

Взаимодействие Matlab с ANSI C, Visual C++, Visual BASIC и Java



Совместно с доктором Джифенгом Ксу

3.1. Введение

Matlab — чрезвычайно мощный инструмент, помогающий разработчикам создавать приложения с графическим интерфейсом, предназначенные для выполнения математических вычислений. Matlab стал базовым инструментом для создания тысяч приложений, используемых в промышленности, учебных заведениях и правительственных учреждениях. Данный пакет широко используется студентами для решения вычислительных задач, анализа данных и визуализации результатов работы. Область применения Matlab могла бы быть еще шире, если бы данный инструмент был совместим с другими программными средами, например Visual C++ (VC++). Программы на C++ могли бы использоваться совместно с Matlab для сбора данных, управления процессами и выполнения других подобных действий.

Для решения любой задачи управления реальными объектами необходим драйвер, обеспечивающий взаимодействие аппаратных и программных средств. Однако найти драйвер, подходящий для сбора информации и управления в реальном времени, чрезвычайно трудно. Очень часто пользователям бывает необходимо получать данные от систем реального времени, используя программы, написанные на C или VC++, и преобразовывать их в формат, пригодный для обработки в среде Matlab. В таких случаях приходится отказываться от режима реального времени, что создает очень большие неудобства.

В управляющих системах часто возникает задача обработки матриц, которые формируются на основе данных, полученных в результате измерений. Организовать обработку матриц в среде C или VC++ достаточно трудно. Данные можно преобразовать в формат, пригодный для обработки в Matlab, однако при отсутствии интерфейса между программами не удастся осуществлять обработку в реальном времени.

Используя средства взаимодействия VC++ и Matlab, можно реализовать сложные алгоритмы обработки данных в системах реального времени. Посредством программ, написанных на C или VC++, легко реализовать низкоуровневое взаимодействие с аппаратными средствами. Эти программы могут использоваться для передачи команд устройствам и обмена данными с ними. При получении данных программы, написанные на C или VC++, могут обращаться к инструментальным средствам Matlab, которые, в свою очередь, выполняют необходимую обработку данных. Результаты обработки можно снова передать программе на C или VC++, которая использует их для управления объектом. Еще одним преимуществом системы, включающей средства VC++ и Matlab, является возможность представления данных в разных форматах. В управляющих и измерительных системах часто бывает необходимо составлять различные графики. Matlab — удобный инструмент для визуализации данных. Он обеспечивает создание графиков 1D, 2D и 3D. Реализовать техмерную графику средствами C или VC++ чрезвычайно трудно.

Интерфейс между VC++ и Matlab позволяет существенно повысить уровень систем сбора и анализа данных. При этом становится возможным решать сложные задачи обработки в реальном времени.

3.2. Библиотеки Matlab для математических вычислений и представления графических данных

При создании приложений средствами объектно-ориентированного программирования приходится уделять большое внимание получению данных и генерации управляющих сигналов в реальном времени. Обеспечить отклик в реальном времени тем труднее, чем быстрее изменяется состояние объектов. При тестировании важно не только получение данных и генерация отклика, часто приходится строить в реальном времени различные графики. Matlab предоставляет мощные средства для обработки графической информации. Обращаясь к функциям Matlab, можно относительно просто выполнять сложные математические расчеты и строить графики.

Разработчик, пытающийся решить задачу представления графики с VC++, вынужден самостоятельно реализовывать сложные алгоритмы. Преодолеть трудности можно, используя интерфейс между Matlab и VC++, т.е. применяя для создания независимого приложения средства Matlab Compiler, C/C++ Math Library и C/C++ Graphic Library.

3.2.1. Среда разработки Matlab

Для создания независимого приложения можно использовать следующие средства.

- Matlab Compiler 2.1 (или более новую версию).
- Matlab C/C++ Math Library.
- Matlab C/C++ Graphic Library.

Matlab Compiler является основным инструментом разработки независимых приложений. Он выступает в роли инструмента, позволяющего преобразовывать файл М-кода в исходный код С или С++. Matlab C/C++ Math Library и Matlab C/C++ Graphic Library представляют собой динамически связываемые библиотеки. Они предоставляют разработчику встроенные математические и графические функции Matlab. Если в приложении не выполняются математические расчеты и не используются графические функции, необходимости в данных библиотеках нет.

После установки Matlab Compiler, Matlab C/C++ Math Library и Matlab C/C++ Graphic Library необходимо настроить среду разработки, используя для этой цели команду `mbuild`. Задать данную команду можно в ответ на приглашение Matlab; вы также можете сделать это в окне DOS. При вызове команды `mbuild` выполняются следующие действия.

- Программа `mbuild` определяет местонахождение компилятора ANSI С или С++ и копирует данные о его расположении в каталог, содержащий профили пользователя. Впоследствии, когда Matlab Compiler обращается к `mbuild` для вызова компилятора С и С++, записанные данные используются для его поиска.
- Программа `mbuild` определяет расположение библиотек, которые могут использоваться для создания приложения. В качестве примера таких библиотек можно привести Matlab C/C++ Math Library, Matlab C/C++ Graphic Library, а также библиотеку ANSI С или С++.

Основные этапы разработки независимого приложения показаны на рис. 3.1.

1. Matlab Compiler вызывается для преобразования М-кода в исходный код С или С++.
2. Matlab Compiler вызывает компилятор С или С++, используя данные о его расположении.
3. Компилятор С/С++ преобразует исходный текст программ в объектный код.
4. По окончании преобразования кода компилятор С/С++ вызывает редактор связей.
5. Редактор связей выполняет компоновку, подключая к объектному коду необходимые библиотеки, например Matlab C/C++ Math Library, Matlab C/C++ Graphic Library и библиотеку С++, и генерирует исполняемый файл приложения.

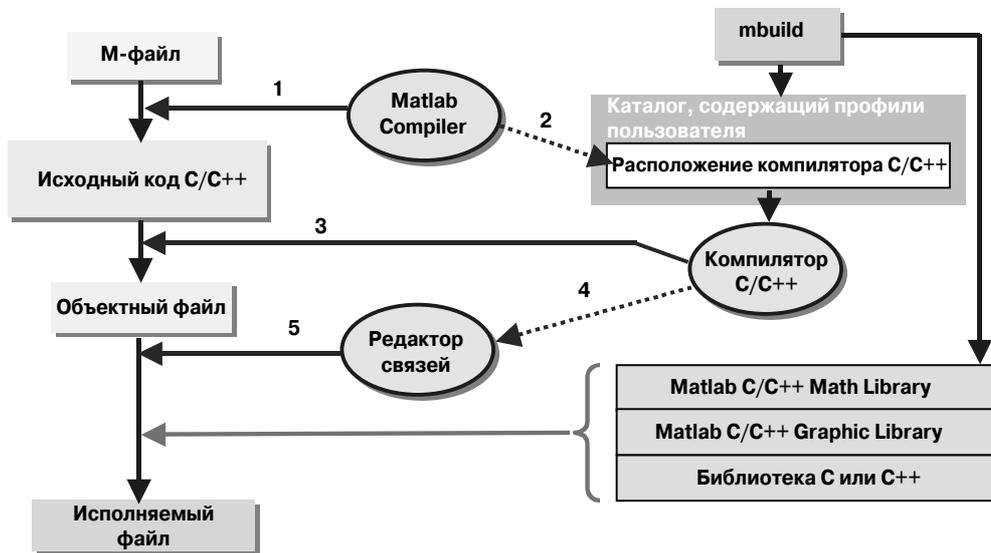


Рис. 3.1. Основные этапы создания независимого приложения

Matlab Compiler генерирует не только исходные тексты C/C++, но и файлы оболочки, содержащие необходимые компоненты независимого приложения, например функцию `main()`.

Независимое приложение может быть запущено либо в окне DOS, либо в командном окне Matlab. Для запуска приложения в командном окне Matlab необходимо указать перед именем исполняемого файла символ `!`. Запуская приложение в окне DOS, необходимо убедиться, что все библиотеки, используемые при выполнении программы, находятся в одном из тех каталогов, в которых система автоматически осуществляет поиск файлов. В данной главе мы не будем непосредственно запускать приложение. Вместо этого мы будем обращаться к нему из программы на C/C++. Такое обращение имеет некоторые преимущества по сравнению с непосредственным вызовом. Так, например, вы можете создать в программе на C/C++ необходимые файлы данных (это может быть информация, полученная в результате взаимодействия с аппаратными средствами) и лишь после этого вызвать приложение, предназначенное для обработки этих данных и представления их в виде графика.

3.2.2. Настройка библиотек Matlab C/C++

Для настройки Matlab C/C++ Math Library или Matlab C/C++ Graphic Library задайте в командном окне Matlab или в окне DOS команду `mbuild`. На экране появится следующее меню*.

* Источник: MathWorks, Inc. November, 2000. Configuring the MATLAB C/C++ Graphics Library. MATLAB C/C++ Graphics Library Documentation. Печатается с разрешения. — Прим. авт.

```
Please choose your compiler for building standard-alone MATLAB
applications:
Would you like mbuild to locate installed compilers [y]/n? n
Select a compiler:
[1] Borland C++ Builder version 5.0
[2] Borland C++ Builder version 4.0
[3] Borland C++ Builder version 3.0
[4] Borland C/C++ version 5.02
[5] Borland C/C++ version 5.0
[6] Borland C/C++ (free command line tools) version 5.5
[7] Lcc C version 2.4
[8] Microsoft Visual C/C++ version 6.0
[9] Microsoft Visual C/C++ version 5.0
[0] None
Compiler: 8
Your machine has a Microsoft Visual C/C++ compiler located at
C:\Program Files\Microsoft Visual Studio.
Do you want to use this compiler? [y]/n y
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Program Files\Microsoft Visual Studio
Are these correct? ([y]/n): y
```

Если в ответ на приглашение вы введете значение *y*, Matlab отобразит в рабочей области приведенный ниже текст. Этот текст оканчивается напоминанием о том, что если вы хотите использовать дополнительный модуль Matlab в среде Microsoft Visual C++, вам необходимо выполнить две команды*.

```
The default options file:
"C:\WINDOWS\Application Data\MathWorks\MATLAB\R12\compopts.bat" is
being updated from
C:\MATLAB6P1\BIN\WIN32\mbuildopts\msvc60compp.bat ...
Installing the MATLAB Visual Studio add-in ...
Updated C:\Program Files\Microsoft Visual
Studio\common\msdev98\template\MATLABWizard.awx from
C:\MATLAB6P1\BIN\WIN32\MATLABWizard.awx
Updated C:\Program Files\Microsoft Visual
Studio\common\msdev98\template\MATLABWizard.hlp from
C:\MATLAB6P1\BIN\WIN32\MATLABWizard.hlp
Updated C:\Program Files\Microsoft Visual
Studio\common\msdev98\addins\MATLABAddin.dll from
C:\MATLAB6P1\BIN\WIN32\MATLABAddin.dll Merged
C:\MATLAB6P1\BIN\WIN32\usertype.dat with C:\Program
Files\Microsoft Visual Studio\common\msdev98\bin\usertype.dat
Note: If you want to use the MATLAB Visual Studio add-in with the
MATLAB C/C++ Compiler, you must start MATLAB and run the following
commands:
cd(prefdir);
mccsavepath;
(You only have to do this configuration step once.)
```

* Источник: Mathworks, Inc. November, 2000. Using an Integrated Development Environment. MATLAB Compiler Documentation, pp. 1-11. Печатается с разрешения. — Прим. авт.

По окончании настройки `mbuild` создаст в каталоге `C:\WINDOWS\Profiles` файл опций. Заметьте, что информацию о расположении компилятора задает пользователь. На этом заканчивается настройка библиотек и установка среды разработки. Теперь все готово для создания приложений в среде Matlab.

Разработчики, использующие версию Matlab R12, могут применять различные методы вызова функций Matlab из среды VC++. В данной книге мы рассмотрим возможности, предоставляемые каждым из методов. Ниже описаны четыре из них.

Метод I

Вызов из программы на VC++ независимого приложения (файла `.exe`), скомпилированного с помощью Matlab Compiler 2.1.

Метод II

Использование стратегии Matlab Add-In для встраивания функций Matlab в программную среду VC++. В среде VC++ функции Matlab могут быть модифицированы.

Метод III

Использование Matlab Engine. Данный метод эквивалентен методу ActiveX.

Метод IV

Создание функций Matlab в виде разделяемой библиотеки DLL, к которой может осуществляться обращение из программной среды VC++.

При использовании любого из этих методов необходимы Matlab Compiler, Matlab C/C++ Math Library и C/C++ Graphic Library.

Библиотека Math Library, предоставляемая Matlab, является набором инструментов для математических вычислений, включающих функции общего назначения, средства для решения задач линейной и нелинейной алгебры, выполнения действий над матрицами, массивами и строками, а также инструменты дискретной математики и символьных операций. Все функции написаны на языке C. Данная библиотека используется в качестве интерфейса к соответствующим средствам Matlab.

Библиотека Graphics Library применяется для реализации графических средств, например построения графиков 1D, 2D и 3D. Указанные библиотеки необходимы для организации эффективного взаимодействия программ C/C++ с Matlab.

3.2.3. Основной элемент, обеспечивающий взаимодействие VC и Matlab

Основным компонентом, обеспечивающим интерфейс между VC и Matlab, является Matlab Compiler — инструмент для преобразования программ и функций, написанных в формате Matlab (M-файлы или M-функции), в формат C или VC++. В настоящее время доступна версия 2.1 данного компилятора.

В последующих разделах мы подробно обсудим четыре перечисленных выше метода организации взаимодействия и проиллюстрируем их примерами.

3.3. Вызов независимого приложения

Метод I. Компилятор Matlab Compiler 2.1 преобразует М-функции в независимое приложение (файл `.exe`), после чего это приложение вызывается из среды VC++.

Подобному преобразованию можно подвергнуть любую М-функцию или набор функций, после чего файл приложения можно выполнить, не устанавливая библиотеки Matlab. Более того, Matlab Compiler создает исходный код C/C++ и соответствующие файлы заголовков. Это означает, что почти все М-функции, предназначенные для выполнения в среде Matlab, могут быть преобразованы в функции C/C++, пригодные для использования в среде VC++ (исключением являются функции Matlab Graphic; их нельзя непосредственно вызвать из среды VC++).

Строго говоря, из программы на VC++ можно вызвать любые функции, предназначенные для математических вычислений, например средства решения нелинейных уравнений или выполнения операций над матрицами. Графические операции представляют собой отдельный набор возможностей, предоставляемых Matlab. С их помощью можно создавать сложные графики 1D, 2D и 3D.

Стандартный способ использования графических функций Matlab предполагает компиляцию М-функций, созданных в среде Matlab, и преобразование их в независимое приложение с использованием Matlab Compiler 2.1. Полученный исполняемый файл можно вызвать из программы на VC++ для создания графиков в реальном времени. Ниже приведен пример, демонстрирующий этот способ.

В листинге 3.1 показана М-функция `dataplot()`, созданная в среде Matlab. Откройте в рабочей области Matlab новый М-файл с именем `dataplot.m` и введите в него код, показанный в листинге 3.2. Перед тем как обращаться к этой функции, необходимо сохранить в текстовом файле данные, предназначенные для вывода. Этот файл будет размещаться в каталоге `C:\data`. Присвоим файлу имя `mdata.txt`. Чтобы при вызове М-функции можно было извлечь информацию, Matlab открывает файл, помещает прочитанные данные в массив `[s]` и вызывает функцию построения графика.

Листинг 3.1. М-функция `dataplot()`

```
% Описание: функция для отображения данных
%           в реальном масштабе времени
% Входные данные: отсутствуют
% Выходные данные: отсутствуют
% Дата: 5/25/2001
% Имя функции: dataplot()

% Определение функции
function dataplot()
```

```

m = 1;
n = 10;
fid = fopen( 'C:\data\mdata.txt ', ' r ');
[s] = fscanf(fid, ' %f ', [m, n])

plot( s);
grid
xlabel(' Time ');
title('Testing Program for Plotting a Response ');
ylabel(' Response ');

```

Для открытия файла и получения его идентификатора используется функция `fopen()`. Функция `fscanf()` извлекает данные из потока и помещает их в массив `[s]`. Остальные функции интуитивно понятны и не требуют подробных объяснений. Следует лишь заметить, что функция `plot()` выполняет построение графика.

Для того чтобы преобразовать М-функцию в исполняемый файл, к которому можно обратиться из программы на VC++, выполните следующие действия.

- В командном окне Matlab вызовите Matlab Compiler (`mcc`) и преобразуйте с его помощью М-функцию в независимое приложение (опция `-m` указывает на то, что на основе М-файла должен быть сгенерирован код C).

```
mcc -B sgl dataplot.m
```

Опция `-B` указывает на наличие дополнительных файлов; опция `sgl` сообщает, что при компиляции должна использоваться Matlab C/C++ Graphics Library, а генерируемый файл должен содержать код независимого графического приложения. Ниже приведены файлы, созданные в результате выполнения данной команды.

```

dataplot.c    dataplot.h    dataplot.exe
title.c      title.h
xlabel.c     xlabel.h
ylabel.c     ylabel.h

```

В данном случае нам нужен только файл `dataplot.exe`, представляющий собой независимое приложение.

- Запустите VC++ 6.0 и выберите пункт `New` меню `File`. Затем на вкладке `Project` выберите `Win32 Console Application`, создав таким образом новый проект консольного приложения `Win32`. В поле редактирования `Project name`: введите имя проекта `MatlabPlot`, а в поле `Location`: задайте каталог, в котором вы хотите разместить проект, например `D:\Vc`. Щелкните на кнопке `OK`, чтобы закрыть диалоговое окно. В следующем диалоговом окне оставьте установки по умолчанию неизменными и щелкните на кнопке `Finish`. В появившемся после этого окне щелкните на кнопке `OK`. В результате ваших действий будет создана рабочая область нового проекта.

- Щелкните на вкладке FileView и выберите папку Source Files. Щелкните на пункте меню File и снова выберите пункт New, чтобы открыть новое диалоговое окно. На этот раз по умолчанию будет выбрана вкладка Files. Выберите в ней пункт C++ Source File, приступив таким образом к созданию нового исходного файла C++. В поле редактирования File name: введите имя MatlabPlot и щелкните на кнопке ОК. Этим вы создадите новый исходный файл с именем MatlabPlot.
- Включите в созданный файл код, показанный в листинге 3.2. Как видно в листинге, в функции main() содержится вызов независимого приложения dataplot.exe, созданного ранее с использованием Matlab Compiler 2.1.

Листинг 3.2. Содержимое файла MatlabPlot.cpp

```

/*-----*/
Файл:      MatlabPlot.cpp
Описание:  построение графика с помощью C-функции,
           созданной на базе Matlab.
Дата:      5/29/01
/*-----*/

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

PROCESS_INFORMATION proInfo ;
STARTUPINFO  startInfo ;
double data_array[] = {1.2, 2.3, 3.4, 8.1, 7.3, 3.3, 2.1, 10.05, 5.88,
5.77 };          // Имитация реальных данных
FILE*  fp;

void main()
{
    int  i;
    DWORD  ErrCode;
    char  currDri[] = " C:\\Vc ";
    char  mod[] = "C:\\MATLAB6p1\\work\\dataplot.exe "; // Путь
                                                    // к файлу
    fp = fopen(" C:\\data\\mdata.txt", "w" ); // Получение
                                                    // дескриптора файла
    for (i=0; i<=10; i++)
        fprintf(fp, " %f ", data_array[i]); // Запись в файл
    fclose( fp);
        // Подготовка к созданию процесса
    startInfo.cb = sizeof(STARTUPINFO);
    startInfo.cbReserved2 = 0;
    startInfo.dwFillAttribute = BACKGROUND_GREEN;
    startInfo.lpTitle = NULL;
    startInfo.dwX = CW_USEDEFAULT;
    startInfo.dwY = CW_USEDEFAULT;
    startInfo.dwXSize = CW_USEDEFAULT;
    startInfo.dwYSize = CW_USEDEFAULT;
    startInfo.dwXCountChars = 0;

```

```

startInfo.dwYCountChars = 0;
startInfo.lpReserved 2 = NULL;
startInfo.lpReserved = NULL;
startInfo.lpDesktop = NULL;
startInfo.dwFlags = STARTF_USEFILLATTRIBUTE;
// Создание процесса
ErrCode =
    CreateProcess( mod,NULL,NULL,NULL,FALSE,0,NULL,
                  currDri ,&startInfo ,&proInfo );
if (!ErrCode)
{
    ErrCode = GetLastError();
    printf("CreateProcess is failed & ErrCode = %x\n  ",
          ErrCode);
}
return;
}

```

В листинге 3.2 присутствует выражение `currDri[] = "D:\\Vc"`, с помощью которого задается каталог, содержащий файлы проекта. Возможно, вы захотите разместить проект в одном из подкаталогов интегрированной среды разработки VC++ (C:\Program Files\Microsoft Visual Studio). В этом случае соответствующий параметр должен иметь значение NULL.

Массив `mod[] = "C:\\MATLAB6p1\\work\\dataplot.exe"` указывает на то, что создаваемое приложение хранится в файле `dataplot.exe`, содержащемся на диске C: в каталоге `\\MATLAB6p1\\work`. Этот массив передается функции `CreateProcess()`.

Функция `CreateProcess()` используется для выполнения в оконной среде исполняемого файла Matlab `dataplot.exe`. В функции Matlab `dataplot.m` мы извлекаем данные из файла `mdata.txt`, а затем вызываем функцию `plot()` для построения графика.

Скомпилированную программу можно запускать на выполнение. График, создаваемый в процессе ее выполнения, показан на рис. 3.2.

