в тексте модуля связи Un_DLL_MX в Приложении А в строках 18÷29.

б) Состав и структура формальных параметров функций библиотеки.

Созданная Компилятором Matlab наша DLL-библиотека организована как библиотека совместного доступа в контексте языка C. Экспортирует эта библиотека C-функции, т.е., в частности, произвольная экспортируемая функция может получать произвольное количество параметров, но возвращать не более одного параметра. С другой стороны, функции Matlab, оформленные в виде m-файлов, могут получать исходные данные в виде списка входных параметров и возвращать результаты своей работы также в виде списка из *нескольких* выходных параметров [1]. Поэтому необходимо было разрешить это несоответствие в синтаксисе функций языка Matlab и языка C.

Прошлые версии Компилятора формировали заголовки экспортируемых функций следующим образом. Если m-функция возвращает не более одного параметра, то никакой проблемы с её аналогом в виде C-функции не возникает. Если m-функция должна возвращать более одного параметра (в нашем случае это функция Eigvak), то соответствующая C-функция возвращает один выходной параметр через своё имя, а остальные – через список обычных параметров функции.

Начиная с версии Matlab R14 (Matlab 7.x), Компилятор Matlab (Matlab Compiler 4.x) изменил интерфейс экспортируемых mlf-функций из библиотек совместного использования. Теперь эти C-функции не возвращают никаких значений (void) через свое имя, а весь обмен данными ведется через обычные параметры функций. Поэтому для конвертации заголовков таких C-функций в подпрограммы языка Delphi наиболее подходит такая структура как процедура (procedure).

Список параметров экспортируемых из DLL-библиотек C-функций состоит из следующих по порядку объектов:

– целочисленный параметр – фактически определяющий количество выходных параметров соответствующей m-функции;

– параметры-переменные, в количестве, соответствующем числу выходных параметров m-функции;

– параметры-значения, в количестве, соответствующем числу входных параметров m-функции.

в) Представление заголовков функций библиотеки в нотации Delphi.

В соответствии с требованиями, описанными в предыдущем пункте, в нотации Delphi все функции нашей DLL-библиотеки проще всего представлять в виде процедур языка Delphi. В качестве примера рассмотрим функцию для вычисления собственных чисел и собственных векторов произвольной квадратной матрицы. Текст этой m-функции приведен в таблице 2.1

```
function [c,v] = Eigvak(a1)
  [c,v] = eig(a1);
```

Здесь используется функция eig в контексте возвращения ею двух объектов – [c, v], т.е., квадратной матрицы c из собственных векторов и v – диагональной матрицы из собственных чисел матрицы a1.

Заголовок нашей процедуры **_mlfEigvak** в модуле связи Un_DLL_MX имеет следующий вид

```
procedure _mlfEigvak(i:Integer; var pout1,pout2:pmxArray;
                     A:pmxArray); cdecl; external 'matr_dll.dll';
```

Первый целочисленный параметр-значение **i:Integer** – число, фактически показывающее количество выходных параметров в нашей m-функции Eigvak (два параметра – c и v).

Два последующих параметра-переменных **var pout1,pout2:pmxArray** – два выходных параметра типа pmxArray для получения матриц c и v.

Последний параметр-значение **A:pmxArray** – единственный входной параметр нашей функции типа pmxArray для представления исходной матрицы a1.

Еще одна существенная особенность проявляется при включении в DLL-библиотеку математических функций Matlab, которые требуют для своего исполнения дополнительных m-функций. Например, чтобы решить систему дифференциальных уравнений с помощью функции ode45 необходимо подготовить отдельную m-функцию для вычисления правых частей этой системы [1, 2]. Аналогичные ситуации возникают и в других случаях – при вычислении определенных интегралов, нахождении корней нелинейных уравнений, решении задач оптимизации и др. Основная проблема здесь – это передача данных из вызывающей функции во вспомогательную функцию. Данные можно передавать либо через глобальные переменные, либо через обычные параметры функций. Механизм глобальных переменных реализовать в нашем случае (при разработке DLL-библиотеки из m-файлов функций) не представляется возможным. Остается второй путь – передавать данные через параметры функций. Здесь нужна особая аккуратность, так как многие функции Matlab могут использовать переменное количество параметров. Для иллюстрации возможных действий в данном случае в Приложении B приведен пример Delphi-программы для решения стационарной линейной системы дифференциальных уравнений заданной в нормальной форме.

г) Предварительные операции перед первым вызовом функций Matlab.

Для работы с компонентами Matlab, созданными Компилятором (в нашем случае – с DLL-библиотекой), из Delphi-приложения необходимо инициализировать

Среду выполнения компонентов MCR. Данная операция инициализация производится с помощью вызова процедуры

```
mclInitializeApplication(0,0);
```

Она вызывается перед первым обращением Delphi-приложения к функциям API Matlab и к любым другим функциям, созданным Компилятором Matlab.

После инициализации Среды MCR следует инициализировать нашу собственную DLL-библиотеку matr_dll.dll с помощью функции

```
_matr_dllInitialize;
```

Если Delphi-приложение использует несколько DLL-библиотек, то каждая библиотека должна быть инициализирована отдельно своей собственной функцией инициализации

```
<_libname>Initialize;
```

Здесь префикс **_libname** представляет собственно имя файла соответствующей DLL-библиотеки.

Перед окончанием работы Delphi-приложения необходимо в обратном порядке провести деинициализацию пользовательских DLL-библиотек и, затем, Среды выполнения MCR. Данные операции осуществляются соответственно функциями

```
_matr_dllTerminate;
```
и
```
mclTerminateApplication;
```

Функции инициализации среды выполнения MCR находятся в стандартной dll-библиотеке Matlab в файле **mclmcr.dll**.

Функции инициализации нашей DLL-библиотеки находятся в самой этой библиотеке и имеют следующие имена:

```
_matr_dllInitialize;
```
и
```
_matr_dllTerminate;
```

Применение функций инициализации/деинициализации продемонстрировано в тексте модуля Un_MX формы иллюстративной программы в Приложении Б в строках 59, 60 и 135, 136.

3. ЗАДАНИЕ НА РАБОТУ

Задание студенту на разработку Delphi-приложения формулируется, конкретизируется и выдается преподавателем. Приложение должно обладать функционально необходимым и интуитивно понятным интерфейсом и визуально представлять в числовой и/или графической форме, как исходные данные задачи, так и результаты вычислений. Все *объёмные* необходимые исходные данные к решаемой задаче рекомендуется генерировать в сам***о***м приложении.

Примерный перечень заданий

Разработать Delphi-приложение, которое реализует:

1. Решение системы линейных алгебраических уравнений: $A \cdot x = b$;
2. Операции векторно-матричной алгебры:

$$A \pm B;\quad A \cdot B;\quad A^{-1};\quad A^{T};\quad A \cdot x;\quad x^{T} \cdot A;\quad x \pm y;\quad x^{T} \cdot y;$$

3. Вычисление характеристик квадратной матрицы: определитель; ранг; норма; собственные числа и собственные векторы; коэффициенты характеристического полинома;

4. Операции с полиномами: $a(x) \pm b(x)$; $a(x) \cdot b(x)$; $a(x) / b(x)$; нахождение нулей полинома; определение коэффициентов полинома по его нулям;

5. Полиномиальное сглаживание по методу наименьших квадратов одномерного массива данных;

6. Интерполяцию сплайнами одномерного массива данных;

7. Вычисление определенного интеграла;

8. Решение системы обыкновенных линейных дифференциальных уравнений с постоянными коэффициентами;

9. Вычисление статистических характеристик одномерного массива отсчетов стационарного случайного процесса;

10. Вычисление амплитудного спектра полигармонического сигнала;

11. Решение задачи анализа устойчивости объекта управления, заданного своей моделью в пространстве состояний;

12. Решение задачи анализа устойчивости объекта управления, заданного своей передаточной функцией;

13. Решение задачи анализа устойчивости замкнутой САУ, заданной передаточной функцией разомкнутого контура;

14. Решение задачи анализа управляемости объекта, заданного своей моделью в пространстве состояний;

15. Решение задачи анализа наблюдаемости объекта, заданного своей моделью в пространстве состояний;

16. Построение единичной переходной функции $h(t)$ для объекта управления, заданного моделью в пространстве состояний;

17. Построение импульсной переходной функции $g(t)$ для объекта управления, заданного моделью в пространстве состояний;

18. Построение единичной переходной функции $h(t)$ для объекта управления, заданного своей передаточной функцией;

19. Построение импульсной переходной функции $g(t)$ для объекта управления, заданного своей передаточной функцией;

20. Построение собственного движения для объекта управления, заданного моделью в пространстве состояний;

21. Построение амплитудно-частотной характеристики для объекта управления, заданного своей передаточной функцией;

22. Построение частотного годографа Найквиста для объекта управления, заданного своей передаточной функцией;

23. Построение частотного годографа Михайлова для объекта управления, заданного своей передаточной функцией;

24. Изображение на комплексной плоскости нулей и полюсов передаточной функции объекта управления;

25. Изображение на комплексной плоскости собственных чисел матрицы динамических коэффициентов A объекта управления.

4. СОДЕРЖАНИЕ ОТЧЕТА И ПОРЯДОК ЗАЩИТЫ РАБОТЫ

В отчете по лабораторной работе должны быть отражены цель, необходимые сведения о методах решения рассматриваемых задач и средствах системы Matlab, тексты программ, численные и графические результаты расчетов и построений, выводы по работе.

Защита результатов лабораторной работы производится за компьютером при наличии работающей программы и полностью оформленного отчета. В ходе защиты студент может получить дополнительные практические задания по существу работы.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сформулируйте своими словами цель лабораторной работы.

2. Опишите последовательность действий при создании DLL-библиотеки.

3. Прокомментируйте структуру и назначение формальных параметров произвольной функций DLL-библиотеки.

4. Как Вы понимаете назначение Среды MCR?

5. Каким образом приложение инициализирует Среду MCR?

6. Каким образом приложение инициализирует DLL-библиотеку?

7. В чем принципиальное отличие предлагаемого здесь способа использования средств Matlab от технологии Matlab Engine?

8. Почему программа в Приложении В предназначена исключительно для решения однородных систем ОДУ?

9. Объясните наличие и значение четвертого параметра в функции ode45 в m-файле Odevak (страница 25).

10. Поясните перечень и назначение всех параметров процедуры Odevak из Приложения В.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ануфриев И.Е. Matlab 7 /И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1104 с.
2. Лазарев Ю.Ф. Моделирование процессов и систем в Matlab. Учебный курс /Ю.Ф. Лазарев. – СПб.: Питер; Киев: BHV, 2005. – 512 с.
3. Мартынов Н.Н. MATLAB 7. Элементарное введение /Н.Н. Мартынов. – М.: КУДИЦ-ОБРАЗ, 2005. – 416 с.
4. Подкур М.Л. Программирование в среде Borland C++ Builder с математическими библиотеками Matlab C/C++ /М.Л. Подкур, П.Н. Подкур, Н.К. Смоленцев. – М.: ДМК Пресс, 2006. – 496 с.
5. Бей И. Взаимодействие разноязыковых программ / И. Бей. – М.: Издательский дом "Вильямс", 2005. – 880 с.
6. Смоленцев Н.К. MATLAB и программирование на Visual C#, Borland JBuilder, VBA: Учебный курс (+CD) / Н.К. Смоленцев. – М.: ДМК Пресс; СПб.: Питер, 2009. – 464 с.
7. MATLAB® Compiler™ User’s Guide. – The MathWorks, Inc., 2010. – 474 p.
8. Использование технологии Matlab Engine в приложениях Delphi: Методич. указ. к выполнению лаб. работы по дисциплине "Разработка ППО ЭВМ" для студентов дневной формы обучения направления подготовки 6.0914 "Компьютеризированные системы, автоматика и управление" /Разраб. В.А. Карапетьян. – Севастополь: Изд-во СевНТУ, 2005. – 18 с.
9. Использование библиотек Matlab в приложениях Delphi: Методич. указ. к выполнению лаб. работы по дисциплине "Разработка ППО ЭВМ" для студентов дневной формы обучения направления подготовки 6.0914 "Компьютеризированные системы, автоматика и управление" /Разраб. В.А. Карапетьян. – Севастополь: Изд-во СевНТУ, 2005. – 17 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Модуль связи Delphi-приложения и DLL-библиотеки

Ниже приведен листинг модуля связи Delphi-приложения с компонентами Matlab. Модуль собственно и содержит все необходимые программные объекты для реализации межпрограммного интерфейса Delphi и библиотек Matlab, созданных в Компиляторе Matlab.

В этом модуле определены все необходимые для наших задач типы данных (строки 3÷8):

– указатель на вещественное число pDouble;

– перечислимый тип для работы с вещественными и комплексными числами mxComplexty;

– тип "запись" для представления структуры произвольного массива данных mxArray;

– указатель на этот тип pmxArray.