Исходные коды данного примера находятся в папке `Chapter 3\\MatlabPlot` на прилагаемом компакт-диске.

3.4. Использование Matlab Add-In

Метод II. Для встраивания функций Matlab в программную среду VC++ используется Matlab Add-In. В среде VC++ функции Matlab могут быть модифицированы.

3.4.1. Использование M-файлов в Visual C++

При разработке многих приложений Matlab применяются средства Visual C++. Взаимодействие Matlab и C/C++ организуется путем преобразования M-файлов в независимые программы, создания и отладки файлов C-MEX для Matlab и интеграции M-файлов в приложения C++.

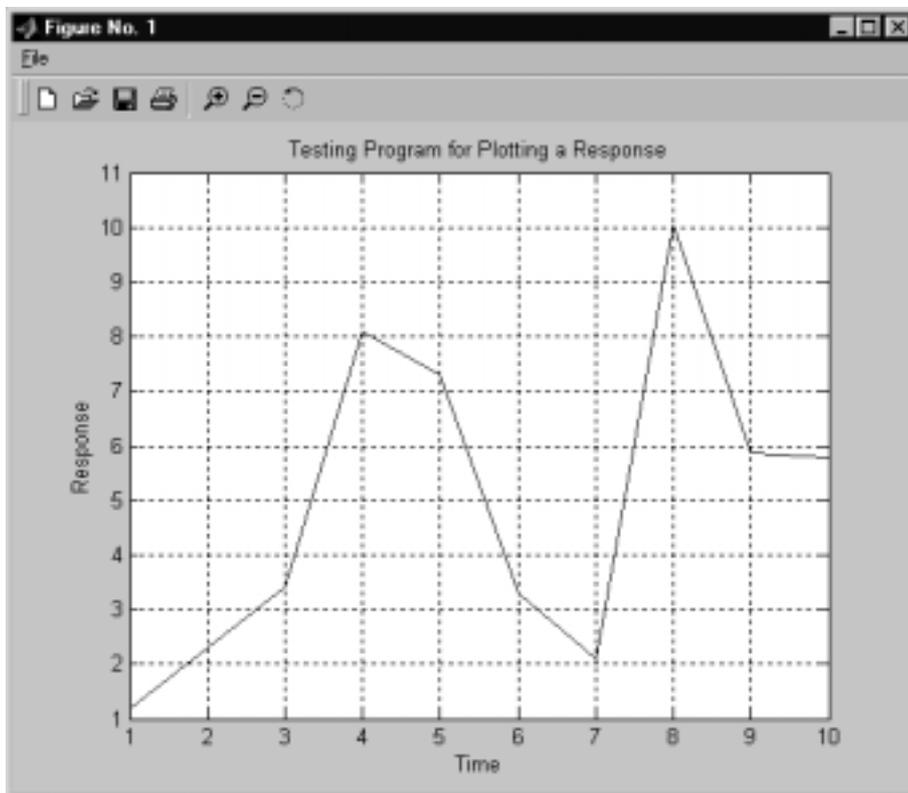


Рис. 3.2. График, формируемый в процессе работы приложения

Средства Matlab Add-In for Visual Studio автоматизируют задачу включения кода Matlab в программы на Visual C++. В проекте на Visual C++ устанавливаются пути к библиотекам и файлам заголовков, определяются необходимые константы и задаются библиотеки, необходимые для использования совместно с скомпилированными М-файлами. В результате работа над проектами упрощается. Matlab Add-In использует команды MСС и MEX, исключая необходимость вызова их вручную.

Matlab Add-In представляет собой мощный инструмент, который может применяться разработчиками. Создание приложений, включающих средства Matlab и C/C++, существенно упрощается за счет интеграции Matlab с Visual C++.

Благодаря использованию Matlab Add-In for Visual Studio вы получаете следующие возможности.

- Включать М-файлы в существующие проекты Visual C++.
- Создавать на базе М-файлов независимые приложения C или C++.
- Создавать файлы C-MEX или М-MEX.
- Создавать на базе М-файлов разделяемые библиотеки или МEX-файлы.

- Повторно использовать М-файлы в приложениях на C/C++.
- Непосредственно отлаживать исходные коды М-файлов. Альтернативой данному решению является отладка файлов C/C++, сгенерированных с помощью компилятора.
- Автоматизировать процесс редактирования М-файлов и компиляции их в файлы на C/C++. Редактирование и компиляция осуществляются в среде Visual C++.
- Проверять в процессе отладки данные, содержащиеся в составе матриц. Для этой цели используется Visual Matrix Viewer.
- Оформлять готовые программы в виде дистрибутивных пакетов.

Перед тем как продолжать изучение материала данного раздела, необходимо выполнить следующие действия.

- Инсталлировать Matlab 6, Matlab Compiler Suite и Visual C++ 6.0.
- Запустить `mbuild - setup` в командном окне Matlab или в окне DOS. Это необходимо, если вы собираетесь создавать независимые приложения. Посредством меню надо указать версию компилятора Microsoft Visual C/C++, инсталлированного на вашем компьютере.
- Запустить в окне DOS `mex - setup`. Это необходимо, если вы собираетесь создавать MEX-файлы для Visual Studio.
- Вызвать в командном окне Matlab `cd (prefdir); mcsavepath;`. Эти команды сохраняют текущий путь к Matlab в файле с именем `mcsrpath` в папке `user preferences`. Поскольку Matlab Add-In for Visual Studio выполняется за пределами среды Matlab, путь к Matlab необходимо сохранить. Этот путь использует Visual Studio при взаимодействии с Matlab. Каталог `prefdir` применяется по умолчанию для хранения пути к Matlab. По желанию вы можете заменить `prefdir` любым другим каталогом.
- Сконфигурировать Matlab Add-In for Visual Studio для работы с Microsoft Visual C/C++. Для этого необходимо выполнить следующие действия.
 - Выбрать в меню MSVC пункт Tools⇒Customize, открыв тем самым диалоговое окно Customize.
 - В появившемся диалоговом окне выбрать вкладку Add-ins and Macro Files.
 - Установить флажок MATLAB Add-in в списке Add-ins and Macro Files и закрыть диалоговое окно, щелкнув на кнопке Close.
 - В результате выполненных действий в окне Visual Studio отобразится плавающая панель инструментов Add-In for Visual Studio. Поскольку вы установили флажок опции, при последующих запусках MSVC будет автоматически загружаться соответствующий дополнительный модуль.

Внешний вид диалогового окна Customize показан на рис. 3.3.

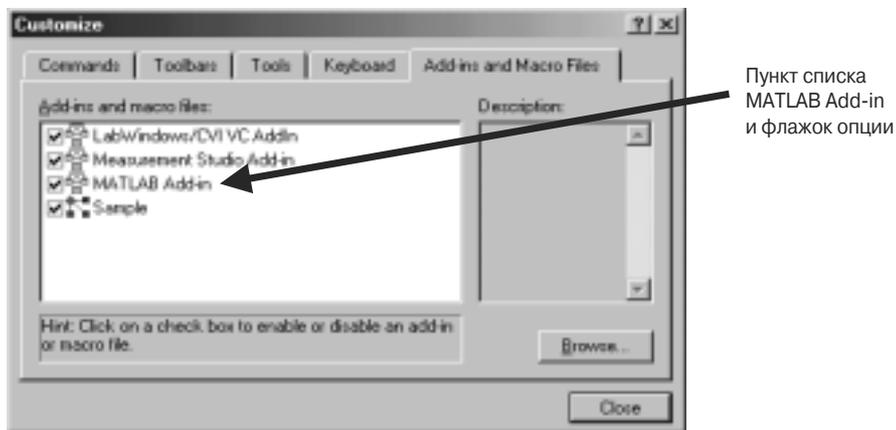


Рис. 3.3. Диалоговое окно Customize

Настройка Matlab Add-In for Visual Studio в Windows 95/98

Добавьте в файл `config.sys` следующую строку:

```
shell=C:\command.com/e: 32768/p
```

Настройка Matlab Add-In for Visual Studio в Windows Me

- Найдите с помощью Windows Explorer файл `C:\windows\system\conagent.exe`.
- Щелкните правой кнопкой мыши на пиктограмме `conagent.exe`.
- В появившемся меню выберите пункт `Properties`, чтобы открыть окно свойств `Conagent.exe`.
- Выберите в этом окне вкладку `Memory`.
- Установите в поле `Initial Environment` значение `4096`.
- Щелкните на кнопке `Apply`.
- Щелкните на кнопке `OK`, чтобы закрыть окно.

На этом этапе мы готовы к созданию программ с использованием Matlab Add-In for Visual Studio.

3.4.2. MATLAB Project Wizard

MATLAB Project Wizard автоматизирует создание проектов Visual C++. Этот инструмент позволяет выбрать тип приложения, генерируемого в рамках проекта, и указать M-файлы для включения в проект. Используя MATLAB Project Wizard, достаточно просто сгенерировать код приложения и запустить его на выполнение.

3.4.3. Панель инструментов MATLAB Add-In

Главные компоненты Visual Matlab доступны с помощью панели инструментов MATLAB Add-In. Эта панель инструментов представляет собой компонент интегрированной среды разработки Visual C++. Внешний вид панели показан на рис. 3.4.

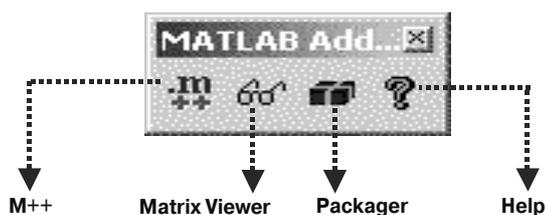


Рис. 3.4. Панели инструментов MATLAB Add-In

Ниже описано назначение каждой из кнопок.

- M++ включает в существующий проект дополнительные M-файлы.
- Matrix Viewer предоставляет доступ к средству просмотра матриц. Этот инструмент можно использовать для контроля содержимого матриц при отладке M-файлов или проектов C-MEX.
- Packager позволяет оформлять готовые программы в виде дистрибутивных пакетов.
- Кнопка Help открывает справочные данные Visual Matlab.

3.4.4. Matrix Viewer

Инструмент Matrix Viewer позволяет просматривать матрицы Matlab при отладке приложений Visual C++, сгенерированных на базе M-файлов, либо программ, содержащих файлы C-MEX. Matrix Viewer поддерживает большинство типов Matlab (матрицы, содержащие целочисленные значения, или числа с плавающей точкой, массивы ячеек, структуры и т.д.) и представляет данные в удобной для восприятия форме. Данный инструмент обеспечивает навигацию по массивам ячеек и структурам (рис. 3.5).

Для того чтобы просмотреть переменную в приложении Visual C++, надо щелкнуть на кнопке Matrix Viewer.

Чтобы просмотреть переменную в сеансе Matlab, надо использовать выражение `mviewer имя_переменной`.

3.4.5. Packager

Инструмент Packager обеспечивает средства для организации исполняемых файлов и DLL в виде пакета. Все компоненты приложения помещаются в один упакованный zip-файл. Данный инструмент предоставляет дружественный интерфейс (диалоговое окно Application Packager показано на рис. 3.6).

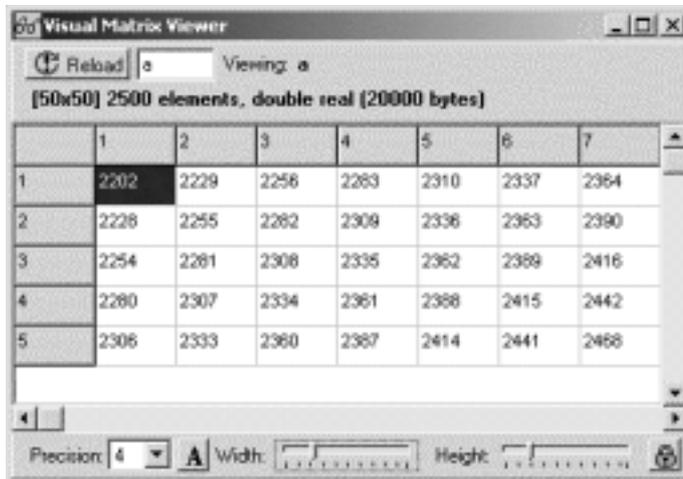


Рис. 3.5. Окно Matrix Viewer

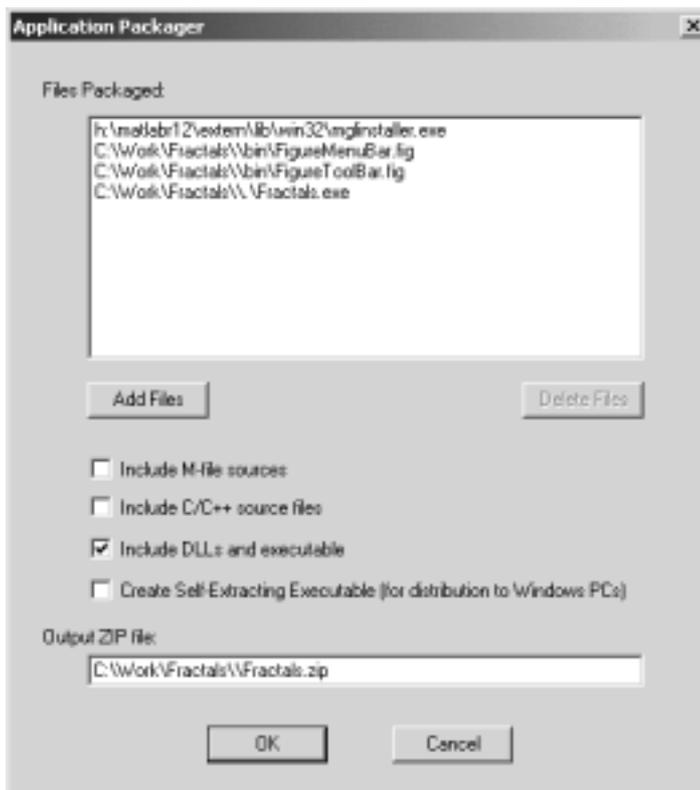


Рис. 3.6. Диалоговое окно Application Packager

Диалоговое окно MATLAB Project Wizard показано на рис. 3.7.

Создадим программу Matlab Visual Add-In. Для этой цели воспользуемся М-функцией `dataplot.m`, созданной в предыдущем разделе. Данный компонент мы интегрируем в среду MSVC, реализовав таким образом средства построения графиков.

Этап 1. Создание проекта Visual C++ с М-файлами

На данном этапе мы сгенерируем проект Visual C++, содержащий М-файл `dataplot.m`, и преобразуем М-файл в файл C/C++.

- Запустите Visual C++. Выберите пункт меню `File⇒New⇒Projects`.
- Выберите `MATLAB Project Wizard` и задайте имя проекта `AddInMatlab`. Щелкните на кнопке `OK`, в результате чего на экране появится диалоговое окно `MATLAB Project Wizard`.
- Не изменяя установки, предложенные по умолчанию, щелкните на кнопке `Finish`.
- Щелкните на кнопке `OK` в диалоговом окне `New Project Information`.

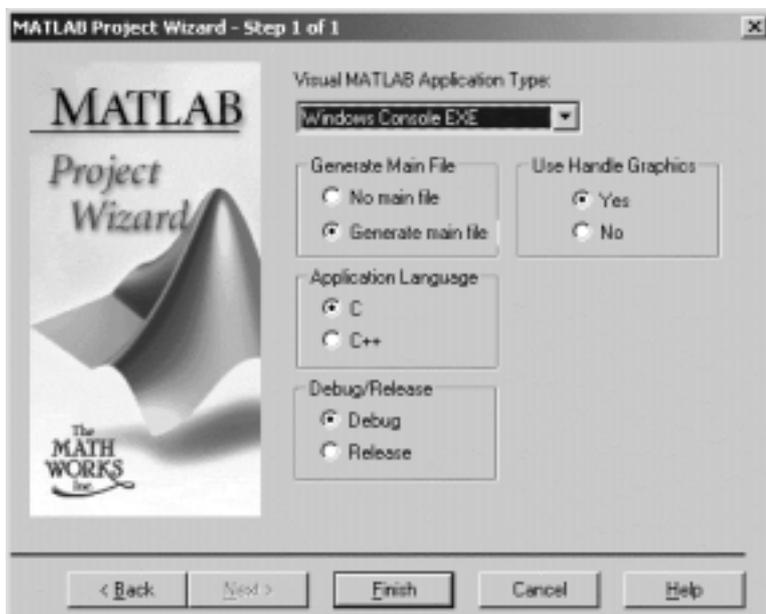
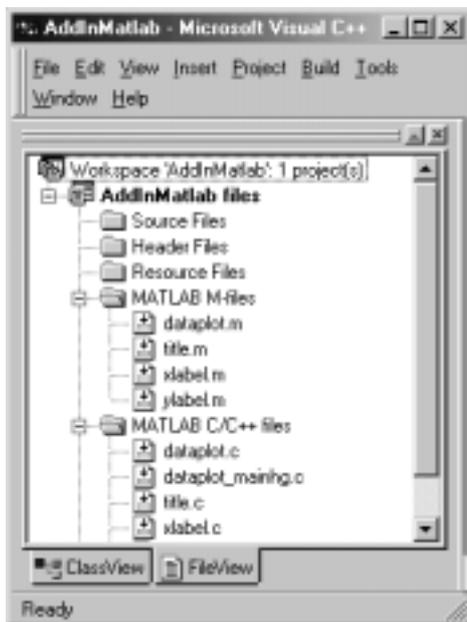


Рис. 3.7. Диалоговое окно MATLAB Project Wizard*

* Источник: Math Works, Inc. September, 2000. Using an Integrated Development Environment. MATLAB Compiler Documentation, pp. 4–19. Печатается с разрешения. — Прим. авт.

- В ответ на предложение задать М-файл выберите в каталоге C:\MATLAB6P1\work файл `dataplot.m` и щелкните на кнопке `Open`.
- Visual Matlab сгенерирует проект Visual C++ с М-файлами (это может занять несколько секунд). Вкладка `FileView` в рабочей области Visual C++ должна выглядеть так, как показано на рис. 3.8. Matlab Visual Add-In также создает пустой файл `AddInMatlab.c`. Исходный С-файл, созданный Matlab Appwizard, показан в листинге 3.3.

Рис. 3.8. Внешний вид вкладки `FileView` в рабочей области Visual C++
Листинг 3.3. Содержимое файла, созданного с помощью Matlab Appwizard

```

/*
 * MATLAB Compiler: 2.2
 * Date: Fri Apr 12 17:44:40 2002
 * Arguments: "-B" "macro_default" "-O" "all" "-O"
 "fold_scalar_mxarrays:on" "-O"
 "fold_non_scalar_mxarrays:on" "-O"
 "optimize_integer_for_loops:on" "-O"
 "array_indexing:on" "-O" "optimize_conditionals:on" "-B"
"C:\WINDOWS\Application
Data\MathWorks\MATLAB\R12\mccpath"
"-I" "C:\MATLAB6P1\toolbox\matlab\general"
"-I" "C:\MATLAB6P1\toolbox\matlab\ops" "-I"
"C:\MATLAB6P1\toolbox\matlab\lang"
"-I" "C:\MATLAB6P1\toolbox\matlab\elmat"
"-I" "C:\MATLAB6P1\toolbox\matlab\elfun"
"-I" "C:\MATLAB6P1\toolbox\matlab\specfun"

```

```

-I" "C:\MATLAB6P1\toolbox\matlab\matfun"
-I" "C:\MATLAB6P1\toolbox\matlab\datafun"
-I" "C:\MATLAB6P1\toolbox\matlab\audio"
-I" "C:\MATLAB6P1\toolbox\matlab\polyfun"
-I" "C:\MATLAB6P1\toolbox\matlab\funfun"
-I" "C:\MATLAB6P1\toolbox\matlab\sparsfun"
-I" "C:\MATLAB6P1\toolbox\matlab\graph2d"
-I" "C:\MATLAB6P1\toolbox\matlab\graph3d"
-I" "C:\MATLAB6P1\toolbox\matlab\specgraph"
-I" "C:\MATLAB6P1\toolbox\matlab\graphics"
-I" "C:\MATLAB6P1\toolbox\matlab\uitools"
-I" "C:\MATLAB6P1\toolbox\matlab\strfun"
-I" "C:\MATLAB6P1\toolbox\matlab\iofun"
-I" "C:\MATLAB6P1\toolbox\matlab\timefun"
-I" "C:\MATLAB6P1\toolbox\matlab\datatypes"
-I" "C:\MATLAB6P1\toolbox\matlab\verctrl"
-I" "C:\MATLAB6P1\toolbox\matlab\winfun"
-I" "C:\MATLAB6P1\toolbox\matlab\demos"
-I" "C:\MATLAB6P1\toolbox\local"
-I" "C:\MATLAB6P1\toolbox\compiler"
-I" "C:\MATLAB6P1\work" "-k" "C:\Chapter
3\AddInMatlab\mcc.mak" "-/n" "-B" "sgl" "-m" "-W" "main" "-L" "C" "-t"
-T" "link:exe" "-h" "libmmfile.mlib" "-W" "mainhg" "libmwsglm.mlib"
-A" "line:on" "-g" "-G" "-A" "debugline:on"
-O" "none" "-O" "fold_scalar_mxarrays:off"
-O" "fold_non_scalar_mxarrays:off"
-O" "optimize_integer_for_loops:off"
-O" "array_indexing:off"
-O" "optimize_conditionals:off" "libmmfile.mlib" "libmwsglm.mlib" "-
v" "-h" "C:\MATLAB6p1\work\dataplot.m"
*/
#include "dataplot.h"
#include "libmatlbm.h"
#include "libmmfile.h"
#include "title.h"
#include "xlabel.h"
#include "ylabel.h"
void InitializeModule_dataplot(void)
{
}
void TerminateModule_dataplot(void)
{
}
static void Mdataplot(void);
_mexLocalFunctionTable _local_function_table_dataplot = { 0,
(mexFunctionTableEntry *)NULL };
/*
 * The function "mlfDataplot" contains the normal interface for the
"dataplot" M-function from file "C:\MATLAB6P1\work\dataplot.m" (lines
1-19). This function processes any input arguments and passes them to
the implementation version of the function, appearing above. */
void mlfDataplot(void) {
    mlfEnterNewContext(0, 0);
    Mdataplot();
    mlfRestorePreviousContext(0, 0);
}

```

Исходный код выглядит достаточно громоздким, но это не должно волновать вас. Всю сложную работу по его подготовке выполняют инструментальные средства. Разработчику не приходится прилагать усилия для обеспечения интеграции. Инструмент Matlab Visual Add-in преобразует М-функции в С-код. Результатом преобразования являются следующие файлы заголовков.