Объявлены заголовки некоторых функций и процедур из стандартной библиотеки libmx.dll для формирования и доступа к массивам данных в формате системы Matlab (строки 9÷15).

В строках 16 и 17 приведены заголовки функций инициализации среды MCR для выполнения компонентов Matlab. Эти две функции находятся в стандартной библиотеке mclmcr.dll.

Заголовки десяти наших процедур и функций из библиотеки matr_dll.dll приведены в строках 20÷29. В этой же библиотеке находятся две функции инициализации нашей DLL-библиотеки (строки 18 и 19).

В объявлениях всех функций и процедур модуля указан механизм их взаимодействия с Delphi-приложением – декларация cdecl. Это объявление необходимо для корректного взаимодействия основной программы и вызываемых функций, т.к. тела всех функций и процедур модуля написаны на языке программирования C.

Текст модуля связи (интерфейсного модуля)

```
{1} unit Un_DLL_MX;
{2} interface
{3} type
{4}   pDouble     = ^Double;
{5}   mxComplexty = (mxREAL, mxCOMPLEX);
{6}   pmxArray    = ^mxArray;
{7}   mxArray     = record
                      { Просто обозначить этот тип }
{8}                 end;

    {*************** Из библиотеки libmx.dll ***************}
{9} function  mxCreateDoubleMatrix(m,n:Integer;
              mxData:mxComplexty):pmxArray;   cdecl; external 'libmx.dll';

{10}procedure mxSetName(arr_ptr:pmxArray; const name:String);
                                        cdecl; external 'libmx.dll';

{11}function  mxGetPr(arr_ptr:pmxArray):pDouble;cdecl; external 'libmx.dll';
{12}function  mxGetPi(arr_ptr:pmxArray):pDouble;cdecl; external 'libmx.dll';
{13}procedure mxDestroyArray(arr_ptr:pmxArray); cdecl; external 'libmx.dll';
```

```
{14}procedure mxFree(var ram:THandle);          cdecl; external 'libmx.dll';
{15}function  mxTranspose(x:pmxArray):pmxArray; cdecl; external 'libmx.dll';

    {*************** Из библиотеки mclmcr.dll ***************}
{16}function mclInitializeApplication(A:THandle;B:Integer):Boolean;
                                              cdecl; external 'mclmcr.dll';
{17}function mclTerminateApplication:Boolean;   cdecl; external 'mclmcr.dll';

    {******** Наша dll-библиотека matr_dll.dll из функций Matlab *******}
{18}function  _matr_dllInitialize:Boolean;  cdecl; external 'matr_dll.dll';
{19}function  _matr_dllTerminate :Boolean;  cdecl; external 'matr_dll.dll';
{20}procedure _mlfMinusvak(i:Integer;var pout:pmxArray;A,B:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{21}procedure _mlfMtimesvak(i:Integer;var pout:pmxArray;A,B:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{22}procedure _mlfDetvak(i:Integer;var pout:pmxArray;A:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{23}procedure _mlfInvvak(i:Integer;var pout:pmxArray;A:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{24}procedure _mlfEigvak(i:Integer; var pout1,pout2:pmxArray;A:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{25}procedure _mlfSinvak(i:Integer; var pout:pmxArray;  A:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{26}procedure _mlfRankvak(i:Integer; var pout:pmxArray; A:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{27}procedure _mlfPolyvak(i:Integer; var pout:pmxArray; A:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{28}procedure _mlfPlusvak(i:Integer; var pout:pmxArray; A,B:pmxArray);
                                            cdecl; external 'matr_dll.dll';
{29}procedure _mlfStepvak(i:Integer; var pout:pmxArray; A:pmxArray);
                                            cdecl; external 'matr_dll.dll';

{30}implementation
{31}end.
```

Функции из стандартной библиотеки libmx.dll предназначены для манипуляций с массивами данных Matlab. Их назначения, перечни параметров, варианты использования подробно описаны в [3 – 9]. Функциональные назначения процедур из нашей DLL-библиотеки matr_dll.dll описаны в таблице 2.1.

Если в проектируемом Delphi-приложении требуется использовать новые функции из собственных dll-библиотек, созданных в Компиляторе Matlab, то их заголовки необходимо просто добавить в интерфейсную часть модуля связи и перекомпилировать его.

ПРИЛОЖЕНИЕ Б
(обязательное)
Пример приложения Delphi

Здесь описано иллюстративное Delphi-приложение, аналогичное примеру в [9]. Все математические функции, используемые в этом приложении, находятся в нашей DLL-библиотеке matr_dll.dll.

Назначение приложения – проиллюстрировать механизмы взаимодействия Delphi-приложения и библиотек совместного использования (DLL-библиотек), созданных Компилятором Matlab. Это однооконное приложение, где в окне присутствуют три компоненты StringGrid, соответственно для двух матриц (размерности 3x3) операндов и одной – результата. Кнопка OK служит для осуществления вычисления; кнопка FreeMass – для очистки памяти, выделенной программой под массивы Matlab; кнопка Close – для выхода из программы (становится доступной после очистки массивов). Окно приложения представлено на рисунке А.1.

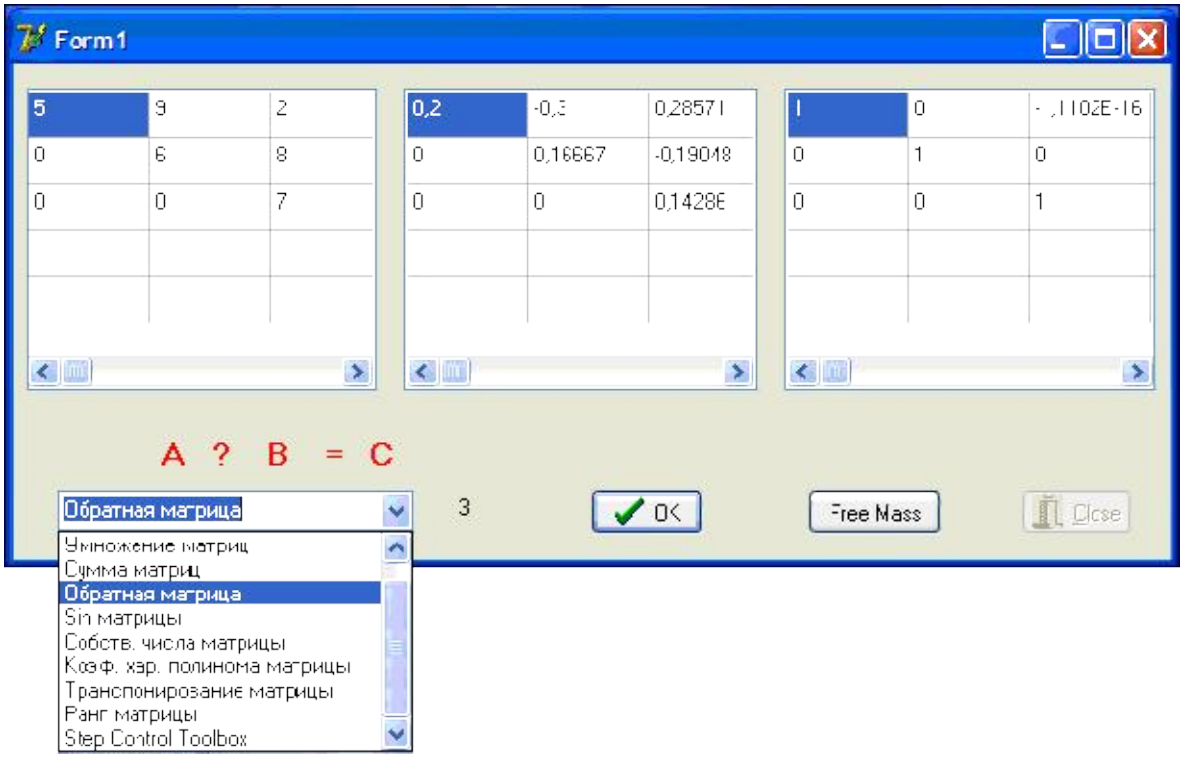


Рисунок А.1 – Окно иллюстративного Delphi-приложения

В выпадающем списке выбирается интересующая математическая операция и затем нажимается кнопка OK. После этого в программе автоматически генерируются необходимые данные-операнды, выполняется соответствующая операция и отображается результат. В качестве исходных данных формируются целочисленные верхняя треугольная и обычная квадратные матрицы. Это сделано для того, чтобы визуально можно было бы сразу оценить правильность выполнения операции.

Текст модуля формы программы приведен ниже. Он во многом повторяет текст аналогичного модуля приложения, описанного в [9]. Однако здесь отражена как специфика работы в среде MCR (`mclInitializeApplication`, `mclTerminateApplication`), так и дополнительные операции по инициализации библиотеки matr_dll (`_matr_dllInitialize`, `_matr_dllTerminate`).

Текст модуля формы иллюстративного приложения

```
 {1} unit Un_MX;
 {2} interface
 {3} uses
 {4}   Windows, Messages, SysUtils, Classes, Graphics,
       Controls, Forms, Dialogs, StdCtrls, Buttons, Grids, XPMan;
 {5} type
 {6}   TForm1 = class(TForm)
 {7}     bb_OK   : TBitBtn;          bb_Clear : TBitBtn;
 {8}     sg_A    : TStringGrid;      sg_B     : TStringGrid;
 {9}     sg_C    : TStringGrid;      bbClose  : TBitBtn;
{10}     cb_oper : TComboBox;
{11}     lb_A    : TLabel;           lb_B     : TLabel;
{12}     lb_C    : TLabel;           lb_oper  : TLabel;
{13}     lb_eq   : TLabel;           lb_cb    : TLabel;
{14}     procedure bb_OKClick(Sender: TObject);
{15}     procedure bb_ClearClick(Sender: TObject);
{16}     procedure cb_operChange(Sender: TObject);
{17}   private
{18}     { Private declarations }
{19}   public
{20}     { Public declarations }
{21}   end;

{22} var
{23}   Form1: TForm1;
{24} implementation
{25} uses Un_DLL_MX;
{26} {$R *.DFM}
{27} const
{28}      col = 3;  row = 3;
{29} type
{30}   vMass = array[1..col*row] of Double;
{31} var
{32}   arrA_ptr, arrB_ptr, arrC_ptr, arrD_ptr  : pmxArray;
{33}   AMass, BMass, CMass           : array[1..row,1..col] of Double;
{34}   dyn_AMass, dyn_BMass          : array of double;
{35}   Flag_MCR_Initialize           : Boolean = False;

{36} procedure TForm1.bb_OKClick(Sender: TObject);
{37} var
{38}   i, j        : Integer;
{39} begin
{40}   for i:=1 to row do
{41}     for j:=1 to col do
{42}       begin
{43}         AMass[i,j]:=random(10)+1;
{44}         if i>j then AMass[i,j]:=0;
{45}         BMass[i,j]:=random(10)+1;
{46}         sg_A.Cells[j-1,i-1]:=FloatToStr(AMass[i,j]);
{47}         sg_B.Cells[j-1,i-1]:=FloatToStr(BMass[i,j]);
{48}       end;
{49}   SetLength(dyn_AMass, row*col);
{50}   SetLength(dyn_BMass, row*col);
{51}   for i:=1 to row do
{52}     for j:=1 to col do
{53}       begin
{54}         dyn_AMass[(i-1)*col+(j-1)]:=AMass[i,j];
```

```
 {55}         dyn_BMass[(i-1)*col+(j-1)]:=BMass[i,j];
 {56}       end;
 {57}   if Flag_MCR_Initialize=False then
 {58}       begin
 {59}         mclInitializeApplication(0,0);
 {60}         _matr_dllInitialize;
 {61}       end;
 {62}   arrA_ptr := mxCreateDoubleMatrix(row,col,mxREAL);
 {63}   Move(AMass, mxGetPr(arrA_ptr)^, row*col*sizeof(double));
 {64}   arrA_ptr:=mxTranspose(arrA_ptr);
 {65}   arrB_ptr := mxCreateDoubleMatrix(row,col,mxREAL);
 {66}   Move(BMass, mxGetPr(arrB_ptr)^, row*col*sizeof(double));
 {67}   arrB_ptr:=mxTranspose(arrB_ptr);
 {68}   arrC_ptr:=Nil;
 {69}   case cb_oper.ItemIndex of
 {70}     1 : _mlfMtimesvak(1,arrC_ptr,arrA_ptr,arrB_ptr);
 {71}     2 : _mlfPlusvak(1,arrC_ptr,arrA_ptr,arrB_ptr);
 {72}     3 : _mlfInvvak(1,arrC_ptr,arrA_ptr);
 {73}     4 : _mlfSinvak(1,arrC_ptr,arrA_ptr);
 {74}     5 : _mlfEigvak(2,arrD_ptr,arrC_ptr,arrA_ptr);
 {75}     6 : _mlfPolyvak(1,arrC_ptr,arrA_ptr);
 {76}     7 : arrC_ptr:=mxTranspose(arrA_ptr);
 {77}     8 : _mlfRankvak(1,arrC_ptr,arrA_ptr);
 {78}     9 : _mlfStepvak(1,arrC_ptr,arrA_ptr);
 {79}     else
 {80}         exit;
 {81}   end;
 {82}   arrC_ptr:=mxTranspose(arrC_ptr);
 {83}   Move(mxGetPr(arrC_ptr)^, CMass, row*col*sizeof(double));
 {84}   for i:=1 to row do for j:=1 to col do sg_C.Cells[j-1,i-1]:='';
 {85}   case cb_oper.ItemIndex of
 {86}     1,2 : begin
 {87}             for i:=1 to row do
 {88}                for j:=1 to col do
 {89}                 sg_C.Cells[j-1,i-1]:=FloatToStrF(CMass[i,j],ffGeneral,5,2);
 {90}           end;
 {91}     3, 4, 5, 6, 7, 8 :
 {92}        begin
 {93}          for i:=1 to row do
 {94}             for j:=1 to col do sg_B.Cells[j-1,i-1]:='';
 {95}          case cb_oper.ItemIndex of
 {96}             3:
 {97}               begin
 {98}                 for i:=1 to row do
 {99}                   for j:=1 to col do
{100}                    sg_B.Cells[j-1,i-1]:=FloatToStrF(CMass[i,j],ffGeneral,5,2);
{101}                 arrC_ptr:=mxTranspose(arrC_ptr);
{102}                 _mlfMtimesvak(1,arrC_ptr,arrA_ptr,arrC_ptr);
{103}                 arrC_ptr:=mxTranspose(arrC_ptr);
{104}                 Move(mxGetPr(arrC_ptr)^, CMass, row*col*sizeof(double));
{105}                 for i:=1 to row do
{106}                   for j:=1 to col do
{107}                    sg_C.Cells[j-1,i-1]:=FloatToStrF(CMass[i,j],ffGeneral,5,2);
{108}               end;
{109}             4, 7 :
{110}               for i:=1 to row do
{111}                 for j:=1 to col do
{112}                 sg_C.Cells[j-1,i-1]:=FloatToStrF(CMass[i,j],ffGeneral,5,2);
{113}             5, 6, 8 :
```

```
{114}               case cb_oper.ItemIndex of
{115}                 5 : for i:=1 to row do
{116}                      sg_C.Cells[i-1,i-1]:=FloatToStrF(CMass[i,i],ffGeneral,5,2);
{117}                 6 : begin
{118}                      for j:=1 to col do
{119}                       sg_C.Cells[j-1,0]:=FloatToStrF(CMass[1,j],ffGeneral,5,2);
{120}                      sg_C.Cells[0,1]:=FloatToStrF(CMass[2,1],ffGeneral,5,2);
{121}                     end;
{122}                 8 : sg_C.Cells[0,0]:=FloatToStrF(CMass[1,1],ffGeneral,5,2);
{123}               end;
{124}           end;
{125}         end
{126}      else
{127}          exit;
{128}   end;
{129}   bbClose.Enabled:=False;
{130} end;

{131} procedure TForm1.bb_ClearClick(Sender: TObject);
{132} begin
{133}   if Flag_MCR_Initialize=True then
{134}       begin
{135}         _matr_dllTerminate;
{136}         mclTerminateApplication;
{137}       end;
{138}   mxDestroyArray(arrA_ptr);
{139}   arrA_ptr:=Nil;
{140}   mxDestroyArray(arrB_ptr);
{141}   arrB_ptr:=Nil;
{142}   if arrC_ptr<>nil then
{143}         begin
{144}           mxDestroyArray(arrC_ptr);
{145}           arrC_ptr:=Nil;
{146}         end;
{147}   bbClose.Enabled:=True;
{148} end;

{149} procedure TForm1.cb_operChange(Sender: TObject);
{150} var
{151}    i : Byte;
{152} begin
{153}    i:=cb_oper.ItemIndex;
{154}    lb_cb.caption:=IntToStr(i);
{155} end;
{156} end.
```