- `dataplot.h`
- `title.h`
- `xlabel.h`
- `ylabel.h`

В процессе компиляции используются также файлы `libmatlbm.h` и `libmmfile.h`. Эти файлы заголовков предоставляют прототипы функций, содержащихся в библиотеке.

Этап 2. Компиляция и запуск программы

На этом этапе осуществляется компиляция сгенерированных программ и проверка их работоспособности.

- Скомпилируйте проект в среде Visual C++. Для этого надо нажать клавишу <F7> или выбрать пункт меню Build⇒Build.
- Запустите приложение, чтобы убедиться, что оно выполняется корректно. Для этого нажмите клавишу <F5> или выберите пункт меню Build⇒Execute.

На экране должен отобразиться график, подобный тому, который был представлен на рис. 3.2.

Этап 3. Отладка и просмотр содержимого переменных

При отладке приложения происходит выявление и исправление ошибок. Для просмотра переменных, представляющих матрицы, можно использовать инструмент Matrix Viewer (рис. 3.9).

- Откройте файл `dataplot.m` для редактирования.
- Установите в требуемых позициях точки останова.
- Запустите программу. Дойдя до точки останова, программа прекратит выполнение.
- Вызовите Matrix Viewer, щелкнув на пиктограмме с изображением очков. Эта пиктограмма располагается на панели инструментов MATLAB Add-in.
- В поле редактирования, расположенном в окне Matrix Viewer, введите X и щелкните на кнопке Reload. На экране отобразится содержимое матрицы X.
- Вернитесь в среду Visual C++ и проверьте М-код.

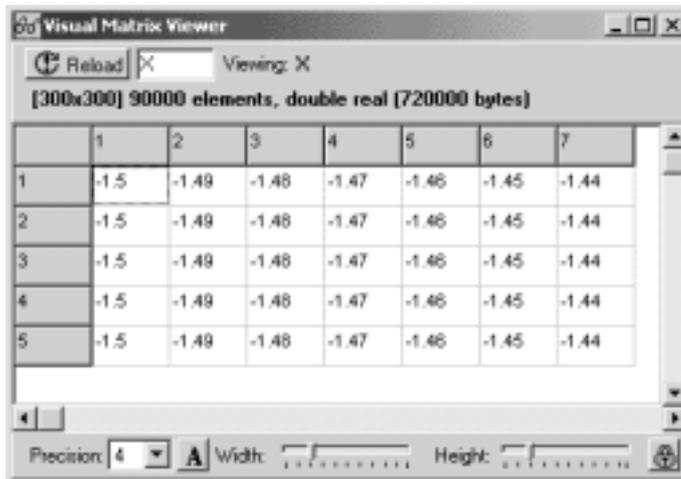


Рис. 3.9. Использование Matrix Viewer для просмотра данных

Этап 4. Создание дистрибутивного пакета

Для того чтобы подготовить приложение к доставке, надо оформить все файлы в виде одного zip-архива. Активизируйте инструмент Packager, щелкнув на пиктограмме Packager панели инструментов MATLAB Add-in. Для того чтобы начать создание пакета, щелкните на кнопке ОК (рис. 3.10). Инструмент Packager генерирует файл dataplot.zip и помещает его в каталог проекта. Созданный zip-архив содержит все файлы приложения.

Файлы рассмотренного выше проекта находятся на прилагаемом компакт-диске в каталоге Chapter 3\AddInMatlab. Проект называется AddInMatlab.

3.5. Метод Matlab Engine

Метод III. Данный метод эквивалентен методу ActiveX и предполагает использование библиотеки Matlab Engine для взаимодействия с функциями Matlab.

3.5.1. Взаимодействие с Matlab

В системе Unix библиотека взаимодействует с Matlab с помощью механизма каналов. По мере необходимости используются средства удаленного вызова rsh. В Microsoft Windows взаимодействие с Matlab происходит посредством ActiveX. Детальное описание ActiveX содержится в документации.

При использовании библиотеки Matlab Engine интерфейс между VC++ и Matlab существенно упрощается. Подобно управляющим элементам ActiveX, после обращения к среде Matlab и выполнения некоторых действий VC++ сохраняет контроль над выполнением программ. Еще одно преимущество использования библиотеки Matlab Engine состоит в том, что при этом нет необхо-

димости хранить данные в файле в среде VC++ и извлекать их, открывая файл в среде Matlab. Вместо этого вы можете сформировать данные в программе VC++ и вызывать М-функции, передавая им информацию в виде массивов.

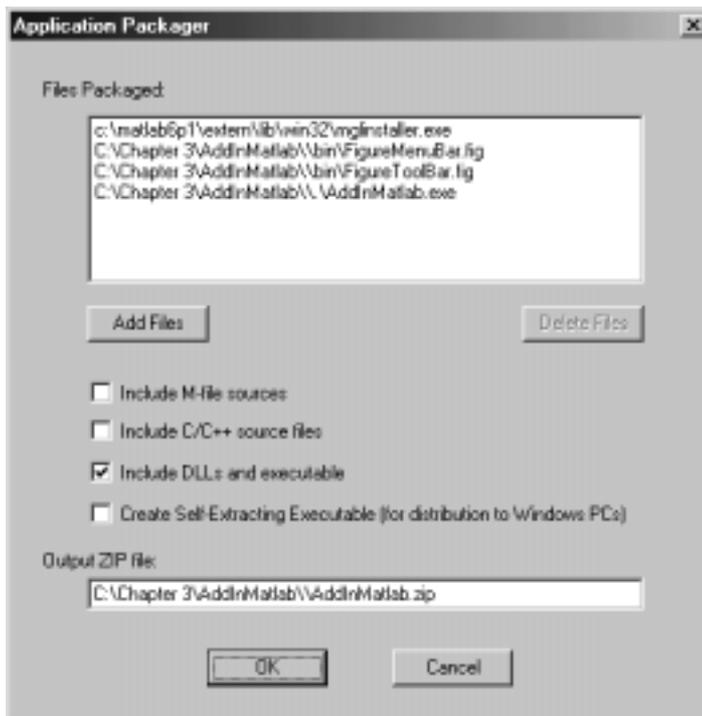


Рис. 3.10. Окно Application Packager

В табл. 3.1 перечислены некоторые функции общего назначения, которые могут быть применены в программной среде C в системе Microsoft Windows.

Таблица 3.1. Функции C Engine

Функция	Назначение
engOpen	Запуск Matlab Engine
engClose	Прекращение работы Matlab Engine
engGetArray	Получение от Matlab Engine массива Matlab
engPutArray	Передача Matlab Engine массива Matlab
engEvalString	Выполнение команды Matlab
engOutputBuffer	Создание буфера для хранения выходных данных Matlab
engOpenSingleUse	Создание сеанса Matlab Engine для монопольного использования

Необходимо представлять себе последовательность действий, которые надо выполнить для того, чтобы воспользоваться функциями Matlab Engine. Например, перед использованием `engPutArray` надо создать матрицу, присвоить ей имя и организовать ее заполнение.

3.5.2. Пример использования библиотеки Matlab Engine

Пример, который будет рассмотрен ниже, демонстрирует использование Matlab Engine для вызова М-функции, осуществляющей построение двух графиков: синусоиды и косинусоиды.

Данная программа компилируется в исполняемый файл (`.exe`) с помощью Matlab Compiler и может выполняться в среде Visual C++ или непосредственно в системе Windows.

Запустите VC++ 6.0 и создайте новый проект типа Win32 Console Application. В качестве имени проекта введите в поле Project name: значение `mEngPlot`. Щелкните на кнопке ОК, инициировав тем самым создание нового проекта `mEngPlot`.

Для того чтобы включить в проект исходный файл на языке C, выполните следующие действия.

- Выберите пункт меню Project⇒Add To Project⇒New.
- В диалоговом окне File выберите тип Text File. Введите в поле File name: имя файла `mEngPlot`. Щелкните на кнопке ОК, чтобы закрыть диалоговое окно.
- Откройте диалоговое окно Insert Files into Project, выбрав пункт меню Project⇒Add To Project⇒Files.
- В диалоговом окне щелкните на стрелке, расположенной рядом с полем Files of type:, и выберите последний пункт списка, All Files (*.*) .
- Щелкните правой кнопкой мыши на вновь созданном текстовом файле `mEngPlot.txt` и выберите в контекстном меню пункт Rename. Измените расширение текстового файла с `.txt` на `.c`.
- В появившемся окне с сообщением щелкните на кнопке Yes, чтобы закрыть это окно.
- В результате выполненных действий файл `mEngPlot.txt` будет переименован в `mEngPlot.c`.
- Щелкните на новом файле, а затем на кнопке ОК, чтобы добавить этот файл к проекту.
- По мере необходимости вы можете удалить текстовый файл `mEngPlot.txt` из проекта. Для этого надо щелкнуть на нем в области FileView, а затем выбрать меню Edit и пункт Delete.

В первую очередь нам надо сформировать два сигнала: синусоидальный и косинусоидальный и сохранить их в каталоге `C:\data` в файлах `sdata` и `fdata`. В реальном приложении мы получим эти сигналы от серводвигателя, управляющего потоком жидкости.

Один набор данных будет соответствовать скорости двигателя, а другой — скорости потока. Поскольку мы ставим задачу продемонстрировать способ взаимодействия программ, мы ограничиваемся лишь имитацией реальных данных.

3.5.2.1. Файл заголовка

Щелкните на пункте меню File⇒New, чтобы открыть диалоговое окно New. По умолчанию в этом окне выбирается вкладка Files. Щелкните на пункте списка C/C++ Header File и введите в поле File name: имя файла заголовков mEngPlot. Файл заголовков необходим для объявления прототипов данных и функций, используемых в проекте. Содержимое файла заголовков показано в листинге 3.4.

Листинг 3.4. Файл заголовков mEngPlot.h

```
#ifndef MENG_PLOT_H
#define MENG_PLOT_H

#define PI 3.14159
#define cycletime 5
#define ITEMS 200

double sArray[200];
double fArray[200];

int dataSource(int unit);
int dataPlot(int unit, double* sdataArray, double* fddataArray);

#endif
```

Два массива используются для хранения соответственно синусоидального и косинусоидального сигналов. Каждый из массивов включает 200 элементов. Функция dataSource() используется для создания наборов данных, имитирующих входные сигналы. В данном случае параметр unit не применяется, но в реальном приложении он указывает единицы, в которых измеряется скорость потока. Параметр unit может принимать одно из восьми допустимых значений. Функция dataPlot() вызывается из основной программы и строит графики, исходными данными для которых является содержимое массивов sdataArray и fddataArray.

3.5.2.2. Определение функций

Код функции dataSource() показан в листинге 3.5. Цикл for используется для того, чтобы заполнить массивы данными, соответствующими синусоидальному и косинусоидальному сигналу. Возвращаемое значение ret равно нулю, что означает отсутствие ошибок. Код функции dataPlot() показан в листинге 3.6.

Листинг 3.5. Функция dataSource()

```
int dataSource(int unit)
{
    int n, ret = 0;

    for (n = 0; n < ITEMS, n++)
    {
        sArray[n] = sin((2 * PI/ITEMS) * n);
        fArray[n] = cos((2 * PI/ITEMS) * n);
    }

    return ret;
}
```

Листинг 3.6. Функция dataPlot()

```
int dataPlot(int unit, double* sdataArray, double* fdataArray)
{
    int rc = 0;
    int i, j;
    Engine *ep;
    mxArray *T = NULL, *D = NULL, *F = NULL;
    double time[ITEMS], sdata[ITEMS], fdata[ITEMS];

    for (i = 0; i < ITEMS; i++)
    {
        time[i] = 20 * i; // Интервал между получением
                        // очередных данных равен 20 миллисекундам
        sdata[i] = sdataArray[i]; // Получение данных о скорости
        fdata[i] = fdataArray[i]; // Получение характеристики
                                // потока
    }

    /* Запуск Matlab Engine
    for (j = 0; j < cycletime; j++)
    {
        if (!(ep = engOpen(NULL)))
        {
            rc = MessageBox((HWND)NULL, (LPSTR)"Can't start MATLAB
engine",
                (LPSTR) "mEngPlot.c", MB_RETRYCANCEL);
            if (rc == IDCANCEL)
                return (1);
            Sleep(5);
        }
        else
            break;
    }

    /* Создание переменной T на базе массива
    T = mxCreateDoubleMatrix(1, 200, mxREAL);
    mxSetName(T, "T");
    memcpy((char *) mxGetPr(T), (char *) time, 200*sizeof(double));
```

```

    /** Переменная T помещается в рабочую область Matlab
    engPutArray(ep, T);

    /** Создание переменной D на основании данных о скорости
    D = mxCreateDoubleMatrix(1, 200, mxREAL);
    mxSetName(D, "D");
    memcpy((char *) mxGetPr(D), (char *) sdata, 200*sizeof(double));

    /** Переменная D помещается в рабочую область Matlab
    engPutArray(ep, D);

    /** Создание переменной F на основании данных о потоке
    F = mxCreateDoubleMatrix(1, 200, mxREAL);
    mxSetName(F, "F");
    memcpy((char *) mxGetPr(F), (char *) fdata, 200*sizeof(double));

    /** Переменная F помещается в рабочую область Matlab
    engPutArray(ep, F);

    /** Вывод результатов в виде графика
    engEvalString(ep, "plot(T,D,'b-',T,F,'g-');");
    engEvalString(ep, "grid");
    engEvalString(ep, "title('Velocity, Flow-Rate vs Time of the Smart
Motor');");
    engEvalString(ep, "xlabel('Time (ms)');");
    engEvalString(ep, "ylabel('Motor Velocity (RPM) & Flow-Rate
(Ml/Min)');");
    engEvalString(ep, "legend('Motor Velocity (RPM)', 'Flow-Rate
(ml/min)', 0);");

    /** Освобождение памяти, прекращение работы
    // Matlab Engine и завершение программы
    MessageBox((HWND)NULL, (LPSTR)"PLOT IS SUCCESSFUL!",
    (LPSTR)"mEngPlot.c", MB_OK);
    engClose(ep);
    mxDestroyArray(T);
    mxDestroyArray(D);
    mxDestroyArray(F);

    return(0);
}

```

Объекты Engine и mxArray определены в среде Matlab. Для их определения в Matlab создаются файлы заголовков engine.h и matrix.h. В данном приложении мы преобразуем массивы данных в другие массивы: sdata[] и fdata[]. Если такой подход вас не устраивает, следовать ему не обязательно.

После преобразования массивов данных вызывается системная функция engOpen(), предназначенная для активизации Matlab Engine. Чтобы исключить случайную ошибку, мы пытаемся установить в цикле for соединение пять раз. Если при установлении соединения возникнет ошибка, отобразится окно с соответствующим сообщением. В случае успешного соединения цикл прерывается и Matlab Engine продолжает работу.

Функция `mxCreateDoubleMatrix()` вызывается для того, чтобы создать массив данных `T`, насчитывающий 200 элементов (`mxREAL` соответствует реальным данным в матрице Matlab). Функция `mxSetName()` присваивает массиву данных `T` имя "T". Функция `memcpy()` копирует данные из среды C в среду Matlab.

Функция `engPutArray(ep, T)` вызывается для того, чтобы установить массив данных `T` в среде Matlab.

Вслед за массивом `T` аналогичным образом создаются массивы `D` и `S`. Данные копируются из `sdata` и `fdata` в среду Matlab.

Вызов `engEvalString(ep, "plot(T,D,'b-',T,F,'g-');")` помещает функцию построения графика в виде строки в рабочую область Matlab. Данная функция используется для формирования кривых, отражающих характеристики двигателя (`sdata`) и потока (`fdata`) на одном графике. Кривые отображаются соответственно синим и зеленым цветом. Вызов `engEvalString(ep, "plot(T,D,'b-',T,F,'g-');")` позволяет представить обычную функцию в виде строки, поместить в среду Matlab и выполнить ее. Таким образом, мы можем пометить на графике оси `x` и `y`.

Перед выходом из рабочей области Matlab отображается сообщение. Когда пользователь щелкнет на кнопке ОК, функция построения графика завершится. С этого момента до окончания работы программы остается лишь выполнить действия по освобождению ресурсов. Функция `engClose(ep)` закрывает Matlab Engine, а `mxDestroyArray()` используется для очистки области памяти, занимаемой массивами `T`, `D` и `S`.

Код функции `main()` данного приложения показан в листинге 3.7. Действия, выполняемые этой функцией, предельно просты. Сначала `main()` обращается к функции `dataSource()`, которая создает два массива: `sArray` и `fArray`, а затем вызывает функцию `dataPlot()`, которая, используя Matlab Engine, строит в среде Matlab графики, соответствующие массивам.

Листинг 3.7. Файл `mEngPlot.c`, содержащий функцию `main()`

```

/*-----
 * Файл:      mEngPlot.c
 * Описание:  программа, демонстрирующая вызов функций
 *            Matlab Engine из программы на C
 *            и построение графика.
 * Дата:      6/6/01
 *-----*/

#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>
#include "engine.h"
#include "mEngPlot.h"

int main(void)
{

```

```
int unit = 1;

dataSource(unit);
dataPlot(unit, sArray, fArray);

return (0);
}
```

Помимо файлов заголовков, обсуждавшихся ранее, в проект входят следующие файлы.

- basetsd.h
- engine.h
- matrix.h
- tmwtypes.h

Файлы заголовков размещаются в каталоге `C:\MATLAB6p1\extern\include`. Их надо скопировать в локальный каталог, предназначенный для включаемых файлов, и указать этот каталог на вкладке `Directories` в окне `Options`. Для того чтобы отобразить диалоговое окно `Options`, в окне `Visual C++` выберите пункт `Options` меню `Tools`. Добавить файлы заголовков к проекту можно также, вызвав пункт меню `Add To Project`⇒`Files`.

На вкладке `Link` окна `Project Settings` надо указать приведенные ниже библиотечные файлы (чтобы отобразить окно `Project Settings`, выберите пункт `Settings` меню `Project`).

- libeng.lib
- libmx.lib

Оба указанных файла находятся в каталоге

```
C:\MATLAB6p1\extern\lib\win32\Microsoft\msvc60
```

Скопируйте эти файлы в локальный каталог `lib` и укажите путь к библиотекам на вкладке `Directories` окна `Options`. Для этой цели вы можете также выбрать пункт `Project`⇒`Add To Project`⇒`Files`.

В рассмотренной программе для вызова `M`-функций используется строковый формат. Окно с сообщением, которое выводится в конце программы, позволяет удержать на экране результат построения графика. При отсутствии этой функции графики быстро исчезли бы с экрана.

После построения и запуска программы вы увидите графики, отображающие синусоиду и косинусоиду. Щелкнув на кнопке `ОК`, вы закроете окно с сообщением, после чего выполнение приложения завершится.

В листинге 3.8 показан код данного проекта. Исходные файлы расположены на прилагаемом компакт-диске в папке `Chapter 3\mEngPlot`. Имя проекта — `mEngPlot`.