Дадим краткие пояснения текста модуля формы.

В модуле определены три (строки 14÷16) процедуры-обработчика событий **bb_OKClick, bb_ClearClick** и **cb_operChange**. Первые две из них, соответственно, связаны с кнопками OK и Free Mass. Третья процедура обрабатывает событие выбора *элемента* из комбинированного списка – перечня математических операций, которые может выполнять приложение. Кнопки и комбинированный список изображены на рисунке А.1.

Текст последней процедуры (**cb_operChange**) представлен в строках 149÷155. В результате обработки данного события рядом с комбинированным списком выводится число – порядковый номер выбранной для выполнения математической операции (рисунок А.1).

Кнопка Free Mass служит для выполнения подготовительных операций перед окончанием работы приложения. Из текста обработчика OnClick для этой кнопки (процедура **bb_ClearClick**, строки 131÷148) следует, что сначала проводится деинсталляция нашей DLL-библиотеки и среды выполнения MCR (строки 133÷137), затем – освобождение памяти, занятой под массивы данных для функций Matlab (строки 138÷146) и, в конце, становится доступной кнопка Close для закрытия приложения (строка 147).

Вся функциональность приложения сосредоточена в инструкциях обработчика OnClick кнопки OK – в процедуре **bb_OKClick** (строки 36÷130):

– *строки 40÷48*. Формируются две целочисленных случайных квадратных 3x3 матрицы A и B. Первая – верхняя треугольная, вторая – обычная матрицы. Элементы двух матриц – положительные целые числа. Первая матрица или обе сразу служат в качестве операндов для выбранной математической операции. Значения элементов матриц помещаются в ячейки первых двух таблиц окна приложения (компоненты StringGrid);

– *строки 49÷56*. Выделяется память под одномерные динамические массивы и эта память построчно заполняется значениями элементов матриц A и B;

– *строки 57÷61*. Производится инициализация среды выполнения компонентов Matlab и нашей DLL-библиотеки;

– *строки 62÷67*. Исходные данные задачи (матрицы A и B) конвертируются в массивы структуры типа mxArray для использования непосредственно в функциях нашей DLL-библиотеки;

– *строки 69÷81*. Выполняются математические вычисления в соответствии с выбранной операцией из списка;

– *строки 82÷129*. В соответствии с выполненной операцией производится вывод результатов в компоненты StringGrid окна приложения. Блокируется кнопка выхода из приложения.

ПРИЛОЖЕНИЕ В
(обязательное)
Пример решения системы ОДУ

Приводится краткое описание Delphi-приложения, иллюстрирующего возможность решения линейной стационарной системы обыкновенных дифференциальных уравнений (ОДУ) с помощью функции ode45 системы Matlab.