Листинг 3.8. Полный C-код проекта

```

/***** Файл заголовков *****/
#ifndef MENGPlot_H
#define MENGPlot_H

#define PI 3.14159
#define cycletime 5
#define ITEMS 200

double sArray[200];
double fArray[200];

int dataSource(int unit);
int dataPlot(int unit, double* sdataArray, double* fdataArray);

#endif
/***** Основная функция *****/
* Файл: mEngPlot.c
* Описание: программа, демонстрирующая вызов функций
* Matlab Engine из программы на C
* и построение графика.
* Дата: 4/12/2002
* Автор: Y. Bai
*-----*/
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <math.h>
#include "engine.h"
#include "mEngPlot.h"

int main(void)
{
    dataSource(1);
    dataPlot(1, sArray, fArray);

    delete [] sArray;
    delete [] fArray;
    return (0);
}

int dataSource(int unit)
{
    int n, ret = 0;

    for (n = 0; n < ITEMS, n++)
    {
        sArray[n] = sin((2 * PI/ITEMS) * n);
        fArray[n] = cos((2 * PI/ITEMS) * n);
    }
    return ret;
}

```

```
}

int dataPlot(int unit, double* sdataArray, double* fddataArray)
{
    int rc = 0;
    int i, j;
    Engine *ep;
    mxArray *T = NULL, *D = NULL, *F = NULL;
    double time[ITEMS], sdata[ITEMS], fddataArray[ITEMS];

    for (i = 0; i < ITEMS; i++)
    {
        time[i] = 20 * i; // Интервал между получением
                        // очередных данных равен 20 миллисекундам
        sdata[i] = sdataArray[i]; // Получение данных о скорости
        fddataArray[i] = fddataArray[i]; // Получение характеристики потока
    }

    /* Запуск Matlab Engine
    for (j = 0; j < cycletime; j++)
    {
        if (!(ep = engOpen(NULL)))
        {
            rc = MessageBox((HWND)NULL,
                (LPSTR)"Can't start MATLAB engine",
                (LPSTR) "mEngPlot.c", MB_RETRYCANCEL);
            if (rc == IDCANCEL)
                return (1);
            Sleep(5);
        }
        else
            break;
    }

    /* Создание переменной T на базе массива
    T = mxCreateDoubleMatrix(1, 200, mxREAL);
    mxSetName(T, "T");
    memcpy((char *) mxGetPr(T), (char *) time, 200*sizeof(double));

    /* Переменная T помещается в рабочую область Matlab
    engPutArray(ep, T);

    /* Создание переменной D на основании данных о скорости
    D = mxCreateDoubleMatrix(1, 200, mxREAL);
    mxSetName(D, "D");
    memcpy((char *) mxGetPr(D), (char *) sdata,
        200*sizeof(double));

    /* Переменная D помещается в рабочую область Matlab
    engPutArray(ep, D);

    /* Создание переменной F на основании данных о потоке
    F = mxCreateDoubleMatrix(1, 200, mxREAL);
    mxSetName(F, "F");
    memcpy((char *) mxGetPr(F), (char *) fddataArray,
        200*sizeof(double));
```

```

    /* Переменная F помещается в рабочую область Matlab
    engPutArray(ep, F);

    /* Вывод результатов в виде графика
    engEvalString(ep, "plot(T,D,'b-',T,F,'g-');");
    engEvalString(ep, "grid");
    engEvalString(ep,
        "title('Velocity, Flow-Rate vs Time of the Smart Motor');");
    engEvalString(ep, "xlabel('Time (ms)');");
    engEvalString(ep,
        "ylabel('Motor Velocity (RPM) & Flow-Rate (ml/Min)');");
    engEvalString(ep,
        "legend('Motor Velocity (RPM)', 'Flow-Rate (ml/min)',0);");

    // Освобождение памяти, прекращение работы
    // Matlab Engine и завершение программы
    MessageBox((HWND)NULL, (LPSTR)"PLOT IS SUCCESSFUL!",
        (LPSTR)"mEngPlot.c", MB_OK);
    engClose(ep);
    mxDestroyArray(T);
    mxDestroyArray(D);
    mxDestroyArray(F);

    return(0);
}

```

3.6. Оформление функций Matlab в виде DLL

Метод IV. Функция Matlab создается в виде разделяемой библиотеки DLL, к которой осуществляется обращение из программной среды VC++. Основные этапы работы по формированию библиотеки описаны ниже.

Этап 1. Создайте в среде Matlab M-функцию `dataplotdll.m`, код которой приведен в листинге 3.9.

Чтобы не усложнять данный пример, мы применим в нем массив данных, состоящий из 10 элементов. Массив генерируется в среде Visual C++ и используется в среде Matlab. Данные, составляющие массив, хранятся в файле `dlldata.txt`, который находится в каталоге `C:\data`. M-функция `dataplotdll()` извлекает эти данные, используя функцию `fscanf()`, и помещает их в массив `[s]`, принадлежащий рабочей области Matlab (листинг 3.9).

Листинг 3.9. M-функция `dataplotdll()`

```

% Функция для представления данных в виде графика.
% Входная информация: массив данных s
% Выходная информация: отсутствует
% Файл: dataplotdll.m

function dataplotdll()
fid = fopen('C:\data\dlldata.txt', 'r');
[s] = fscanf(fid, '%f ', [1, 200]);

```

```

plot(s);
grid
xlabel('Time');
title('Testing DLL Program for Plotting a Response');
ylabel('Response');

```

В переменной `fid` хранится дескриптор файла, который возвращает функция `fopen()`. Этот дескриптор записывается в переменную в том случае, если файл был открыт успешно. Программа, созданная в среде Visual C++, обращается к М-функции так же, как и к обычной функции динамической библиотеки. Преобразование М-функции в динамическую библиотеку осуществляется на этапе 2.

Этап 2. Преобразуйте М-функцию в DLL с помощью Matlab Compiler 2.2.

Для этой цели необходимо выполнить приведенную команду

```
mcc -B sgl -t -W libhg:dataplotlib -T link:lib dataplotdll.m
```

В результате преобразования генерируются следующие файлы.

<i>Файлы заголовков</i>	<i>Исходные файлы</i>	<i>Библиотечные файлы</i>	<i>Файлы DLL</i>
grid.h	dataplotdll.c	dataplotlib.lib	dataplotlib.dll
xlabel.h	dataplotlib.c		
ylabel.h			
title.h			
dataplotdll.h			
dataplotlib.h			

В файлах заголовков содержатся описания прототипов функций, сгенерированных Matlab Compiler. Библиотечный файл `dataplotlib.lib` обеспечивает связь программы на Visual C++ с динамической библиотекой. Файл `dataplotlib.dll` представляет собой динамическую библиотеку, используемую в процессе работы программы на Visual C++.

Этап 3. Создайте консольное приложение, которое будет обращаться к динамической библиотеке, сгенерированной Matlab Compiler. В данном примере для этой цели открывается проект с именем `MatlabPlotdll`.

Загрузите Visual C++ 6.0 и создайте новый проект, задав в поле `Project name:` имя `MatlabPlotdll`. Создайте новый файл `MatlabPlotdll.c` с исходным кодом на языке C. Процесс создания C-файла был описан ранее.

Скопируйте с помощью Windows Explorer в папку проекта `MatlabPlotdll` следующие файлы заголовков.

```

dataplotlib.h    matrix.h
libmatlb.h      mex.h
libmatlbm.h     mwutil.h
libsgl.h        tmwtypes.h

```

Некоторые системные файлы заголовков хранятся в каталоге `$(Root)\MATLAB6p1\extern\include`.

Щелкните на пункте меню Project⇒Add To Project⇒Files, выделите все указанные выше файлы заголовков (выделяя несколько файлов, удерживайте нажатой клавишу <Ctrl>) и щелкните на кнопке ОК, чтобы добавить файлы заголовков к проекту.

Для того чтобы включить в проект библиотечный файл dataplotlib.lib, выполните следующие действия.

- Скопируйте файл dataplotlib.lib в папку проекта MatlabPlotdll.
- Выберите пункт меню Project⇒Add To Project⇒Files, щелкните на стрелке, расположенной рядом с полем Files of type:, и выберите последний пункт списка, All Files (*.*). Найдите в списке файлов библиотеку dataplotlib.lib и выделите соответствующий пункт списка. Щелкните на кнопке ОК, чтобы добавить выбранный файл к проекту.

Введите в файл MatlabPlotdll.c код, показанный в листинге 3.10.

Листинг 3.10. Содержимое файла MatlabPlotdll.c

```

/*-----
* Имя файла: MatlabPlotdll.c
* Описание: вызывает DLL, созданную посредством
*          Matlab compiler 2.1, для построения графика
* Дата:      4/12/2002
* Автор:     Y. Bai
-----*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "dataplotlib.h"

#define PI 3.14159

double data_array[200];
FILE* fp;

void main()
{
    int i;
    fp = fopen(" C:\\data\\dlldata.txt ", "w "); // В этом
                                                // каталоге хранятся данные
    for ( i = 0; i < 200; i ++ )
    {
        data_array[i] = 10 * sin(( 2 * PI/200) * i );
        fprintf( fp, " %f ", data_array[i]);
    }
    fclose( fp);

    /* Следующая функция вызывается для инициализации
       dataplotdll. Эта функция создается посредством Matlab
       Compiler и помещается в dataplotlib.dll */
    dataplotlibInitialize();
}

```

```

/* Данная функция обеспечивает доступ к DLL, созданной
   посредством Matlab Compiler */
mlfDataplotdll();

/* Приведенный ниже вызов завершает работу dataplotdll,
   созданной посредством Matlab Compiler и находящейся
   в dataplotlib.dll */
dataplotlibTerminate();

return;
}

```

Файл `dataplotlib.h`, созданный компилятором при преобразовании М-функции в динамическую библиотеку, очень важен. Содержимое этого файла показано в листинге 3.11. Файл `dataplotlib.h` включает прототипы всех функций, которые были определены в среде Matlab и должны использоваться программой на Visual C++. Некоторые из этих функций описаны ниже.

Листинг 3.11. Содержимое файла `dataplotlib.h`, сгенерированного при преобразовании М-функции

```

/* MATLAB Compiler: 2.2
 * Date: Fri Apr 12 22:06:07 2002
 * Arguments: "-B" "macro_default" "-O" "all" "-O"
 * "fold_scalar_mxarrays:on"
 * "-O" "fold_non_scalar_mxarrays:on" "-O"
 * "optimize_integer_for_loops:on" "-O"
 * "array_indexing:on" "-O" "optimize_conditionals:on"
 * "-B" "sgl" "-m" "-W" "main" "-L" "C" "-t" "-T"
 * "link:exe" "-h" "libmmfile.mlib" "-W" "mainhg"
 * "libmwsglm.mlib" "-t" "-W" "libhg:dataplotlib"
 * "-T" "link:lib" "dataplotdll.m"
 */
#ifndef MLF_V2
#define MLF_V2 1
#endif

#ifndef __dataplotlib_h
#define __dataplotlib_h 1

#ifdef __cplusplus
extern "C" {
#endif

#include "libmatlb.h"

extern void InitializeModule_dataplotlib(void);
extern void TerminateModule_dataplotlib(void);
extern void mlfDataplotdll(void);
extern void mlxDataplotdll(int nlhs,
                           mxArray * plhs[],
                           int nrhs,
                           mxArray * prhs[]);

```

```

extern mxArray * mlfNTitle(int nargout, mxArray * string, ...);
extern mxArray * mlfTitle(mxArray * string, ...);
extern void mlfVTitle(mxArray * string, ...);
extern void mlxTitle(int nlhs, mxArray * plhs[],
                    int nrhs, mxArray * prhs[]);
extern mxArray * mlfNXlabel(int nargout, mxArray * string, ...);
extern mxArray * mlfXlabel(mxArray * string, ...);
extern void mlfVXlabel(mxArray * string, ...);
extern void mlxXlabel(int nlhs, mxArray * plhs[],
                    int nrhs, mxArray * prhs[]);
extern mxArray * mlfNYlabel(int nargout, mxArray * string, ...);
extern mxArray * mlfYlabel(mxArray * string, ...);
extern void mlfVYlabel(mxArray * string, ...);
extern void mlxYlabel(int nlhs, mxArray * plhs[],
                    int nrhs, mxArray * prhs[]);
extern void dataplotlibInitialize(void);
extern void dataplotlibTerminate(void);

#ifdef __cplusplus
}
#endif

#endif

```

Функция `mlfDataplotdll()` создается Matlab Compiler и представляет собой интерфейс для взаимодействия с М-функцией `dataplotdll()`. Эта функция размещается в файле `dataplotlib.dll`, соответственно ее исходный код находится в файле `dataplotlib.c`. По этой причине нельзя непосредственно обращаться к функции `dataplotdll()` из программы на VC++, вместо этого надо использовать функцию `mlfDataplotdll()`.

Перед вызовом функции `mlfDataplotdll()` надо вызвать функцию `dataplotlibInitialize()`, а после ее завершения — функцию `dataplotlibTerminate()`. Эти функции осуществляют соответственно инициализацию среды для выполнения `mlfDataplotdll()` и освобождение ресурсов, использованных в процессе работы.

Этап 4. Напишите программу, которая будет использовать функции DLL для формирования графика, и запустите ее.

Перед запуском программы скопируйте файл динамической библиотеки (`dataplotlib.dll`) в один из каталогов, в которых система осуществляет поиск файлов. В данном примере файл DLL находится в каталоге `C:\Matlabdll`, и, чтобы система имела возможность обнаружить его, вам надо указать этот путь в соответствующем системном файле. Если вы не хотите создавать новую папку для библиотечного файла `dataplotlib.dll`, скопируйте его в тот каталог, в котором находятся системные разделяемые библиотеки, например в `C:\WINDOWS\SYSTEM`. В этом каталоге находится большинство системных DLL. Выполняя поиск DLL-файлов, система в первую очередь проверяет данный каталог.

Исходные коды рассматриваемого примера находятся в каталоге `Chapter 3\MatlabPlotdll` на прилагаемом к книге компакт-диске.

В некоторых случаях для идентификации точек входа DLL в проект VC++ приходится включать файл определений. Пример содержимого DEF-файла приведен ниже.

```
LIBRARY dataplotlib.dll
EXPORTS
mlfDataplotdll
mlxDataplotdll
dataplotlibInitialize
dataplotlibTerminate
```

3.7. Использование Matlab для поддержки операций над матрицами

3.7.1. Введение

Ранее мы сосредоточили внимание на приложениях, предназначенных для формирования графиков. Эти приложения базировались на совместной работе средств VC++ и Matlab. Использование средств Matlab в программах на VC++ позволяет также выполнять сложные операции над многомерными матрицами.

Диаграмма, представленная на рис. 3.11, иллюстрирует принципы подобного взаимодействия. Перед обращением к функциям Matlab, предназначенным для обработки матриц, программа, созданная на VC++, должна подготовить данные, в частности собрать информацию, предоставляемую аппаратными средствами, и создать файлы данных в формате ASCII. После этого выполняются действия с помощью Matlab. Matlab извлекает данные из файлов, подготовленных программой на VC++, а затем выполняет операции над матрицами. Результаты передаются программе на VC++ через файлы. Пользуясь результатами, предоставленными Matlab, программа на VC++ может осуществлять управление устройствами, подключенными к компьютеру.

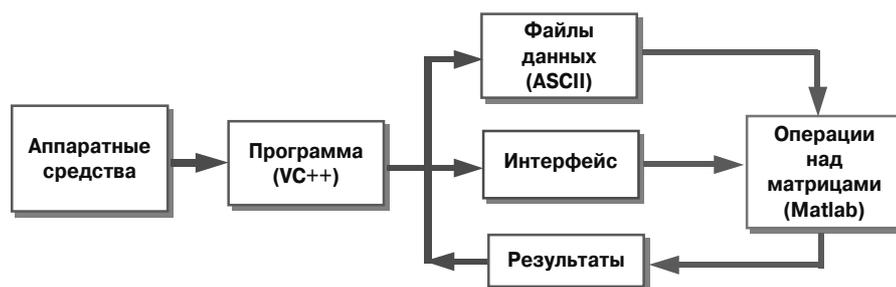


Рис. 3.11. Взаимодействие средств VC++ и Matlab

Таким образом, обработка матриц осуществляется в два этапа. В первую очередь программа на VC++ подготавливает данные (они могут быть сгенерированы на аппаратном уровне) и сохраняет информацию в файлах. После этого выполнение программы на VC++ приостанавливается и средства Matlab начинают

обработку данных, читая их из файлов и выполняя требуемые операции. Очевидно, что такой способ не обеспечивает высокой эффективности работы. В частности, он не позволяет управлять системами в реальном масштабе времени.

Используя средства взаимодействия VC++ и Matlab, можно организовать более удобный режим обработки матриц и обеспечить работу в реальном времени. Ниже рассматривается пример использования независимого приложения для организации совместной работы компонентов Matlab и MSVC и выполнения с их помощью операций над матрицами.

3.7.2. Математические проблемы и их решение

В некоторых случаях при решении инженерных задач необходимо выполнять сложные операции над матрицами. Для этой цели удобно использовать математические функции Matlab. В частности, Matlab предоставляет обширный набор функций для выполнения операций над матрицами. Здесь мы рассмотрим лишь простой пример, иллюстрирующий вызов M-функций, выполняющих требуемые действия, из среды MSVC.

Однородные матрицы часто применяются при калибровке манипуляторов роботов. С их помощью осуществляется перевод позиции эффектора из одной координатной системы в другую. Иногда целевая позиция в одной координатной системе уже известна и необходимо лишь представить ее в другой системе.

Для того чтобы упростить задачу, рассмотрим преобразование координат одной точки в трехмерном пространстве. Эта точка будет представлять позицию эффектора. Ее можно определить с помощью приведенных ниже уравнений. Процесс преобразования координат также демонстрируется на рис. 3.12.

$$\begin{aligned} z &= r * \sin(\theta_1) \\ y &= r * \cos(\theta_1) * \sin(\theta_2) \\ x &= r * \cos(\theta_1) * \cos(\theta_2) \end{aligned}$$

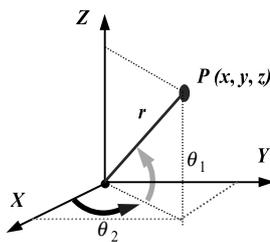


Рис. 3.12. Преобразование координат точки

Если известны длина вектора r и два угла, θ_1 и θ_2 , несложно определить позицию P в декартовой системе координат. Иногда новая координатная система формируется путем вращения исходной системы координат вокруг одной из осей (например, оси z). В этом случае также приходится находить позицию точки P в целевой системе координат. Для того чтобы выполнить подобную операцию, используется однородная матрица преобразования.

Предположим, например, что имеется система координат UVW, которая получена путем вращения исходной системы вокруг оси z на угол Θ , и точка P имеет в системе UVW координаты u , v и w . Зависимость между координатами в разных системах можно выразить с помощью следующего выражения:

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 & 0 \\ \sin \Theta & \cos \Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_{z,\Theta} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Здесь $T_{z,\Theta}$ называется однородной матрицей преобразования. С ее помощью можно отобразить позицию точки P из системы XYZ в систему UVW.

3.7.3. Однородная матрица преобразования

Возможна ситуация, когда мы знаем позицию точки P в системе координат UVW и нам требуется определить координаты той же точки в системе XYZ. Для этой цели нам нужна обратная матрица $T_{z,\Theta}^{-1}$. Функции, предоставляемые Matlab, позволяют без труда решать проблемы подобного рода. С помощью Matlab мы можем получить псевдообратную матрицу, пригодную для инженерных расчетов.

Посредством программы, написанной в среде MSVC, определим позицию точки P в системе координат XYZ, а также создадим однородную матрицу преобразования. В качестве параметра будет задаваться угол. Позицию P , значение угла поворота для матрицы преобразования и индекс операции над матрицей мы разместим в текстовом файле. Индекс 0 означает операции над обычными матрицами, а значение 1 — операции над обратными матрицами. Для того чтобы найти позицию в системе координат XYZ по известной позиции в системе UVW, необходимо выполнить операцию над обратной матрицей. Для этой цели будем использовать M-функцию, преобразованную посредством Matlab Compiler в исполняемый файл.

Пример, который мы рассмотрим ниже, значительно проще, чем задачи, которые приходится решать на практике. Для реального приложения придется собирать в реальном времени 30–50 значений, определяющих различные позиции эффектора робота, и сохранять их в файле данных. Каждая позиция представляет собой однородную матрицу размерностью 4×4 , поэтому нам надо иметь до 50 матриц. Такой набор значений необходим для того, чтобы точно определить параметры робота. Без использования Matlab реализовать необходимые операции над матрицами было бы очень трудно.

3.7.4. Операции над однородными матрицами

Прежде всего нам надо создать М-функцию, которая обеспечивала бы преобразование координат из одной системы в другую. Выполнение этой функции можно условно разбить на три этапа.

- Сбор данных. Параметры извлекаются из файла, созданного программой на MSVC. К числу необходимых параметров относятся следующие.
 - Индекс операции, который определяет действия с обычной или обратной матрицей.
 - Угол поворота вокруг оси z в системе XYZ.
 - Однородная матрица преобразования.
- Преобразование вектора, определяющего позицию.
- Запись вектора позиции в файл данных. Программа на MSVC откроет этот файл и прочитает результаты работы М-функции.

Исходный код М-функции показан в листинге 3.12. В коде функции условно выделены четыре раздела.

Листинг 3.12. Код М-функции MatlabMatrix()

```
% М-функция MatlabMatrix
% Программа MSVC сохраняет вектор, определяющий позицию и
% однородную матрицу преобразования в файле данных.
% 4/7/2002
function MatlabMatrix()
clear all;
inp_Pos = zeros(4, 1);    % Определение векторов и матрицы
out_Pos = zeros(4, 1);
Trans = zeros(4, 4);

fid = fopen('C:\data\mtxdata.txt','r');
[T, COUNT] = fscanf(fid,'%f ', [4, 6]);
S = T';
mIndex = S(1, 1);        % Извлечение индекса
mAngle = S(1, 2);        % Извлечение угла поворота
inp_Pos = S(2, :);       % Извлечение исходного вектора,
                        % определяющего позицию
Trans    = S(3:6, :);    % Извлечение однородной матрицы
                        % преобразования

Trans(1, 1) = Trans(1, 1) * cos(mAngle);    % Получение реальной
                                           % матрицы
Trans(1, 2) = Trans(1, 2) * sin(mAngle);
Trans(2, 1) = Trans(2, 1) * sin(mAngle);
Trans(2, 2) = Trans(2, 2) * cos(mAngle);

if mIndex == 0
    out_Pos = Trans * inp_Pos';            % Операция с обычной матрицей
else
```

```
    out_Pos = inv(Trans) * inp_Pos'; % Операция с обратной матрицей
end

fid = fopen('C:\data\cMatrix.txt','w'); % сохранение
    % результирующего вектора в файле данных
fprintf(fid,'%f %f %f %d\n', out_Pos);

fclose(fid);
```

- Определение переменных для векторов и матриц и чтение данных из файла.

Для определения двух векторов, соответствующих позиции, `inp_Pos` и `out_Pos`, используется функции `zeros()`. Здесь же объявляется однородная матрица преобразования размерностью 4×4 , элементы которой инициализируются нулевыми значениями.

Файл данных открывается с помощью функции `fopen()`, а функция `fscanf()` извлекает значения из файла `mtxdata.txt`, созданного в среде MSVC. Полученные данные помещаются в матрицу `T`. Заметьте, что оператор транспонирования преобразует исходную матрицу данных (6×4) в матрицу 4×6 . Такое преобразование необходимо для выполнения функции `fscanf()`. Затем с помощью выражения `S = T'` восстанавливается исходный вариант матрицы.

Обратите внимание, что при загрузке файла данных в среду Matlab и связывании его с переменной необходимо указывать полный путь. Кроме того, независимо от типа данных, содержащихся в файле, полный путь к нему помещается в одинарные кавычки.

Необходимо обратить внимание еще на одну особенность M-функции. Для загрузки файла данных можно было бы использовать функцию `load`, однако она работает только в среде Matlab. Если бы мы последовали такому подходу, то при выполнении функции в рабочей области Matlab проблемы бы не возникли. Но после компиляции функции в исполняемый файл и вызова его из программы MSVC мы получили бы сообщение об ошибке `Cannot open the data file`. Поэтому, создавая интерфейс между MSVC и Matlab, мы используем функции `fopen()` и `fscanf()`, которые похожи на одноименные C-функции.

- Извлечение параметров из файла данных. После загрузки мы извлекаем следующие четыре параметра.
 - Индекс операции над матрицей, `mIndex`. Этот целочисленный параметр указывает на то, должна ли выполняться операция над обычной или над обратной матрицей. Значение индекса, равное 0, соответствует преобразованию из системы XYZ в систему UVW. Для преобразования из системы UVW в систему XYZ используется обратная матрица, поэтому индекс должен быть равен 1.

- Угол поворота, `mAngle`. Данный параметр указывает угол в радианах, который должен использоваться для получения матрицы. Поскольку мы не хотим ограничиваться фиксированным значением угла, то для обеспечения гибкости вычислим матрицу преобразования в `M`-функции.
- Исходный вектор позиции, `inp_Pos`, предоставленный программой на MSVC. В реальном приложении для формирования нескольких векторов в реальном масштабе времени используются специальные инструменты сбора данных. В данном примере мы ограничиваемся одним вектором.
- Однородная матрица преобразования, `Trans`. Матрица, хранящаяся в файле данных, представляет собой лишь прототип, или базовую модель. Для получения результирующей матрицы используется параметр `mAngle`.

В операциях над матрицей применяется двоеточие (:). Оно может использоваться двумя способами. Во-первых, с помощью двоеточия определяется диапазон, в который должны попадать элементы, извлекаемые из матрицы. Набор элементов, принадлежащих этому диапазону, рассматривается как подматрица. Например, выражение `3:6` означает, что результирующая подматрица содержит с третьей по шестую строку исходной матрицы. Кроме того, с помощью двоеточия можно указывать все строки или столбцы матрицы. Например, выражение `(1,:)` определяет все элементы первой строки (все столбцы первой строки) исходной матрицы.

В данном примере нам надо извлечь элементы исходной матрицы, принадлежащие всем столбцам с третьей по шестую строку. Поэтому для представления данной операции используется выражение `S(3:6, :)`. Для получения результирующей матрицы умножим элементы входной матрицы на функции от угла поворота `mAngle`.

- Выполнение операции над матрицами для получения результирующего вектора. В данном случае операции достаточно просты. Сначала проверим индекс, чтобы выяснить тип операции. Так как нам надо получить вектор в системе координат XYZ, используется операция над обратной матрицей; соответственно указывается индекс, равный 1.

Кроме того, необходимо указать размерность входного вектора. Когда этот вектор создается в программе MSVC, его размерность равна 1×4 . Для того чтобы умножить матрицу преобразования 4×4 на вектор 1×4 , последний должен быть преобразован в вектор 4×1 . В Matlab для обозначения такого преобразования используется апостроф (символ `'`).

- Сохранение вычисленной позиции в файле данных. При получении результатов вычислений их надо сохранить в файле данных, доступном из программы на MSVC. Информация, содержащаяся в файле, используется в дальнейшем MSVC-программой для выполнения задачи управления. В Matlab форматированный вывод в файл выполняет функция `fprintf()`. Эта функция вызывается следующим образом:

```
fprintf(fid, '%.3f %.3f %.3f %d\n', out_Pos);
```

Ниже описывается назначение параметров функции.

- `fid` — это дескриптор целевого файла, возвращаемый функцией `fopen()`.
- Последовательность символов `%.3f` представляет формат, в котором хранятся данные. Данное выражение указывает на то, что компоненты результирующего вектора должны выводиться с точностью до третьего знака после десятичной точки. Исключением является последнее значение, которое выводится в целочисленном формате.
- Параметр `out_Pos` — это результирующий вектор. Matlab позволяет непосредственно включать в вызов функции `fprintf()` вектор или матрицу и сохранять их в нужном формате. Для сохранения данных можно также использовать функцию `save`, которая записывается в формате `save('имя_файла', 'значение_1', 'значение_2', ...)`. Эта функция позволяет сохранить данные в случае, если имя файла и имена переменных представлены в строковом виде.

Перед тем как переходить к работе с MSVC, нам надо преобразовать М-функцию в исполняемый файл, используя для этой цели Matlab Compiler. В командном окне Matlab вызовите Matlab Compiler (`mcc`) и преобразуйте с его помощью М-функцию в независимое приложение (опция `-m` указывает на то, что М-файл должен быть преобразован в код С).

```
mcc -m MatlabMatrix.m
```

Ниже приведены файлы, созданные в результате выполнения данной команды.

```
MatlabMatrix.c  MatlabMatrix.h  MatlabMatrix.exe
title.c         title.h
xlabel.c        xlabel.h
ylabel.c        ylabel.h
```

В данном случае нам нужен только файл `MatlabMatrix.exe`, представляющий собой независимое приложение.

3.7.5. Преобразование координат путем вызова М-функции

Загрузите MSVC++ 6.0 и создайте новый проект. Укажите тип проекта `Win32 Console Application` и введите в поле `Project name:` имя `MatlabMatrix`. Создайте новый исходный файл С, указав в поле `File name:` имя `MatlabMatrix`. В качестве содержимого файла введите код, показанный в листинге 3.13. Текст программы можно условно разделить на несколько частей.

Листинг 3.13. Функция main() в файле MatlabMatrix.c

```

/*****
* Файл:      MatlabMatrix.c
* Описание:  проверка операций над матрицами путем вызова
*            исполняемого файла, созданного посредством
*            Matlab Compiler.
* Дата:      4/12/2002.
* Автор:     Y. Bai
*****/
#include <cstdio>
#include <cstdlib>
#include <cassert>
#include <iostream>
#include <windows.h>

using namespace std;

PROCESS_INFORMATION    proInfo;
STARTUPINFO            startInfo;

int dataSource(double thet, int index);
void initStartupInfo(STARTUPINFO info);
int readVector(int dTime, char* revFile);

void main()
{
    int ret = 0;
    long ErrCode;
    // Путь к файлу
    char mod[] = "C:\\MATLAB6p1\\work\\MatlabMatrix.exe";
    char rev[] = "C:\\data\\cMatrix.txt";

    ret = dataSource( 0.78539, 1);
    if ( ret != 0)
    {
        cout << "ERROR in dataSource!\n    " << endl;
        return;
    }
    initStartupInfo(startInfo);

    // Запуск процесса
    ErrCode =
        CreateProcess( mod ,NULL,NULL,NULL,TRUE,0,NULL,NULL,
                      &startInfo,&proInfo);

    // Проверка на наличие ошибок
    if (!ErrCode)
    {
        ErrCode = GetLastError();
        cout << " CreateProcess is failed & ErrCode = "
              << ErrCode << endl;
    }

    // Получение результирующего вектора

```

```
if (readVector( 100, rev) != 0)
    cout << " ERROR in readVector \n " << endl;

return;
}
```

- Объявление переменных и функций, используемых при выполнении программы.
- Создание информации, необходимой в работе.
 - Индекс операции над матрицей, `mIndex`.
 - Угол поворота, `mAngle`.
 - Однородная матрица преобразования, `Trans`.
- Сохранение информации в файле данных, используемом M-функцией. Данные, содержащиеся в файле, применяются для выполнения операции над матрицей и вычисления вектора, представляющего позицию.
- Обращение к исполняемому файлу, созданному Matlab Compiler, для выполнения операции над матрицей в среде Matlab.
- Извлечение из файла данных `cMatrix.txt` результирующего вектора, подготовленного M-функцией. Указанный файл данных находится в каталоге `C:\data`.
- Отображение результирующего вектора. Отображаемые данные позволяют убедиться в правильности выполнения операции.

В файле `MatlabMatrix.c` содержится стандартный исходный код C++, причем в нем используется новый стиль файлов заголовков. Все они располагаются в стандартной библиотеке C++ в пространстве имен `std`. Выражение `using namespace std` сообщает компилятору C++ о том, что мы собираемся использовать элементы из этого пространства имен. Переменные, принадлежащие типам `PROCESS_INFORMATION` и `STARTUPINFO`, используются при вызове функции `CreateProcess()`.

В теле функции `main()` вызываются три функции. С помощью этих вызовов выполняются следующие действия.

1. Создание исходных данных.
2. Инициализация структуры для выполнения установок.
3. Подтверждение правильности результатов, полученных из среды Matlab.

Первая из вызываемых функций, `dataSource()`, подготавливает информацию, необходимую для выполнения операций над матрицами. Эта функция помещает подготовленные параметры в файл данных. Впоследствии M-функция открывает файл данных, извлекает параметры и использует их для выполнения операций. Код функции `dataSource()` показан в листинге 3.14.

Листинг 3.14. Код функции `dataSource()`

```
int dataSource(double thet, int index)
{
    FILE* fp;
    // Получение указателя
    fp = fopen("C:\\data\\mtxdata.txt", "w");
    if (fp == NULL)
    {
        cout << "ERROR in fopen \n" << endl;
        return (-1);
    }
    // Сохранение данных в файле
    fprintf(fp, "%d %. 5f %d %d\n", index, thet, 0, 0);
    fprintf(fp, "%. 3f %. 3f %. 3f %d\n",
            15.887, 0.529, 3.779, 1);
    fprintf(fp, "%d %d %d %d\n", 1, -1, 0, 0);
    fprintf(fp, "%d %d %d %d\n", 1, 1, 0, 0);
    fprintf(fp, "%d %d %d %d\n", 0, 0, 1, 0);
    fprintf(fp, "%d %d %d %d\n", 0, 0, 0, 1);

    fclose(fp);
    return (0);
}
```

При вызове функции `dataSource()` передаются два параметра: угол поворота `thet`, представленный в радианах, и индекс операции над матрицей, `index`. Эти параметры формируются в теле функции `main()` и предназначены для записи в файл данных `mtxdata.txt`. В рассматриваемом примере мы выбираем угол поворота, равный 45° , и задаем индекс, равный 1. Заметьте, что угол должен быть представлен в радианах, поэтому при вызове функции ей передается значение $\pi/4$, т.е. 0,78539.

В начале своего выполнения функция `dataSource()` открывает в каталоге `C:\data` текстовый файл `mtxdata.txt`. Затем она вызывает системную функцию `fprintf()` для записи в файл следующих параметров.

- Угол поворота `thet`.
- Индекс операции над матрицей, `index`.
- Вектор, а представляющий позицию (компоненты вектора:

$$x = 15.887, y = 0.529, z = 3.779).$$

- Исходные данные для формирования однородной матрицы преобразования. Реальная матрица преобразования формируется в М-функции с использованием угла поворота `thet`.

Чтобы представить угол поворота с большей точностью, мы задаем отображение пяти значащих цифр после десятичной точки. При выводе остальных переменных типа `double` ограничиваемся тремя значащими цифрами. Индекс операции над матрицей является целым значением, поэтому он выводится как десятичное число.

Вектор, определяющий позицию, может быть задан в системе координат XYZ или в системе UVW. Поскольку мы проверяем выполнение операции над обратной матрицей, мы задаем координаты в системе UVW. Данный вектор предназначен лишь для проверки функционирования программы, поэтому вы можете указать для его элементов любые допустимые значения.

После создания файла данных в теле функции `main()` вызывается функция `initStartupInfo()`. Эта функция инициализирует структуру `STARTUPINFO`, которая используется в дальнейшем при обращении к функции `CreateProcess()`, вызове M-функции, представленной в формате исполняемого файла, и выполнении операций над матрицами. Код функции `initStartupInfo()` приведен в листинге 3.15.

Листинг 3.15. Код функции `initStartupInfo()`

```
void initStartupInfo(STARTUPINFO info)
{
    info.cb                = sizeof(STARTUPINFO);
    info.cbReserved2      = 0;
    info.dwFillAttribute  = BACKGROUND_GREEN;
    info.lpTitle           = NULL;
    info.dwX               = CW_USEDEFAULT;
    info.dwY               = CW_USEDEFAULT;
    info.dwXSize           = CW_USEDEFAULT;
    info.dwYSize           = CW_USEDEFAULT;
    info.dwXCountChars    = 0;
    info.dwYCountChars    = 0;
    info.lpReserved2      = NULL;
    info.lpReserved       = NULL;
    info.lpDesktop         = NULL;
    info.dwFlags           = STARTF_USEFILLATTRIBUTE;
    return;
}
```

Мы не будем уделять много внимания структуре `STARTUPINFO`, а ограничимся лишь кратким рассмотрением ее элементов. Переменная `cb` содержит информацию о размере структуры в байтах. Переменная `lpTitle` содержит строку, отображаемую в качестве заголовка окна. Поля структуры `dwX` и `dwY` представляют соответственно координаты x и y верхнего левого угла окна. Аналогично, `dwXSize` и `dwYSize` содержат ширину и высоту открытого окна. Зарезервированные переменные на сегодняшний день недоступны. Подробную информацию по данному вопросу предоставляет библиотека MSDN.

После функций, описанных выше, в теле `main()` вызывается системная функция `CreateProcess()`, которая запускает исполняемый файл `MatlabMatrix.exe`. С его помощью в среде Matlab вычисляется вектор, представляющий целевые координаты. Прототип функции `CreateProcess()` показан в листинге 3.16. Ниже описано назначение некоторых параметров этой функции.

Листинг 3.16. Прототип функции CreateProcess()

```

BOOL CreateProcess(
    LPCTSTR lpApplicationName, // Имя исполняемого модуля
    LPCTSTR lpCommandLine,    // Командная строка
    LPSECURITY_ATTRIBUTES lpProcessAttributes, // SD
    LPSECURITY_ATTRIBUTES lpThreadAttributes, // SD
    BOOL bInheritHandles,     // Поддержка наследования
    DWORD dwCreationFlags,    // Флаги, указываемые при создании
    LPVOID lpEnvironment,     // Новое окружение
    LPCTSTR lpCurrentDirectory, // Имя текущего каталога
    LPSTARTUPINFO lpStartupInfo, // Информация, используемая
                                // при запуске
    LPPROCESS_INFORMATION lpProcessInformation // Информация
                                // о процессе
);

```

- `lpApplicationName`
Указатель на строку, оканчивающуюся нулевым символом. Посредством этой строки задается имя исполняемого модуля. Данный параметр должен содержать полное имя файла, включающее путь.
- `lpCommandLine`
Указатель на строку, оканчивающуюся нулевым символом. В этой строке можно указать некоторые команды, которые должны выполняться приложением. Данная переменная может иметь значение NULL. В этом случае в качестве командной строки используется последовательность символов, на которую указывает `lpApplicationName`.
- `lpProcessAttributes`
Указатель на структуру `SECURITY_ATTRIBUTES`, которая определяет, наследуется ли возвращаемый дескриптор дочерними процессами. Если значение `lpProcessAttributes` равно NULL, дескриптор не наследуется.
- `lpStartupInfo`
Указатель на структуру `STARTUPINFO`, которая определяет поведение главного окна нового процесса.
- `lpProcessInformation`
Указатель на структуру `PROCESS_INFORMATION`, в которую помещается информация о новом процессе.

Пять описанных выше параметров являются основными для данной системной функции. Информацию об остальных параметрах можно получить, обратившись к библиотеке MSDN.

В нашем примере исполняемый файл находится в каталоге `C:\MATLAB6p1\work`, поэтому полное имя файла имеет вид `C:\MATLAB6p1\work\Matlab-Matrix.exe`. Данная строка должна быть задана в качестве параметра `lpApplicationName`. Кроме того, функции должны быть переданы структуры `STARTUPINFO` и `PROCESS_INFORMATION`.

Последней в теле `main()` вызывается функция `readVector()`. Эта функция применяется для получения результирующего вектора, определенного с помощью М-функции `MatlabMatrix`. Код функции `readVector()` показан в листинге 3.17.

Листинг 3.17. Код функции `readVector()`

```
int readVector(int dTime, char* revFile)
{
    int i;
    FILE* fprev;
    char rdata[4][16];

    Sleep(dTime);
    fprev = fopen(revFile, "r");
    if (fprev == NULL)
    {
        cout << "ERROR in fopen - fprev \n" << endl;
        return (-1);
    }

    for (i = 0; i < 4; i++)
    {
        fscanf(fprev, "%s\n", rdata[i]);
        cout << rdata[i] << endl;
    }
    fclose(fprev);

    return (0);
}
```

При вызове этой функции передаются два параметра. Первый параметр, `dTime`, определяет интервал времени в миллисекундах. Этот параметр предназначен для того, чтобы предоставить М-функции время, достаточное для завершения операций с матрицами и сохранения результирующего вектора в файле данных. Второй параметр представляет собой строку, которая содержит путь и имя файла данных. Строка `C:\data\cMatrix.txt`, представляющая путь и имя, определена в начале функции `main()`.

Функция `readVector()` открывает файл с данными, затем прекращает работу на время, заданное с помощью параметра `dTime`. Системная функция `fscanf()` используется для чтения результирующего вектора в массив `rdata`. В то же время этот вектор отображается на экране, предоставляя возможность проверить корректность выполнения операции. Заметьте, что, поскольку файл данных является текстовым файлом, вектор представляется в строковом виде. Если вам надо преобразовать элементы вектора в тип двойной точности, воспользуйтесь функцией `fprintf()`. На этом этапе мы завершили создание кода программы в среде MSVC. Полностью исходный код `main()` и других функций представлен в листинге 3.18.

Листинг 3.18. Исходный C-код приложения (файл MatlabMatrix.cpp)

```

/*****
* Файл:      MatlabMatrix.cpp
* Описание:  проверка операций над матрицами путем вызова
*            исполняемого файла, созданного посредством
*            Matlab Compiler.
* Дата:      4/7/2002.
*****/
#include <cstdio>
#include <cstdlib>
#include <cassert>
#include <iostream>
#include <windows.h>
using namespace std;
PROCESS_INFORMATION  proInfo ;
STARTUPINFO          startInfo;
int dataSource( double thet, int index);
void initStartupInfo( STARTUPINFO  info);
int readVector( int dTime , char* revFile);
void main()
{
    int ret = 0;
    long ErrCode;
    // Путь к файлу
    char mod [] = "C:\\MATLAB6p1\\work\\MatlabMatrix.exe"; //
    char rev[] = " C:\\data\\cMatrix.txt";

    ret = dataSource(0.78539, 1);
    if (ret != 0)
    {
        cout << "ERROR in dataSource!\n " << endl;
        return;
    }
    initStartupInfo (startInfo);
    // Запуск процесса
    ErrCode =
        CreateProcess( mod ,NULL,NULL,NULL,FALSE,0,NULL,NULL,
        & startInfo ,&proInfo );
    if (!ErrCode) // check error
    {
        ErrCode = GetLastError();
        cout << "CreateProcess is failed & ErrCode = "
            << ErrCode << endl;
    }
    if (readVector (100, rev) != 0) //Получение
        //результатирующего вектора
        cout << " ERROR in readVector \n " << endl;
    return;
}

int dataSource(double thet, int index)
{
    FILE* fp;
    fp = fopen("C:\\data\\mtxdata.txt", "w"); // Получение указателя

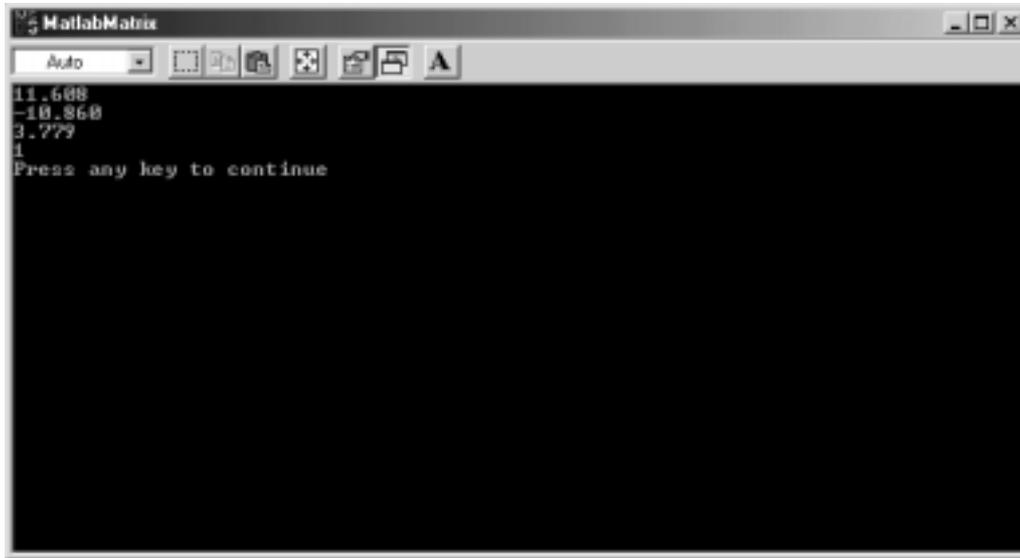
```

```

    if (fp == NULL)
    {
        cout << "ERROR in fopen \n" << endl;    return (-1);
    }
    // Сохранение данных в файле
    fprintf(fp, "%d %. 5f %d %d\n", index, thet, 0, 0);
    fprintf(fp, "%. 3f %. 3f %. 3f %d\n",
        15.887, 0.529, 3.779, 1);
    fprintf(fp, "%d %d %d %d\n", 1, -1, 0, 0);
    fprintf(fp, "%d %d %d %d\n", 1, 1, 0, 0);
    fprintf(fp, "%d %d %d %d\n", 0, 0, 1, 0);
    fprintf(fp, "%d %d %d %d\n", 0, 0, 0, 1);
    fclose(fp);
    return (0);
}
void initStartupInfo(STARTUPINFO info)
{
    info.cb                = sizeof(STARTUPINFO);
    info.cbReserved2      = 0;
    info.dwFillAttribute  = BACKGROUND_GREEN;
    info.lpTitle          = NULL;
    info.dwX              = CW_USEDEFAULT;
    info.dwY              = CW_USEDEFAULT;
    info.dwXSize          = CW_USEDEFAULT;
    info.dwYSize          = CW_USEDEFAULT;
    info.dwXCountChars    = 0;
    info.dwYCountChars    = 0;
    info.lpReserved2      = NULL;
    info.lpReserved       = NULL;
    info.lpDesktop         = NULL;
    info.dwFlags           = STARTF_USEFILLATTRIBUTE;
    return;
}
int readVector(int dTime, char* revFile)
{
    int i;
    FILE* fprev;
    char rdata[4][16];
    Sleep(dTime);
    fprev = fopen(revFile, "r");
    if (fprev == NULL)
    {
        cout << "ERROR in fopen - fprev \n"
            << endl;    return (-1);
    }
    for (i = 0; i < 4; i++)
    {
        fscanf(fprev, "%s\n", rdata[i]);
        cout << rdata[i] << endl;
    }
    fclose(fprev);
    return (0);
}

```

Скомпилируйте программу и запустите ее на выполнение. Вы получите результаты, показанные на рис. 3.13, т.е. вектор, определяющий позицию точки.



```
MatlabMatrix
Auto
11.608
-10.868
3.779
1
Press any key to continue
```

Рис. 3.13. Результаты выполнения программы

Чтобы проверить выполнение операции над обратной матрицей, вам надо при вызове функции `dataSource()` изменить индекс операции с 0 на 1. Например, вы можете изменить в функции `main()` вызов `dataSource(0.78539, 0)` на `dataSource(0.78539, 1)`. Этим вы проверите преобразование вектора при переводе из системы координат UVW в систему XYZ.