Обобщенный заголовок функции ode45 в Matlab имеет следующий вид [1, 2]:

```
[T,Y]=ode45(func_name,tspan,y0,options,varargin)
```

Параметры функции ode45 означают:

– **T** – вектор-столбец отсчетов независимой переменной;
– **Y** – числовая матрица значений решений системы ОДУ. Каждый столбец матрицы Y соответствует вычисленным значениям одной из компонент вектор-функции решений. Каждая строка матрицы Y содержит значения всех компонент вектор-функции решений при одном значении аргумента;
– **func_name** – указатель на функцию вычисления правых частей системы ОДУ. Это может быть и имя m-файла с одноименной функцией;
– **tspan** – вектор-строка [t_0 t_k] из двух чисел – начального и конечного значения аргумента (независимой переменной). Для получения значений решений в заданных точках t_i данный параметр может представлять собой вектор-строку вида [t_0, t_1, t_2, …, t_k] или, даже, tspan=t_0:step:t_k;
– **y0** – вектор начальных значений компонент искомой вектор-функции;
– **options** – строка параметров, определяющих точностн***ы***е характеристики решения. Это необязательный параметр, он указывается, если необходимо изменить параметры, установленные по умолчанию;
– **varargin** – дополнительные параметры, через которые передаются данные в функцию вычисления правых частей **func_name**. Этот параметр не является обязательным.

Синтаксис функции вычисления правых частей системы ОДУ имеет вид:

```
function dydt = right(t,y,varargin)
```

Через параметр t в правую часть системы ОДУ передается текущее значение независимой переменной. Параметр y служит для передачи в функцию right текущих значений компонент вектор-функции решений. Функция возвращает в выходном параметре dydx значения правой части системы ОДУ для текущего значения аргумента. Входные параметры varargin используются для передачи в функцию right необходимых дополнительных данных. Количество параметров в varargin и их порядок должен полностью повторять аналогичный набор параметров в вызывающей функции ode45.

Наше иллюстративное Delphi-приложение предназначено для решения линейной стационарной системы ОДУ, записанной в нормальной форме

$$\frac{dx(t)}{dt} = A \cdot x(t) + B \cdot u(t), \quad t \in [t_0, t_k], \quad x(t_0) = x_0. \qquad \text{(B.1)}$$

В выражении (B.1) введены следующие обозначения:

– t – независимая переменная, определенная на интервале [t_0, t_k];

– $x(t)$ – искомая вектор-функция решений системы ОДУ;
– $u(t)$ – внешняя известная вектор-функция;
– x_0 – вектор начальных значений компонент искомой вектор-функции;
– A, B – числовые матрицы коэффициентов системы ОДУ.

На рисунке В.1 изображено окно иллюстративного Delphi-приложения для решения системы ОДУ. Пользователь может задавать значения начального t_0 и конечного t_k времени интегрирования и вектор начальных значений x_0. Порядок системы, равный трём, задан программно и меняться не может. Матрицы A и B также жёстко заданы в программе. Поле "Графики компонент состояния" определено для вывода графиков решений системы ОДУ. Кнопка "Start" предназначена для выполнения операторов программы, производящих необходимые вычисления и изображение полученных трёх графиков решений системы ОДУ.

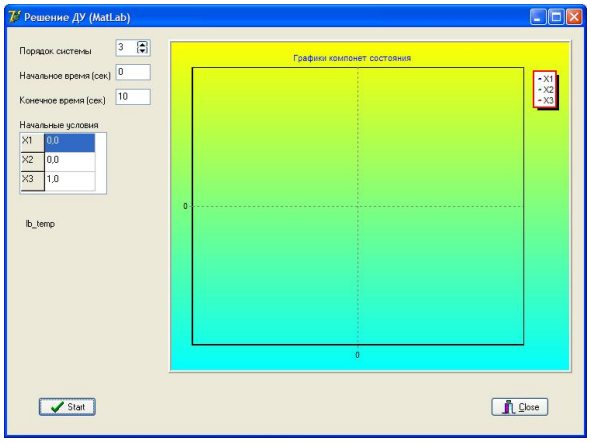


Рисунок В.1 – Окно Delphi-приложения для решения системы ОДУ

Решение системы ОДУ осуществляется с помощью m-функции Odevak, которая оформлена как отдельная DLL-библиотека ode_dll. Текст m-файла Odevak.m с одноимённой m-функцией приведен ниже

```
function [T,Y] = Odevak(n,A,B,x0,time)
     [T,Y]=ode45(@right, time, x0,[],n,A,B);
     plot(T,Y); grid on;

function dx = right(t,x,n,A,B)
     dx = zeros(n,1);
     dx = A*x; %+B*1;
```

В файле представлены две m-функции, собственно Odevak и функция вычисления правых частей right. Функция Odevak имеет пять входных параметров: n – размерность системы ОДУ; A, B – матрицы коэффициентов системы; x0 – вектор начальных условий; time – интервал интегрирования системы ОДУ. Возвращает наша функция Odevak, как и функция ode45, два объекта – T и Y – массив отсчетов независимой переменной и массив решений системы ОДУ, соответственно. Тело функции Odevak содержит собственно одну инструкцию – вызов функции ode45. Через три последних параметра функции ode45 в функцию вычисления правой части right передаются данные о размерности (n) и коэффициентах (A, B) системы ОДУ. Обращаем внимание на то, что количество и порядок этих параметров должен полностью совпадать в обеих функциях – ode45 и right.

Инструкция plot(T,Y) в тексте функции Odevak служит исключительно для проверочных целей – сравнения графиков решений, построенных системой Matlab и нашим Delphi-приложением.

Содержание функции вычисления правых частей right показывает, что мы в данном случае можем решать исключительно однородную систему ОДУ (%+B*1).