При тестировании оказалось, что задержка 100 миллисекунд вполне достаточна для того, чтобы выполняемая программа, сформированная на базе M-функции, завершила операции над матрицей. При задержке меньше 50 миллисекунд иногда возникали ошибки, связанные с тем, что программа на MSVC не могла открыть файл `sMatrix.txt`. Эти ошибки возникали потому, что M-функция не успевала завершить операции с матрицами.

Исходные файлы данного примера, в том числе файлы заголовков и библиотек, находятся в каталоге `Chapter 3\MatlabMatrix` на прилагаемом к книге компакт-диске. Проект называется `MatlabMatrix`.

3.8. Интерфейс между Visual Basic и Matlab

3.8.1. Введение

Интерфейс между Visual Basic и Matlab реализуется путем создания исполняемого файла либо динамической библиотеки, разработанной в среде MSVC++. DLL выступает в роли моста между программой на Visual Basic и функциями Matlab. Детальная информация о создании DLL и обеспечении взаимодействия Visual Basic и VC++ была приведена в главе 2.

Visual Basic является удобным инструментом для создания графических пользовательских интерфейсов. Интерфейс самого Visual Basic прост для восприятия и удобен в работе. Однако программы, созданные в среде Visual Basic, практически не пригодны для управления процессами в реальном времени. Анализ данных с их помощью также осуществляется неэффективно. Взаимодействуя с программами, написанными на VC++, программы на Visual Basic приобретают возможность управлять процессами в реальном масштабе времени и обрабатывать информацию средствами Matlab.

В данном разделе рассматриваются два примера, иллюстрирующие совместное использование различных инструментов для решения одной задачи. В первом примере с помощью Matlab Compiler мы создадим независимую программу на базе М-функции, которая будет использоваться программой на Visual Basic для построения двух графиков. Этот пример относительно прост. Он приведен здесь лишь для демонстрации принципов доступа к функциям Matlab из программы на Visual Basic. Во втором примере применяется DLL, написанная на MSVC++. С помощью этой библиотеки осуществляется обращение к С-файлу, выполняющему действия по анализу данных. Этот С-файл сформирован Matlab Compiler на базе М-функции. Чтобы упростить пример, вместо реальных данных используются данные, сгенерированные путем вычисления значений математических функций. В приложениях, решающих практические задачи, применяются специальные средства сбора данных.

3.8.2. Обращение к исполняемому файлу из программы на Visual Basic

Пример, демонстрирующий обращение к исполняемому файлу из программы на Visual Basic, будет создаваться в такой последовательности.

- Сначала будет создана М-функция, выполняющая следующие действия.
 - Открытие файла с данными и чтение координат x , y и z , содержащихся в матрице размерностью 100×3 .
 - Построение зависимостей x - y , x - z на одном графике.
 - Построение зависимостей x - y , x - z на двух графиках.
- Затем М-функция будет преобразована в исполняемый файл `vbMPlot.exe`. Для этой цели используется Matlab Compiler.
- В последнюю очередь будет создан проект Visual Basic, обеспечивающий следующие функциональные возможности.
 - Формирование файла данных с именем `vbMdata.txt`.
 - Генерация данных, имитирующих синусоидальный и косинусоидальный сигналы, и запись их в файл `vbMdata.txt`. В этот же файл помещается вектор временных данных x .

- Вызов исполняемого файла `vbMPlot.exe`, осуществляющего построение графиков в среде Matlab. Исполняемый файл подготовлен с помощью Matlab Compiler.
- Возвращение управления программе на Visual Basic после завершения построения графиков.
- Выход из программы.

3.8.2.1. Разработка М-функции

Предположим, что файл данных был создан в среде MSVC и хранится в каталоге `C:\data` под именем `vbMdata.txt`. Откройте новый М-файл в рабочей области Matlab и введите код, представленный в листинге 3.19.

Листинг 3.19. М-функция `vbMPlot()`

```
% Файл:      vbMPlot.m
% Описание: М-функция, предназначенная для отображения
%           двух векторов на графике
% Дата:      4/9/2002
% Автор:     Y. Bai
function vbMPlot()

fid = fopen('C:\data\vbMdata.txt','r');
[T] = fscanf(fid,'%f ', [3, 100]);
S = T';

vec_Time = S(:, 1);
vec_Y = S(:, 2);
vec_Z = S(:, 3);

plot( vec_Time, vec_Y, 'y-', vec_Time, vec_Z, 'g-');
title('Two Vectors - Sinusoid and Cosine Simulated Waveforms');
xlabel('Time');
grid
legend('Sine', 'Cosine')
```

М-функция открывает файл данных посредством вызова `fopen()` и записывает информацию из файла в локальную переменную `T`. Заметьте, что информация хранится в матрице размерностью 100×3 , но при извлечении ее матрица преобразовывается в матрицу 3×100 . Так происходит потому, что при записи в файл в среде Visual Basic или Visual C++ происходит транспонирование матрицы.

Три локальные переменные, `vec_Time`, `vec_Y` и `vec_Z`, используются при извлечении векторов, которые хранятся в файле данных в виде столбцов матрицы. Функция `plot()` строит графики, соответствующие двум векторам: один из них отображается желтым цветом, а другой — зеленым.

Введите в рабочей области Matlab приведенную ниже команду. С ее помощью М-функция преобразуется в исполняемый файл `vbMPlot.exe`.

```
mcc -B sgl vbMPlot.m
```

Ниже приведены файлы, созданные в результате выполнения данной команды.

```
vbMPlot.c    vbMPlot.h    vbMPlot.exe
title.c     title.h
xlabel.c    xlabel.h
ylabel.c    ylabel.h
```

В данный момент нас интересует только файл `vbMPlot.exe`, который размещается в каталоге `C:\MATLAB6p1\work`. К этому файлу будет обращаться программа, написанная на Visual Basic.

3.8.2.2. Создание проекта Visual Basic

Загрузите Visual Basic 6.0 и создайте новый проект Standard EXE. Откройте окно формы и добавьте в него следующие интерфейсные элементы.

- Текстовая метка: Interface to Matlab M-Function (полужирный шрифт желтого цвета размером 18 пунктов, отображаемый на темно-сером фоне).
- Два поля редактирования.
 - Имя: `txtAmplitude`. Используется для ввода амплитуды синусоидального и косинусоидального сигналов.
 - Имя: `txtSize`. Позволяет задавать число отсчетов.
- Две текстовые метки.
 - Текст: `Data Amplitude`.
 - Текст: `Number of Data`.
- Три кнопки.
 - Имя: `cmdCreate`. Используется для создания файла данных, содержащего синусоидальный и косинусоидальный сигналы.
Надпись на кнопке: `Create Data`.
 - Имя: `cmdPlot`. Используется для вызова M-функции, осуществляющей построение графиков.
Надпись на кнопке: `Matlab Plot`.
 - Имя: `cmdExit`. Используется для завершения программы.
Надпись на кнопке: `Terminate`.

Сохраните форму в файле `vbMPlot.frm`, а данные проекта в файле `vbMPlot.vbp`. Готовая форма должна выглядеть приблизительно так, как показано на рис. 3.14.

3.8.2.3. Создание процедуры обработки события

- Код, соответствующий кнопке `Terminate`. Откройте форму, дважды щелкните на кнопке `Terminate`, чтобы получить доступ к процедуре обработки события, и введите команду `end`.
- Код для кнопки `Create Data`. Откройте процедуру обработки события, дважды щелкнув на кнопке `Create Data`, и введите код процедуры, показанный в листинге 3.20.

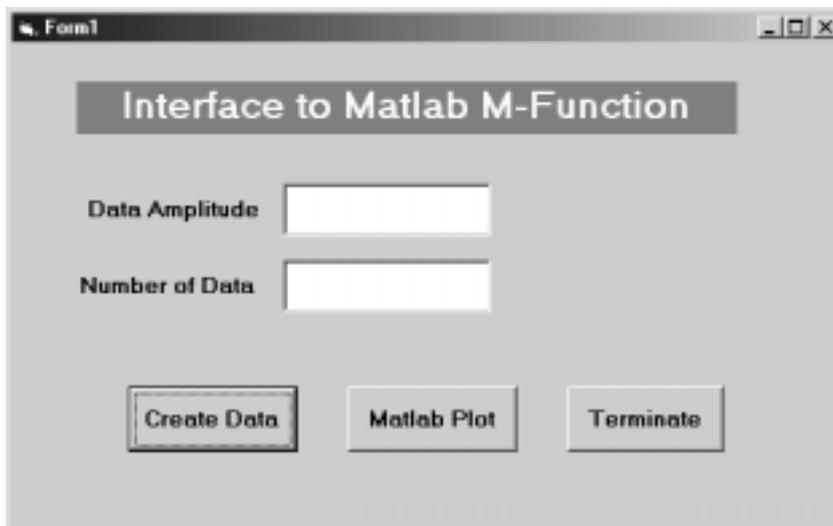


Рис. 3.14. Внешний вид формы vbMPlot.frm

Листинг 3.20. Обработка события, связанного с кнопкой Create Data

```

Private Sub cmdCreate_Click()

Dim i As Integer, amp As Integer, size As Integer, pi As Single
Dim sine(1 To 500) As String, cose(1 To 500) As String

pi = 3.14159
amp = Val(txtAmplitude.Text)
size = Val(txtSize.Text)

If amp = 0 Or size = 0 Then
    MsgBox "Please Enter the Amplitude and Number for the Wavefomes "
Else
    Open "C:\data\vbMdata.txt" For Output As #1

    For i = 1 To size
        sine(i) = Str(amp * Sin((2 * pi) / size * i))
        cose(i) = Str(amp * Cos((2 * pi) / size * i))
        sine(i) = Format(sine(i), "Fixed")
        cose(i) = Format(cose(i), "Fixed")
        Print #1, Str(i) & "    " & sine(i) & "    " & cose(i)
    Next i

    Close #1
End If

txtAmplitude.SetFocus

End Sub

```

После щелчка на кнопке Create Data генерируется событие, которое передается процедуре cmdCreate. В теле процедуры объявляются шесть локальных переменных.

- *i*. Счетчик цикла, используемого для генерации синусоидального и косинусоидального сигналов.
- *amp*. Амплитуда синусоидального и косинусоидального сигналов.
- *size*. Число отсчетов для синусоидального и косинусоидального сигналов.
- *pi*. Символьная константа, приближенно равная значению числа π .
- Символьные строки *sine* и *cosine*. Их максимальный размер составляет 500 элементов.

После объявления переменных три из них инициализируются. Переменной *pi* присваивается значение, равное 3.14159, а величины, записываемые в *amp* и *size*, задает пользователь. Если пользователь забудет ввести амплитуду или размер, отобразится окно с соответствующим сообщением.

После инициализации переменных открывается файл данных *C:\data\vbMdata.txt* и в цикле *for* вычисляются значения синусоидального и косинусоидального сигналов. Подготовленные данные записываются в открытый файл. Для представления информации в виде строки используются встроенные функции *Str()* и *Format()*. При форматировании указан тип *Fixed*; это означает, что данные должны содержать как минимум одну цифру перед десятичной точкой и две цифры после нее. После записи информации в файл вызывается функция *Close*.

Перед завершением процедуры фокус ввода передается полю редактирования, предназначенному для указания амплитуды. Это сделано для того, чтобы пользователю было удобно вводить очередное значение.

- Код для кнопки Matlab Plot. Когда пользователь щелкает на этой кнопке, Visual Basic вызывает независимое приложение *vbMPlot.exe*, созданное посредством Matlab Compiler. С этого момента управление передается Matlab, а М-функция выполняет действия, необходимые для построения графиков.

Введите в тело процедуры обработки события код

```
Private Sub cmdPlot_Click()  
    Shell "C:\MATLAB6p1\work\vbMPlot.exe", 1  
End Sub
```

Исполняемый файл находится в каталоге *C:\MATLAB6p1\work*. Для обращения к нему используется функция *Shell*. Формат вызова этой функции указан ниже.

```
Shell путь, стиль_окна
```

Первый параметр должен содержать полное имя файла, включающее как имя, так и путь. Стилль окна указывает, какой тип окна используется при запуске приложения. Допустимые значения этого параметра приведены в табл. 3.2. В данном примере указано значение 1, т.е. используется нормальное окно с поддержкой фокуса ввода.

Таблица 3.2. Значения параметра, определяющего стиль окна

Символьная константа	Значение	Тип окна
vbHide	0	Скрытое окно с поддержкой фокуса ввода
vbNormalFocus	1	Обычное окно с поддержкой фокуса ввода
vbMinimizedFocus	2	Окно представлено в виде пиктограммы. Фокус ввода поддерживается
vbMaximizedFocus	3	Окно максимальных размеров с поддержкой фокуса ввода
vbNormalNoFocus	4	Обычное окно без поддержки фокуса ввода
vbMinimizedNoFocus	6	Окно представлено в виде пиктограммы. Фокус ввода не поддерживается

- Код процедуры обработки события `Form_Load`. При выполнении программы данная процедура обработки вызывается первой. Поэтому в ней должны быть предусмотрены все действия по инициализации. Для рассматриваемого приложения нам необходимо лишь сделать кнопку `Create Data` кнопкой по умолчанию. Включите в тело процедуры обработки события следующий код:

```
Private Sub Form_Load()
    cmdCreate.Default = True
End Sub
```

И наконец, нам надо ввести в разделе `General Declaration` формы следующее выражение:

```
Option Explicit
```

Данное выражение позволяет избежать ошибок, связанных с некорректными именами переменных. Если при отсутствии данной строки в имени какой-то переменной будет допущена ошибка, Visual Basic примет ее за новую переменную. Выявить подобную ошибку при отладке крайне трудно.

На этом работа по созданию программы на Visual Basic оканчивается. Сохраните исходный текст и запустите приложение на выполнение. Введите амплитуду и размер для синусоидального и косинусоидального сигналов, щелкните на кнопке `Create Data`, а затем на кнопке `Matlab Plot`. На экране появится окно с графиками. Оно будет выглядеть приблизительно так, как показано на рис. 3.15.

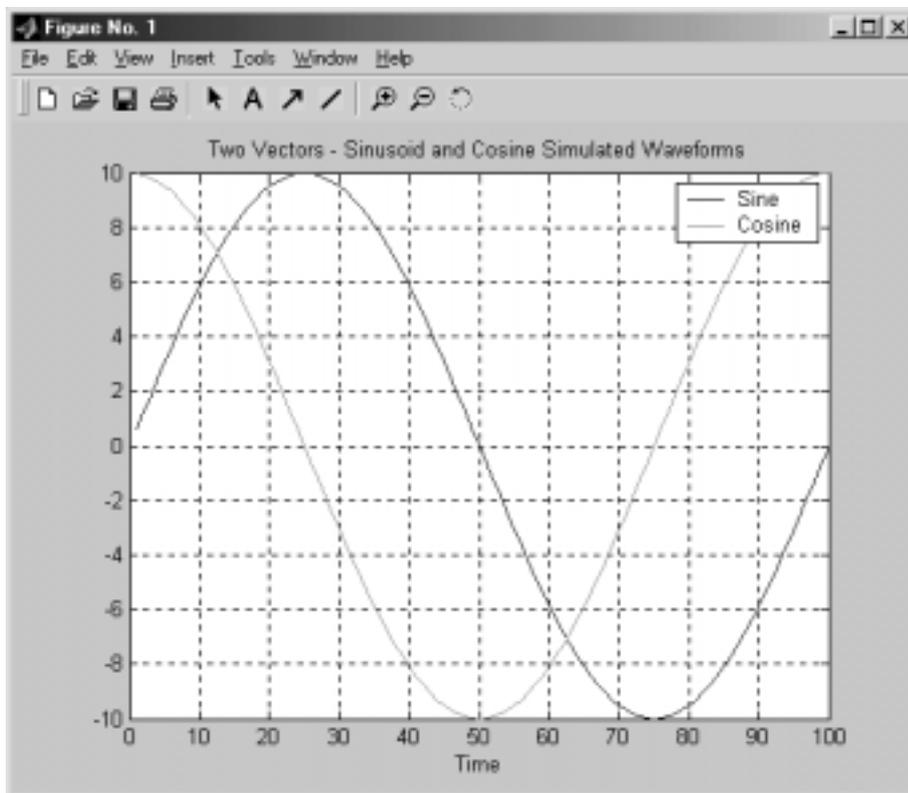


Рис. 3.15. Графики, сформированные в процессе работы программы

Полностью исходный код проекта Visual Basic vbMPlot показан в листинге 3.21. Чтобы составить полное представление о проекте, обратитесь к файлу vbMPlot.vbp, который находится в каталоге Chapter 3\vbMPlot на прилагаемом к книге компакт-диске.

Листинг 3.21. Полный код проекта vbMPlot на Visual Basic

```
Option Explicit

Private Sub cmdCreate_Click()

Dim i As Integer, amp As Integer, size As Integer, pi As Single
Dim sine(1 To 500) As String, cose(1 To 500) As String

pi = 3.14159
amp = Val(txtAmplitude.Text)
size = Val(txtSize.Text)

If amp = 0 Or size = 0 Then
```

```

    MsgBox "Please Enter the Amplitude and Number for the Waveforms "
Else
    Open "C:\data\vbMdata.txt" For Output As #1
    For i = 1 To size
        sine(i) = Str(amp * Sin((2 * pi) / size * i))
        cose(i) = Str(amp * Cos((2 * pi) / size * i))
        sine(i) = Format(sine(i), "Fixed")
        cose(i) = Format(cose(i), "Fixed")
        Print #1, Str(i) & "      " & sine(i) & "      " & cose(i)
    Next i
    Close #1
End If
txtAmplitude.SetFocus
End Sub

Private Sub Form_Load()
cmdCreate.Default = True
End Sub

Private Sub cmdPlot_Click()
Shell "C:\MATLAB6p1\work\vbMPlot.exe", 1
End Sub

Private Sub cmdExit_Click()
End
End Sub

```

3.8.3. Обращение к динамической библиотеке, созданной в среде MSVC

В примере, рассмотренном ранее, размерность матрицы была ограничена. В М-функции размерность была задана равной 100, и изменить ее из программы на Visual Basic было невозможно. Другими словами, программа на Visual Basic не контролировала М-функцию в среде Matlab. В рассматриваемом здесь примере мы создадим на MSVC динамическую библиотеку (DLL), к которой будем обращаться из программы на Visual Basic. При этом мы сможем управлять размером матрицы данных. При обращении к DLL программа на Visual Basic будет передавать сведения о размере в виде параметра, а М-функция будет передавать программе на Visual Basic данные, свидетельствующие о том, что информация о размере принята.

Для того чтобы упростить нашу задачу, воспользуемся М-функцией vbMPlot, созданной в предыдущем примере, слегка модифицировав ее. Во-первых, мы изменим имя файла на vbMdllPlot.m. Во-вторых, добавим параметр size, который будет задавать размер матрицы. Кроме того, функция будет возвращать значение rc. Модифицированный код М-функции показан в листинге 3.22.