Ниже приведен текст модуля сопряжения Delphi-приложения и среды выполнения компонентов Matlab. В начальной части (строки от 1 до 26) он повторяет текст аналогичного модуля из Приложения А. В разделе "**Моя библиотека**" (строки 27, 28 и 29) представлены только три записи – две функции (**_ode_dllInitialize** и **_ode_dllTerminate**) для инициализации DLL-библиотеки ode_dll и процедура **_mlfOdevak**, реализующая нашу m-функцию Odevak. Эти три функции и составляют содержание нашей DLL-библиотеки ode_dll.

```
 {1} unit Un_DLL_MX;
 {2} interface
 {3} Type
 {4}   pDouble = ^Double;
 {5}   mxComplexty = (mxREAL, mxCOMPLEX);
 {6}   pmxArray = ^mxArray;
 {7}   mxArray  = record
 {8}               { Просто обозначить этот тип!!!}
 {9}              end;
{10}
{11} {*************** Из библиотеки Libmx.dll ***************}
{12}   function  mxCreateDoubleMatrix(m,n:Integer;
                  mxData:mxComplexty):pmxArray;   cdecl; external 'libmx.dll';
{13}   procedure mxSetName(arr_ptr:pmxArray; const name:String);
                                                  cdecl; external 'libmx.dll';
{14}   function  mxGetPr(arr_ptr:pmxArray):pDouble;cdecl; external 'libmx.dll';
{15}   function  mxGetPi(arr_ptr:pmxArray):pDouble;cdecl; external 'libmx.dll';
{16}   function  mxGetM(arr_ptr:pmxArray):Integer; cdecl; external 'libmx.dll';
{17}   function  mxGetN(arr_ptr:pmxArray):Integer; cdecl; external 'libmx.dll';
{18}   procedure mxDestroyArray(arr_ptr:pmxArray); cdecl; external 'libmx.dll';
{19}   procedure mxFree(var ram:THandle);          cdecl; external 'libmx.dll';
{20}   function  mxTranspose(x:pmxArray):pmxArray; cdecl; external 'libmx.dll';
{21}
{22} {*************** Из библиотеки mclmcr.dll ***************}
{23}   function mclInitializeApplication(A:THandle;B:Integer):Boolean;
                                                 cdecl; external 'mclmcr.dll';
{24}   function mclTerminateApplication:Boolean;  cdecl; external 'mclmcr.dll';
{25}
{26} {*************** Моя библиотека ***************}
{27}   function  _ode_dllInitialize:Boolean;     cdecl; external 'ode_dll.dll';
{28}   function  _ode_dllTerminate:Boolean;      cdecl; external 'ode_dll.dll';
{29}   procedure _mlfOdevak(i:Integer;var pout1,pout2:pmxArray;
                      n,A,B,x0,time:pmxArray); cdecl; external 'ode_dll.dll';
{30}
{31} implementation
{32} end.
```

Прокомментируем кратко текст (приведен ниже) модуля Main_ODE главной формы нашего Delphi-приложения. Строки 1÷53 содержат вполне очевидные инструкции. Вся функциональность программы сосредоточена в обработчике OnClick кнопки Start – процедура TMain_Form.bb_StartClick (строки 54÷114).

```
 {1} unit Main_ODE;
 {2} interface
 {3} uses
 {4}   Windows, Messages, SysUtils, Classes, Graphics, Controls,
       Forms, Dialogs, StdCtrls, ExtCtrls, XPMan, Grids, Spin, Buttons,
       TeEngine, Series, TeeProcs, Chart, Un_DLL_MX;
 {5} type
 {6}   TMain_Form = class(TForm)
 {7}     lb_Dim_Sysy: TLabel;     lb_Time_End: TLabel;
 {8}     lb_X0: TLabel;           lb_Time_Begin: TLabel;
 {9}     bb_Start: TBitBtn;       bb_Close: TBitBtn;
{10}     ed_Time_End: TEdit;      ed_Time_Begin: TEdit;
{11}     se_Dim_Sys: TSpinEdit;   sg_X0: TStringGrid;
{12}     lb_temp: TLabel;
{13}     Chart_State: TChart;
{14}     Series_X1: TFastLineSeries;    Series_X2: TFastLineSeries;
{15}     Series_X3: TFastLineSeries;    XPManifest1: TXPManifest;
{16}     procedure FormCreate(Sender: TObject);
{17}     procedure bb_StartClick(Sender: TObject);
{18}     procedure bb_CloseClick(Sender: TObject);
{19}   private
{20}     { Private declarations }
{21}   public
{22}     { Public declarations }
{23}   end;
{24} const
{25}   N_step : Integer = 200;
{26} var
{27}   Dim : Byte = 3;
{28}   Main_Form: TMain_Form;
{29}   T0, Tk, step : Double;
{30}   A, B, mass_Y, X0, mass_T  : array of Double;
{31}   Y_print                   : array of array of Double;
{32}   Flag_MCR_Initialize       : Boolean = False;
{33} implementation
{34} {$R *.dfm}

{35} procedure TMain_Form.FormCreate(Sender: TObject);
{36}   var
{37}     i  : Integer;
{38}     ch : Char;
{39} begin
{40}   T0:=StrToFloat(ed_Time_Begin.Text);
{41}   Tk:=StrToFloat(ed_Time_End.Text);
{42}   step:=(Tk-T0)/N_step;
{43}   Dim:=se_Dim_Sys.Value;
{44}   sg_X0.RowCount:=Dim;
{45}   sg_X0.ColWidths[0]:=30;
{46}   ch:=DecimalSeparator;
{47}   for i:=1 to Dim do
{48}     begin
{49}       sg_X0.Cells[0,i-1]:='X'+IntToStr(i);
{50}       sg_X0.Cells[1,i-1]:='0'+ch+'0';
{51}     end;
{52}   sg_X0.Cells[1,Dim-1]:='1'+ch+'0';
{53} end;
```

```
{54} procedure TMain_Form.bb_StartClick(Sender: TObject);
{55} var
{56}   arrA_ptr, arrB_ptr, arrX0_ptr,
{57}   arrT_ptr, arrY_ptr,
{58}   arrTemp_ptr, arrScalar_ptr         : pmxArray;
{59}   temp_Double                        : Double;
{60}   i, j, num_Point                    : Integer;
{61} begin
{62}   Dim:=se_Dim_Sys.Value;
{63}   T0:=StrToFloat(ed_Time_Begin.Text);
{64}   Tk:=StrToFloat(ed_Time_End.Text);
{65}   step:=(Tk-T0)/N_step;
{66}   SetLength(A,Dim*Dim);
{67}   SetLength(B,Dim);
{68}   SetLength(X0,Dim);
{69}   SetLength(mass_T,N_step+1);
{70}     {0}       A[1]:=1;        {0}
{71}     {0}           {0}      A[5]:=1;
{72}   A[6]:=-1;   A[7]:=-2;    A[8]:=-3;
{73}   B[Dim-1]:=1;
{74}   for i:=0 to Dim-1 do X0[i]:=StrToFloat(sg_X0.Cells[1,i]);
{75}   for i:=0 to N_step do mass_T[i]:=T0+i*step;
{76}   if Flag_MCR_Initialize=False then
{77}       begin
{78}         mclInitializeApplication(0,0);
{79}         _ode_dllInitialize;
{80}       end;
{81}   arrA_ptr := mxCreateDoubleMatrix(Dim, Dim, mxREAL);
{82}   Move(A[0], mxGetPr(arrA_ptr)^, Dim*Dim*sizeof(Double));
{83}   arrA_ptr := mxTranspose(arrA_ptr);
{84}   arrB_ptr := mxCreateDoubleMatrix(Dim, 1, mxREAL);
{85}   Move(B[0], mxGetPr(arrB_ptr)^, Dim*sizeof(Double));
{86}   arrTemp_ptr := mxCreateDoubleMatrix(1,N_step+1, mxREAL);
{87}   Move(mass_T[0], mxGetPr(arrTemp_ptr)^, (N_step+1)*sizeof(Double));
{88}   arrX0_ptr := mxCreateDoubleMatrix(Dim, 1, mxREAL);
{89}   Move(X0[0], mxGetPr(arrX0_ptr)^, Dim*sizeof(Double));
{90}   arrScalar_ptr := mxCreateDoubleMatrix(1, 1, mxREAL);
{91}   temp_Double:=Dim;
{92}   Move(temp_Double,mxGetPr(arrScalar_ptr)^, 1*sizeof(Double));
{93}   arrT_ptr:=nil; arrY_ptr:=nil;
{94}   _mlfOdevak(2,arrT_ptr,arrY_ptr,arrScalar_ptr,arrA_ptr,arrB_ptr,
                                                  arrX0_ptr,arrTemp_ptr);
{95}   num_Point:=mxGetM(arrT_ptr);
{96}   SetLength(mass_T,num_Point);
{97}   SetLength(mass_Y,num_Point*Dim);
{98}   Move(mxGetPr(arrT_ptr)^,mass_T[0], num_Point*sizeof(Double));
{99}   Move(mxGetPr(arrY_ptr)^,mass_Y[0], Dim*num_Point*sizeof(Double));
{100}  SetLength(Y_print,num_Point,Dim);
{101}  for i:=0 to Dim-1 do
{102}  for j:=0 to num_Point-1 do
{103}    Y_print[j,i]:=mass_Y[j+num_point*i];
{104}  lb_temp.Caption:='Получил данные из Matlab';
{105}  Series_X1.Clear;
{106}  Series_X2.Clear;
{107}  Series_X3.Clear;
{108}  for i:=0 to num_Point-1 do
{109}    begin
{110}      Series_X1.AddXY(mass_T[i],Y_print[i,0],'',clTeeColor);
{111}      Series_X2.AddXY(mass_T[i],Y_print[i,1],'',clTeeColor);
```

```
{112}       Series_X3.AddXY(mass_T[i],Y_print[i,2],'',clTeeColor);
{113}     end;
{114} end;

{115} procedure TMain_Form.bb_CloseClick(Sender: TObject);
{116} begin
{117}   if Flag_MCR_Initialize=True then
{118}       begin
{119}         _ode_dllTerminate;
{120}         mclTerminateApplication;
{121}       end;
{122}   Close;
{123} end;
{124} end.
```