Листинг 3.22. Код М-функции vbMdllPlot()

```

% Файл:      vbMdllPlot.m
% Описание:  М-функция Matlab, предназначенная для графического
%            представления двух векторов
% Дата:      4/13/2002
% Автор:     Y. Bai
function rc = vbMdllPlot(size)

fid = fopen('C:\data\vbMdlldata.txt','r ');
[T] = fscanf(fid,'%f ', [3, size]);
S = T';

vec_Time = S(:, 1);
vec_Y = S(:, 2);
vec_Z = S(:, 3);

plot( vec_Time, vec_Y, 'r-', vec_Time, vec_Z, 'b-' );
title('VB Call a DLL in MSVC - Sinusoid and Cosine Waveforms');
xlabel('Time');
grid
legend('Sine', 'Cosine')

rc = size;

```

Параметр `size` задает число строк в файле данных `vbMdlldata.txt`, а возвращаемое значение `rc` используется как подтверждение того, что М-функция успешно приняла параметр. Значение `rc` возвращается по завершении функции. Остальная часть кода идентична функции `vbMPlot`, использованной в предыдущем примере.

Для компиляции М-функции в С-код, пригодный для включения в разделяемую библиотеку, применяется следующая команда:

```
mcc -B sgl -t -W libhg:vbMdllPlotlib -T link:lib vbMdllPlot.m
```

В результате компиляции генерируются файлы, приведенные в табл. 3.3.

Таблица 3.3. Файлы, создаваемые при преобразовании М-функции в С-код

Файл заголовков	Исходные файлы	Файлы библиотек	Прочие файлы
<code>vbMdllPlot.h</code>	<code>vbMdllPlot.c</code>	<code>vbMdllPlotlib.mlib</code>	<code>vbMdllPlotlib.exports</code>
<code>vbMdllPlotlib.h</code>	<code>vbMdllPlotlib.c</code>		

В настоящий момент нас интересуют два исходных файла, `vbMdllPlot.c` и `vbMdllPlotlib.c`, а также два файла заголовка, `vbMdllPlot.h` и `vbMdllPlotlib.h`. Эти файлы мы добавим к проекту MSVC по созданию динамической библиотеки, к которой программа на Visual Basic будет обращаться для построения графика.

Создайте в среде MSVC++ 6.0 новый проект `vbMdllPlot` и выполните следующие действия.

- Запустите Microsoft Visual C++ 6.0 и откройте новый проект, выбрав в меню File пункт New. Щелкните на вкладке Projects и выберите пункт Win32 Dynamic-Link Library. В поле редактирования Project name: введите имя `vbMdllPlot`.
- Откройте с помощью Windows Explorer папку `C:\МАТЛАВ6p1\work`. Удерживая нажатой клавишу <Ctrl>, выберите в этой папке следующие файлы.

Файлы заголовков:

- `grid.h`
- `legend.h`
- `title.h`
- `vbMdllPlot.h`
- `vbMdllPlotlib.h`
- `xlabel.h`
- `ylabel.h`

Исходные файлы:

- `title.c`
- `xlabel.c`
- `ylabel.c`
- `vbMdllPlot.c`
- `vbMdllPlotlib.c`
- Скопируйте эти файлы в папку проекта `C:\vbMdllPlot`. При желании вы можете создать для проекта другой каталог.
- Продолжая работу с Windows Explorer, откройте каталог `C:\МАТЛАВ6p1\extern\lib\win32\Microsoft\msvc60`. Удерживая нажатой клавишу <Ctrl>, выберите в нем следующие библиотечные файлы.
 - `libmx.lib`
 - `libmmfile.lib`
 - `libmatlb.lib`
 - `libmwsglm.lib`
 - `sgl.lib`
- Скопируйте выбранные файлы в каталог `C:\vbMdllPlot`. Если вы создали для проекта другой каталог, файлы надо поместить в него.
- Закройте Windows Explorer. Создайте в рабочей области Visual C++ 6.0 новый исходный файл на языке C. Для этого выполните следующие действия.

- Щелкните на пункте меню Project⇒Add To Project⇒New и в появившемся диалоговом окне выберите C++ Source File. Задайте в поле File name: имя файла vbMdllPlotWrap и щелкните на кнопке ОК.
- Выберите пункт меню Project⇒Add To Project⇒Files и в отобразившемся диалоговом окне выберите вновь созданный исходный файл vbMdllPlotWrap.cpp. Щелкните на этом файле правой кнопкой мыши и в контекстном меню выберите пункт Rename. Измените расширение файла с .cpp на .c.
- Введите исходный код, показанный в листинге 3.23.

Листинг 3.23. Содержимое файла vbMdllPlotWrap.c

```

/*****
* Файл:      vbMdllPlotWrap.c
* Описание:  файл-оболочка для vbMdllPlot. Используется
*            при вызове M-функции
* Дата:      4/13/2002
* Автор:     Y. Bai
*****/
#include "vbMdllPlot.h"
#include "vbMdllPlotlib.h"
#include "matlab.h"

double __stdcall vbMdllPlotWrap(int x)
{
    mxArray *x_ptr;
    mxArray *y_ptr;
    double* y;
    double ret;

    x_ptr = mlfScalar(x);
    vbMdllPlotlibInitialize();
    y_ptr = mlfVbMdllPlot(x_ptr);

    y = mxGetPr(y_ptr);
    ret = *y;

    return (ret);
}

```

Данная функция предназначена для связи DLL, сформированной с помощью Matlab Compiler, с DLL, разработанной в среде Visual C+. Этой библиотечной функции передается один параметр, а по окончании работы она возвращает результирующее значение. Параметр предоставляет программа на Visual Basic. Ей же возвращается значение функции. Ключевое слово `stdcall` представляет стандартный формат вызова DLL в программах на C/C++.

В начале выполнения функция создает локальные переменные типа `mxArray`, обеспечивающего совместимость с Matlab. Затем производится обращение к функции `vbMdllPlotlibInitialize()`. Эта функция определена в исходном

файле `vbMdlPlotlib.c`, созданном Matlab Compiler, и предназначена для инициализации Matlab DLL. Далее в теле функции вызывается еще одна интерфейсная функция, `mlfVbMdlPlot()`, которая определена в исходном файле `vbMdlPlot.c`. Эта функция организует обращение к Matlab DLL для построения синусоиды и косинусоиды. Библиотеке передается параметр `x_ptr`, преобразованный из скалярного значения в указатель посредством функции `mlfScalar()`, а возвращаемое значение присваивается локальному указателю `y`. Значение, на которое ссылается указатель, возвращается программе на Visual Basic.

Прототипы всех функций Matlab DLL хранятся в исходных файлах `vbMdlPlot.c` и `vbMdlPlotlib.c`. Обращение к DLL производится лишь посредством интерфейсных функций, предоставляемых Matlab Compiler.

В рабочей области Visual C++ щелкните на пункте меню `Project`⇒`Add To Project`⇒`Files`, чтобы отобразилось диалоговое окно. Убедитесь, что в поле `Look in:` выбран каталог текущего проекта `C:\vbMdlPlot`. Выделите все файлы (файлы заголовков, исходные файлы и файлы библиотек), скопированные из каталогов `C:\MATLAB6p1\work` и `C:\MATLAB6p1\extern\lib\win32\microsoft\msvc60`, нажав и удерживая клавишу `<Ctrl>`. Щелкните на кнопке `OK`, чтобы добавить эти файлы к проекту.

По окончании работы с диалоговым окном оно должно выглядеть так, как показано на рис. 3.16. Заметьте, что в качестве типа файла в поле `Files of type:` надо выбрать значение `All Files (*.*)`, в противном случае не все файлы будут отображаться в окне.

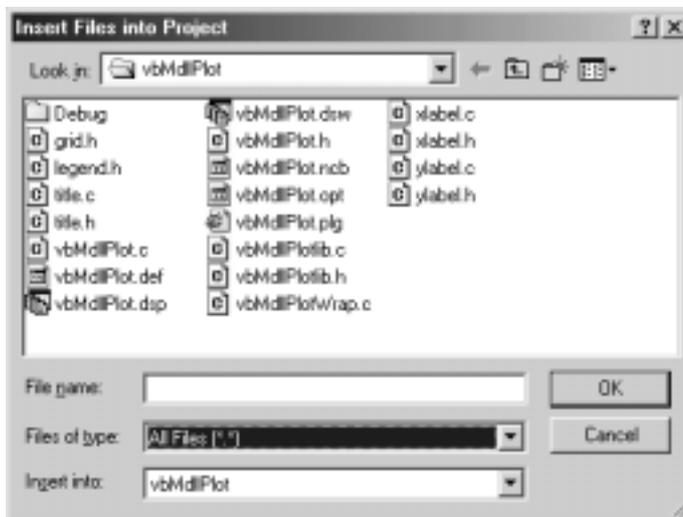


Рис. 3.16. Диалоговое окно Insert Files into Project

Некоторые читатели, наверное, заметили, что в списке файлов, принадлежащих проекту, отсутствуют некоторые библиотечные файлы. Как вы помните, мы добавили пять файлов из каталога `C:\MATLAB6p1\extern\lib\win32\microsoft\msvc60`; именно они отсутствуют в списке. Причина в том, что при использовании пункта меню `MSVC Add To Project` вы лишь устанавливали связь между проектом и файлами, хранящимися в некотором каталоге. Файлы, добавленные к проекту, находятся в том каталоге, в котором они располагались изначально. Поэтому ваш проект зависит от ряда файлов. Подобный проект называется зависимым проектом. Хорошим стилем считается создание независимых проектов, которые содержат все необходимые компоненты и для работы которых не нужны дополнительные файлы, расположенные в других позициях файловой системы. Для создания независимого проекта надо скопировать в его папку все необходимые файлы и установить новые пути к библиотекам. Подробно действия по созданию независимого проекта описаны в разделе 3.8.4.

- Создайте файл определений для DLL. Щелкните на пункте меню `File⇒New` и в появившемся диалоговом окне выберите пункт `Text file`. Введите в поле редактирования `File name`: имя `vbMdllPlot.def` и щелкните на кнопке `OK`, чтобы открыть новый DEF-файл. Введите в файле следующий код:

```
LIBRARY vbMdllPlot.dll
EXPORTS
vbMdllPlotWrap
```

- Щелкните на пункте меню `Tools⇒Options` и в появившемся диалоговом окне выберите вкладку `Directories`. Дважды щелкните на последней (пустой) строке в списке `Directories` и перейдите к каталогу `C:\MATLAB6p1\extern\include`. Щелкните на этом пункте, чтобы добавить соответствующий путь к проекту.
- Продолжая работу с диалоговым окном, выберите в раскрывающемся списке `Show Directories for` пункт `Library files` и найдите каталог `C:\MATLAB6p1\extern\lib\win32\microsoft\msvc60`. Дважды щелкните на этом каталоге, чтобы добавить к проекту путь к библиотеке.
- Создайте проект, активизировав пункт меню `Build⇒Build vbMdllPlot.dll`.
- В процессе построения DLL генерируются файлы `vbMdllPlot.lib` и `vbMdllPlot.dll`, которые размещаются в каталоге `C:\vbMdllPlot\Debug`. Они видны в списке файлов, отображаемом на рис. 3.17.
- Скопируйте файл `vbMdllPlot.dll` в один из каталогов, в которых система осуществляет поиск. В данном случае это каталог `C:\Matlabdll`. При желании вы можете поместить этот файл в каталог `C:\WINDOWS\SYSTEM`, предназначенный для хранения системных DLL. При запуске программы операционная система автоматически обнаружит разделяемую библиотеку.

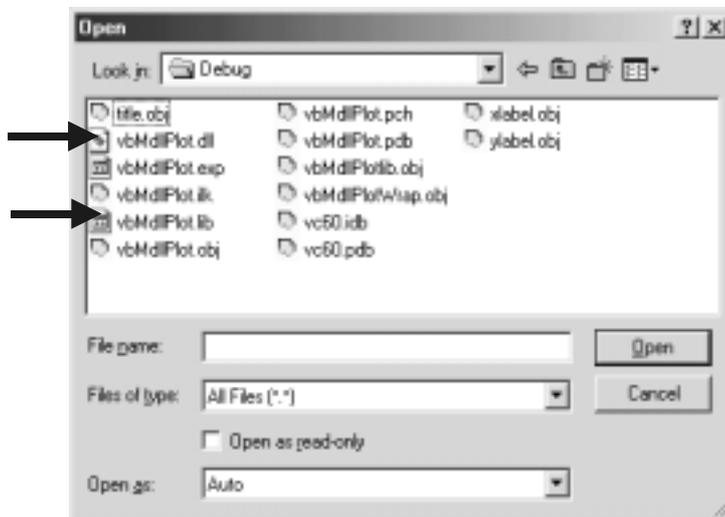


Рис. 3.17. Список файлов, содержащий `vbMdlPlot.lib` и `vbMdlPlot.dll`

Теперь нам надо создать программу на Visual Basic, которая будет обращаться к DLL. Данная программа будет поддерживать пользовательский интерфейс. С ее помощью пользователи смогут вводить данные, а вызываемая функция DLL осуществит построение графиков в среде Matlab.

Загрузите Visual Basic 6.0 и создайте новый проект Standard EXE. При желании вы можете скопировать проект `vbMPlot` и видоизменить его. В новом проекте надо добавить в форму три кнопки (рис. 3.18).

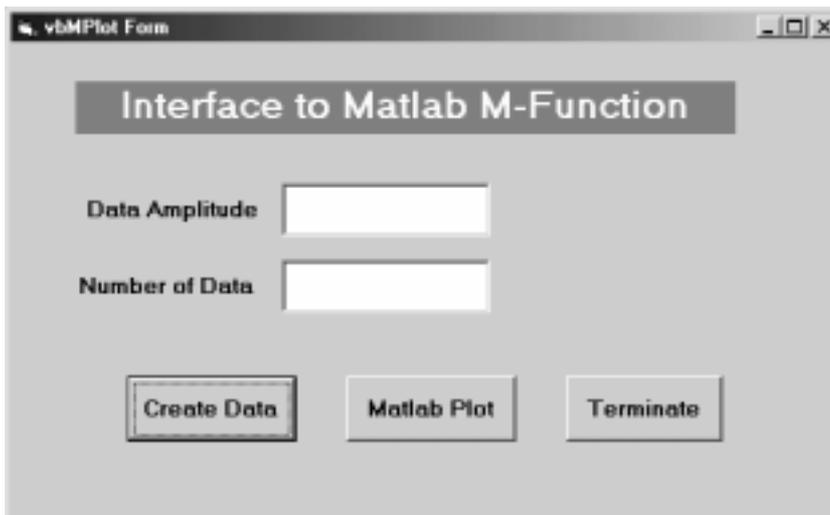


Рис. 3.18. Внешний вид формы, создаваемой в рамках проекта

Завершите создание кода проекта, выполнив следующие действия.

- Объявите переменные уровня формы и прототип функции DLL. Для этого надо открыть окно с кодом, выбрать в списке Object раздел General Declaration и ввести в этом разделе код, показанный в листинге 3.24.

Листинг 3.24. Объявление переменных уровня формы и прототипа функции DLL

```
Option Explicit

Dim size As Integer
Dim ret As Single

Private Declare Function vbMdllPlotWrap Lib
C:\WINDOWS\SYSTEM\vbMdllPlot.dll " _
    (ByVal x As Integer) As Double
```

Как видно в листинге, здесь создаются две переменные. Целочисленная переменная *size* передается М-функции и задает размер матрицы, а переменная *ret* хранит данные, возвращаемые М-функцией.

Объявление функции DLL представляет собой стандартный прототип, созданный по правилам Visual Basic. DLL расположена в каталоге C:\WINDOWS\SYSTEM, перед началом работы надо поместить ее туда. Функции DLL передается один параметр *x*, а возвращает она значение типа *Double*.

Код обработчика события, связанного с кнопкой Create Data, достаточно прост. Имя данной кнопки *cmdCreate*. Дважды щелкните на ней, чтобы открыть процедуру обработки события, и добавьте код, показанный в листинге 3.25.

Листинг 3.25. Обработчик события, соответствующего кнопке Create Data

```
Private Sub cmdCreate_Click()

Dim i As Integer, amp As Integer, pi As Single
Dim sine(1 To 1000) As String, cose(1 To 1000) As String

pi = 3.14159
amp = Val(txtAmplitude.Text)
size = Val(txtSize.Text)

If amp = 0 Or size = 0 Then
    MsgBox "Please Enter the Amplitude and Number for the Waveforms "
Else
    Open "C:\data\vbMdlldata.txt" For Output As #1
    For i = 1 To size
        sine(i) = Str(amp * Sin((2 * pi) / size * i))
        cose(i) = Str(amp * Cos((2 * pi) / size * i))
        sine(i) = Format(sine(i), "Fixed")
        cose(i) = Format(cose(i), "Fixed")
        Print #1, Str(i) & "      " & sine(i) & "      " & cose(i)
```

```

    Next i
    Close #1
End If

txtAmplitude.SetFocus

End Sub

```

Два строковых массива, `sine` и `cose`, используются для хранения данных, составляющих синусоидальный и косинусоидальный сигналы. Каждый массив может хранить до 1000 отсчетов. Два поля редактирования, `txtAmplitude` и `txtSize`, предназначены для ввода амплитуды и числа отсчетов. В цикле `for` создаются данные для двух графиков, которые записываются в файл `vbmdlldata.txt`. М-функция открывает этот файл, считывает из него информацию и использует ее для построения графиков в среде Matlab.

Перед выходом из процедуры фокус ввода передается полю редактирования `txtAmplitude`, чтобы подготовить его для ввода следующего значения.

- Кнопка `Matlab Plot` имеет имя `cmdPlot`. После щелчка на данной кнопке вызывается созданная ранее функция DLL, ей в качестве параметра передается число отсчетов. Эти сведения получает М-функция `vbMdllPlot.m`, которая строит графики в среде Matlab. Дважды щелкните на этой кнопке, чтобы открыть процедуру обработки события, и добавьте код, показанный в листинге 3.26.

Данный код очень прост. Он состоит лишь из обращения к функции DLL `vbMdllPlotWrap`, которой в качестве параметра передается значение `size`. Вызываемая функция DLL возвращает параметр `ret`, отображаемый в окне с сообщением. Это дает возможность пользователю убедиться, что функция DLL была выполнена успешно.

Листинг 3.26. Процедура обработки события, связанного с кнопкой `Matlab Plot`

```

Private Sub cmdPlot_Click()
ret = vbMdllPlotWrap(size)
MsgBox ret

End Sub

```

- Обработчик `Form_Load` и процедура поддержки кнопки `Terminate` также очень просты. Их коды показаны в листинге 3.27.

При запуске программы нам надо сделать кнопку `Create Data` кнопкой по умолчанию, поэтому свойству `Default` этого элемента присваивается значение `True`. Обработчик события, соответствующего активизации кнопки `Terminate`, состоит из выражения `End`, которое завершает программу.

Листинг 3.27. Процедура Form_Load() и обработчик события, связанного с кнопкой Terminate

```
Private Sub Form_Load()  
cmdCreate.Default = True  
End Sub  
  
Private Sub cmdExit_Click()  
End  
End Sub
```

Сохраните форму и проект в файлах vbMPlotdll.frm и vbMPlotdll.vbp. На этом работа над проектом заканчивается. Запустите приложение на выполнение и введите данные, как показано на рис. 3.19. Щелкните на кнопке Matlab Plot, чтобы вызвать функцию DLL, осуществляющую построение графиков. Результаты работы функции показаны на рис. 3.20.

Поверх окна с графиками отображается окно с сообщением, в котором выводится значение 500, возвращенное функцией DLL. Результат корректен, поскольку мы задавали число отсчетов, равное 500. М-функция отображает 500 точек на графике и возвращает это значение программе на Visual Basic. Таким образом, мы передали параметр М-функции и получили в ответ правильное значение. Щелкните на кнопке ОК, расположенной в окне с сообщением. Амплитуда кривых равна 10; именно это значение мы и задавали в поле редактирования.