В строках от 62 до 75 готовятся исходные данные задачи интегрирования системы ОДУ: размерность системы ОДУ (Dim); параметры интервала интегрирования (T0, Tk, step), весь интервал делится на 200 частей (константа N_step); выделение памяти под параметры задачи – матрицы коэффициентов (A, B) и вектор начального состояния (X0); заполнение выделенной памяти числовыми значениями (строки 70÷74); формирование массива отсчетов независимой переменной (mass_T).

Далее (строки 76÷80) производится инициализация Среды выполнения MCR и нашей библиотеки ode_dll. В строках 81÷93 формируются массивы типа pmxArray и наполнение их исходными данными задачи.

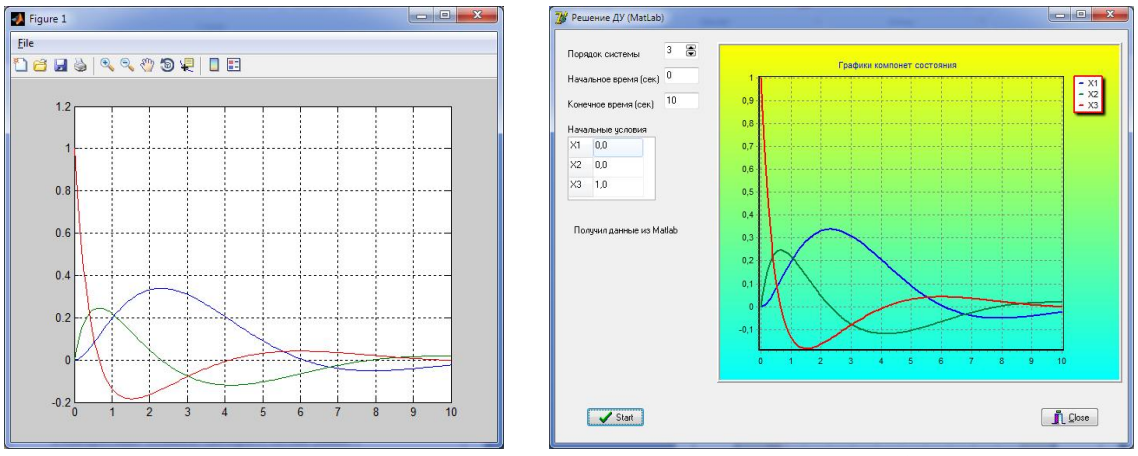
Главная инструкция процедуры-обработчика находится в строке 94. Здесь происходит вызов из нашей DLL-библиотеки процедуры _mlfOdevak с необходимым набором параметров. После отработки процедуры _mlfOdevak мы имеем решение нашей системы ОДУ в массивах arrT_ptr и arrY_ptr.

В строках 95÷103 проводится распаковка pmxArray-массивов в обычные массивы – одномерный массив отсчетов независимой переменной mass_T и двумерный массив решений системы ОДУ Y_print.

В последней части процедуры-обработчика (строки 105÷113) обычным образом в компоненте TChart строятся графики решений системы ОДУ.

В процедуре-обработчике кнопки Close перед выходом из программы проводится выгрузка нашей DLL-библиотеки и деинициализация Среды MCR.

Результаты работы Delphi-приложения изображаются (рисунок В.2) на двух графиках – в отдельном окне системы Matlab (а) и в окне окна приложения (б).



а) б)

Рисунок В.2 – Окна результатов выполнения приложения

Заказ №_____ от "____" _____ 201__г. Тираж _____ экз.

Изд-во СевНТУ