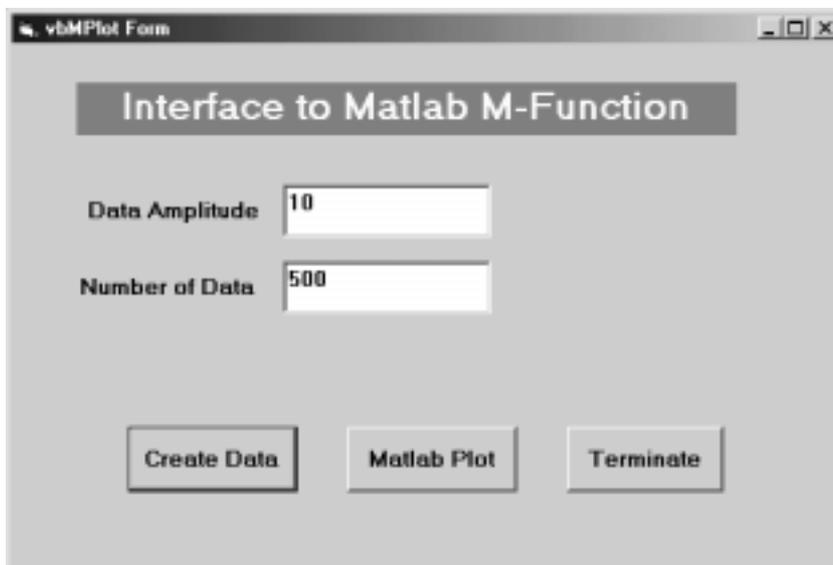


Рис. 3.19. Ввод данных в окне приложения

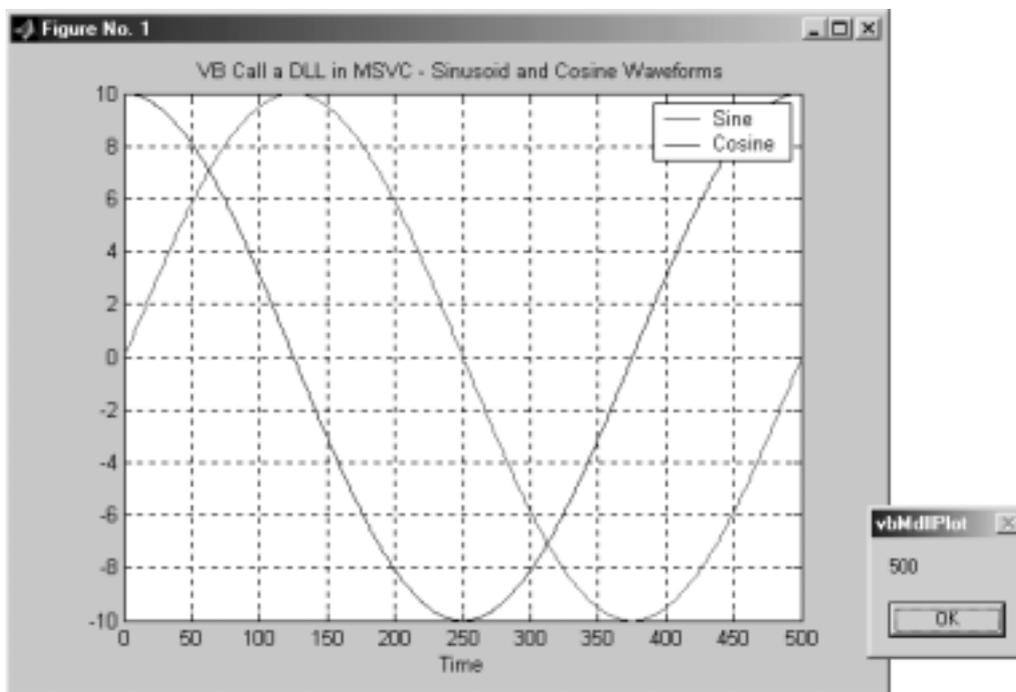


Рис. 3.20. Построение графиков с помощью функции DLL

Обратите внимание на одну особенность данной программы. Если вы снова щелкнете на кнопке `Matlab Plot`, не активизировав перед этим кнопку `Create Data`, в окне с сообщением по-прежнему будет отображаться значение 500. Другими словами, `M`-функция использует данные, переданные ей ранее. Если вы введете в полях редактирования новые данные, вам надо щелкнуть сначала на кнопке `Create Data` и лишь затем на кнопке `Matlab Plot`. Только в этом случае введенные значения будут учтены при построении графиков.

Файлы данного проекта хранятся на прилагаемом к книге компакт-диске в каталогах `Chapter 3\vbMPlotdll` и `Chapter 3\vbMdlPlot`. В этих каталогах содержатся следующие папки.

- `vbMdlPlot`. Динамическая библиотека, созданная в среде MSVC: исходные файлы, файлы заголовков и библиотечные файлы.
- `vbMPlotdll`. Программа на Visual Basic, обращающаяся к динамической библиотеке, созданной средствами MSVC. В этой папке находятся исходный код `M`-функции `vbMdlPlot.m`, библиотека `vbMdlPlot.dll`, созданная посредством Matlab Compiler, и исходные коды на Visual Basic. Здесь же расположен исполняемый файл Visual Basic, `vbMPlotdll.exe`. Исполняемый файл можно непосредственно запускать, не устанавливая предварительно средства Visual Basic.

3.8.4. Создание независимого проекта

Как было сказано в предыдущем разделе, проект `vbMdl1Plot` является зависимым, т.е. некоторые файлы находятся в других разделах файловой системы, и, для того чтобы компиляция была успешной, компилятор должен обнаружить их. При переносе такой программы на другой компьютер должны быть учтены все каталоги, в которых располагаются необходимые файлы. Это очень неудобно; если вы забудете какой-либо из файлов, выполнить компиляцию не удастся.

Избежать подобных трудностей можно, создав независимый проект. При переносе такого проекта на другой компьютер все требуемые файлы перемещаются автоматически. Для того чтобы создать независимый проект, надо собрать все файлы в рабочей области.

Модифицируем проект `vbMdl1Plot` так, чтобы он стал независимым. В первую очередь надо создать в среде MSVC новый проект `indep_vbMdl1Plot`, указав для него тип Win32 Dynamic-Link Library.

- Скопируйте с помощью Windows Explorer файлы из каталога проекта `vbMdl1Plot` во вновь созданный каталог `indep_vbMdl1Plot`. Перечень этих файлов приведен ниже.
 - `grid.h`
 - `legend.h`
 - `title.h`
 - `vbMdl1Plot.h`
 - `vbMdl1Plotlib.h`
 - `xlabel.h`
 - `ylabel.h`
 - `title.c`
 - `xlabel.c`
 - `ylabel.c`
 - `vbMdl1Plot.c`
 - `vbMdl1Plotlib.c`
 - `vbMdl1Plot.def`
 - `vbMdl1PlotWrap.c`

Все указанные файлы будут использоваться при создании независимого проекта.

- Откройте Windows Explorer и найдите папку `C:\MATLAB6p1\extern\lib\win32\microsoft\msvc60`. Скопируйте из нее в папку проекта `indep_vbMdl1Plot` следующие файлы.
 - `libmx.lib`
 - `libmmfile.lib`
 - `libmatlb.lib`
 - `libmwsglm.lib`
 - `sgl.lib`

Обратитесь к каталогу `C:\MATLAB6p1\extern\include` и скопируйте в папку проекта `indep_vbMdl1Plot` следующие файлы.

- `libmatlb.h`
- `libmatlbm.h`
- `libmmfile.h`

- libmwsglm.h
- libsgl.h
- matlab.h
- matrix.h
- mex.h

Закройте Windows Explorer и перейдите в рабочую область MSVC. Выберите пункт меню Project⇒Add To Project⇒Files, в появившемся диалоговом окне выделите скопированные ранее файлы и добавьте их к проекту.

- Откройте Windows Explorer и перейдите в папку нового проекта, indep_vbMdl1Plot. Выберите файл vbMdl1PlotWrap.c, щелкните на нем правой кнопкой мыши и, используя пункт Rename контекстного меню, переименуйте его в indep_vbMdl1PlotWrap.c. Откройте этот файл и внесите в него следующие изменения:

```
double __stdcall indep_vbMdl1PlotWrap(int x)
```

Как видите, здесь перед именем vbMdl1PlotWrap добавлена последовательность символов indep_. Этим формируется новое имя функции indep_vbMdl1PlotWrap.

- Откройте файл vbMdl1Plot.def и измените его код так, чтобы он выглядел следующим образом:

```
LIBRARY indep_vbMdl1Plot.dll
EXPORTS
indep_vbMdl1PlotWrap
```

Здесь также перед именем DLL и именем функции добавляется последовательность символов indep_.

Этот этап очень важен, так как файл DEF определяет DLL и функцию. Если вы не внесете в него изменения, то при выполнении программы система не сможет найти требуемую функцию.

- Щелкните на пункте меню indep_vbMdl1Plot.dll, чтобы создать новый проект.
- Чтобы открыть диалоговое окно Open, щелкните на пункте меню File⇒Open. Удостоверьтесь, что ваш новый проект indep_vbMdl1Plot отображается в поле редактирования Look in:, и дважды щелкните на папке Debug в списке файлов. Щелкните правой кнопкой мыши на файле indep_vbMdl1Plot.dll и выберите пункт Copy, чтобы скопировать данный библиотечный файл в системную папку C:\WINDOWS\SYSTEM.

На этом этапе мы завершили разработку независимого проекта indep_vbMdl1Plot.dll. Для того чтобы проверить его, обратитесь к рабочей области Visual Basic и откройте проект vbMPlotdll. Внесите в него описанные ниже изменения.

- Откройте окно с кодом и измените объявление функции DLL в разделе General Declaration следующим образом:

```
Private Declare Function indep_vbMdllPlotWrap Lib _  
    "C:\WINDOWS\SYSTEM\indep_vbMdllPlot.dll" _  
    (ByVal x As Integer) As Double
```

В данном случае изменения также ограничились добавлением indep. Заметьте, что данная последовательность символов должна быть добавлена дважды: один раз перед именем функции, а второй раз — в кавычках перед именем DLL.

- Откройте окно формы, щелкнув на кнопке View Object, и дважды щелкните на кнопке Matlab Plot, чтобы открыть соответствующую процедуру обработки события. Внесите изменения, аналогичные предыдущим, т.е. включите indep_ перед именем функции:

```
ret = indep_vbMdllPlotWrap(size)
```

На этом работа над проектом окончена. Теперь вы можете запустить программу на Visual Basic, чтобы проверить работоспособность созданного независимого проекта. На экране должны отобразиться те же графики, что и в предыдущем примере.

Если вы захотите перенести проект на другой компьютер, все необходимые файлы будут скопированы автоматически.

Файлы, принадлежащие данному независимому проекту, расположены на прилагаемом компакт-диске в каталоге Chapter 3\indep_vbMdllPlot. Обратившись к содержимому каталога, вы получите полный код приложения.

3.8.5. Использование функций MatrixVB для организации взаимодействия Matlab и Visual Basic

MatrixVB представляет собой COM-библиотеку, содержащую набор функций Matlab. Этот набор расширяет возможности Visual Basic и позволяет пользователям обращаться к математическим функциям в среде Matlab. Благодаря наличию COM-библиотеки доступ к функциям Matlab из программ на Visual Basic осуществляется относительно просто. После включения библиотеки в среду Visual Basic оператору Visual Basic ставится в соответствие оператор в среде Matlab. Информацию об отображении средств Visual Basic в Matlab можно найти в описании функций MatrixVB или в интерактивной справочной системе.

3.8.5.1. Инсталляция MatrixVB

Инсталлировать MatrixVB несложно. Достаточно лишь выполнить описанные ниже действия.

- Скопируйте Matrix VB 4.52 с Web-узла Matlab или воспользуйтесь копией на компакт-диске, поставляемом разработчиками.

- Загрузите Visual Basic и откройте проект Standard EXE с именем test.
- Щелкните на пункте меню Project⇒References, чтобы открыть диалоговое окно References.
- В списке Available References найдите пункт MMatrix, установите флажок рядом с этим пунктом и щелкните на кнопке ОК, чтобы добавить к проекту на Visual Basic библиотеку MatrixVB. В результате ваших действий диалоговое окно References должно выглядеть так, как показано на рис. 3.21.

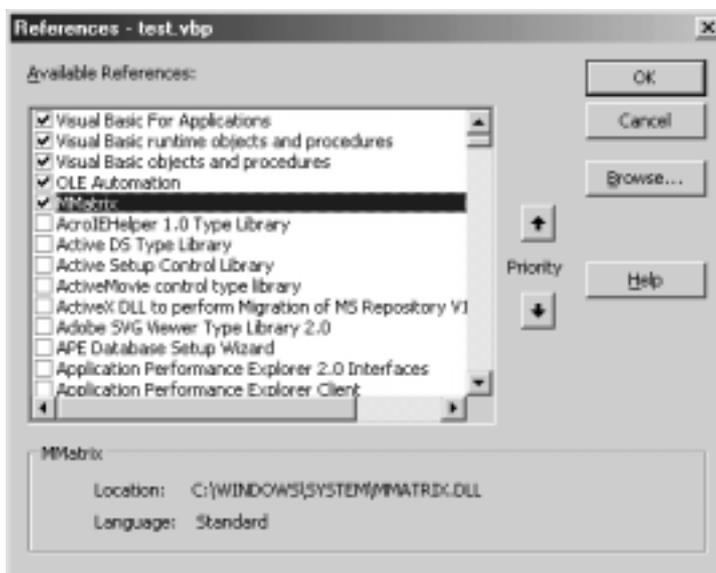


Рис. 3.21. Диалоговое окно References

В отличие от многих других библиотек MatrixVB не представляется пиктограммой в рабочей области Visual Basic. Для того чтобы отобразить промежуточные результаты вызова функций MatrixVB, надо использовать такие функции, как show или plot.

Чтобы проверить функционирование MatrixVB, включите в форму кнопку cmdOK и введите в тело процедуры обработки события код, показанный в листинге 3.28. В данной процедуре создается матрица размером 2×3 и в нее записываются три значения: 6, -3 и 4. Остальные элементы матрицы равны нулю. Функция mabs() вычисляет абсолютное значение элементов матрицы A. Функция Show отображает результаты вычислений в окне MatrixVB.

Запустите программу на выполнение и щелкните на кнопке ОК, расположенной в составе формы. На экране появится окно MatrixVB, показанное на рис. 3.22. Как видно на рисунке, отрицательное значение элемента A(1, 2) преобразовано в положительное число 3. Библиотека MatrixVB работает корректно.

Листинг 3.28. Обработчик события для кнопки cmdOK

```
Private Sub cmdOK_Click()
Dim A(1 To 2, 1 To 3) As Double

A(1, 1) = 6
A(1, 2) = -3
A(2, 3) = 4
B = mabs(A)
B.Show

End Sub
```

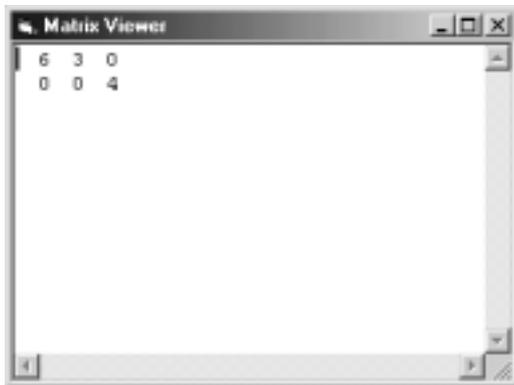


Рис. 3.22. Результаты использования MatrixVB

3.8.5.2. Создание анализирующей программы с помощью MatrixVB

Напишем программу, которая будет анализировать последовательность псевдослучайных значений. Исключите из формы кнопку cmdOK. Сохраните форму и проект test под именем Matrix.

Формирование пользовательского интерфейса

Увеличьте форму и добавьте в нее следующие кнопки.

Имя	Текст
cmdCreate	CREATE
cmdAnalysis	ANALYSIS
cmdExit	EXIT

Добавьте в форму два поля редактирования и две текстовые метки.

Имя	Назначение
txtSize	Ввод числа отсчетов
txtAmp	Ввод амплитуды сигнала

Имя	Текст
Label1	Data Points
Label2	Amplitude

Включите в форму два элемента PictureBox.

Имя	Назначение
picRandom	Отображение псевдослучайного сигнала
picAnaly	Отображение свертки сигнала

Щелкните на форме, перейдите к окну Properties и установите с помощью свойства BackColor желтый цвет фона. По окончании работы над пользовательским интерфейсом он должен выглядеть так, как показано на рис. 3.23.

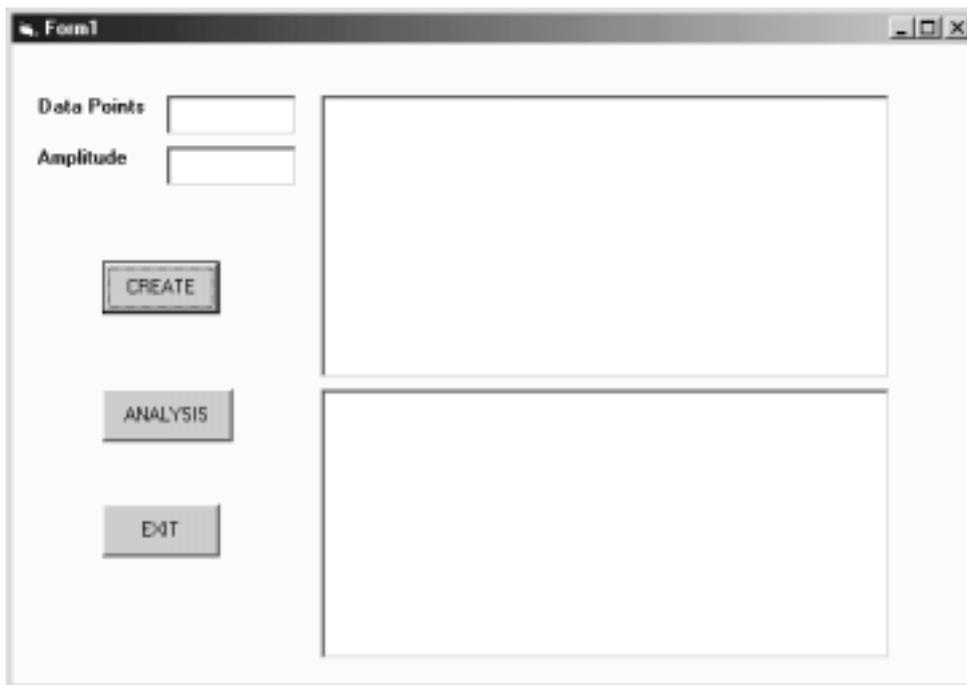


Рис. 3.23. Внешний вид окна с пользовательским интерфейсом

Создание кода программы

Откройте окно с кодом. В первую очередь нам надо объявить переменные уровня формы. Щелкните на списке Object и выберите пункт General, открыв тем самым раздел General Declaration. Введите в этом разделе код, показанный в листинге 3.29. Ниже описано назначение объявленных здесь переменных.

Листинг 3.29. Код, включаемый в раздел General Declaration

```
Option Explicit
```

```
Dim intSize As Integer, intAmp As Integer
Dim mRand As Variant, cRand As Variant
Dim ax As Variant, z As Variant
```

- `intSize`. Число точек, вводимых пользователем в процессе работы программы.
- `intAmp`. Амплитуда псевдослучайного сигнала. Вводится пользователем при работе программы.
- `mRand`. Массив данных, используемый для хранения псевдослучайного сигнала.
- `cRand`. Массив данных, используемый для хранения свертки.
- `ax`. Массив, используемый для хранения координат в рабочей области Visual Basic.
- `z`. Массив, применяемый для хранения данных после умножения.

Откройте окно формы и дважды щелкните на кнопке CREATE, чтобы получить доступ к соответствующей процедуре обработки события. Включите в тело процедуры код, показанный в листинге 3.30.

Листинг 3.30. Процедура обработки события, соответствующего щелчку на кнопке CREATE

```
Private Sub cmdCreate_Click()

If Val(txtSize.Text) = 0 Or Val(txtAmp.Text) = 0 Then
    MsgBox "Enter the Values for Data Points and Amplitude "
    Exit Sub
End If
intSize = Val(txtSize.Text)
intAmp = Val(txtAmp.Text)
mRand = randn(1, intSize)
z = times(intAmp, mRand)
ax = vbaxes(picRandom.hWnd)
plot (z)
grid ("on")
Title ("Normal Distributed Random Signal")
xlabel ("Data Points")

End Sub
```

Вначале данная процедура проверяет корректность значений, введенных пользователем в полях редактирования Data Points и Amplitude. Если значения введены некорректно, отображается окно с сообщением.

После проверки процедура преобразует текстовые строки, введенные в полях `Data Points` и `Amplitude`, в числовые значения и присваивает эти значения переменным `intSize` и `intAmp`. Функция `randn()` создает набор псевдослучайных значений с нормальным распределением. Число значений в наборе равно `intSize`. Функция `times()` умножает псевдослучайные значения на величину `intAmp`. Это позволяет получить требуемую амплитуду сигнала.

Функция `vbaxes()` вызывается для того, чтобы получить вектор осей, а функция `plot()` отображает псевдослучайный сигнал в виде графика. Для того чтобы улучшить внешний вид графика, при его построении используются дополнительные функции `grid`, `title` и `xlabel`.

Дважды щелкните на кнопке `ANALYSIS`, чтобы открыть процедуру обработки события, и добавьте код, показанный в листинге 3.31.

Листинг 3.31. Обработчик события, соответствующего щелчку на кнопке ANALYSIS

```
Private Sub cmdAnalysis_Click()

If Val(txtSize.Text) = 0 Or Val(txtAmp.Text) = 0 Then
    MsgBox "Enter the Values for Data Points and Amplitude "
    Exit Sub
End If

cRand = conv(mRand, mRand)
ax = vbaxes(picAnaly.hWnd)

plot (cRand)
grid ("on")
Title ("Convolution of Distributed Random Signal")
xlabel ("Data Points")

End Sub
```

В начале процедуры осуществляется проверка на допустимость значений, введенных пользователем в полях редактирования. Функция `conv()` вычисляет свертку псевдослучайного сигнала. Функция `vbaxes()` вызывается для того, чтобы получить вектор осей, а функция `plot()` отображает свертку псевдослучайного сигнала в виде графика.

Дважды щелкните на кнопке `EXIT`, открыв тем самым процедуру обработки события, и введите код, показанный в листинге 3.32. Код очень прост; он состоит только из команды `End`, которая завершает программу.

Листинг 3.32. Процедура обработки события для кнопки EXIT

```
Private Sub cmdExit_Click()
End
End Sub
```

На этом работа над проектом окончена. Сохраните созданные файлы и запустите программу на выполнение. Введите в полях редактирования число значений и амплитуду, например 200 и 10. Щелкните на кнопке CREATE, чтобы вывести график псевдослучайного сигнала, а затем активизируйте кнопку ANALYSIS, чтобы отобразить свертку. Окно работающей программы должно выглядеть так, как показано на рис. 3.24. Щелкните на кнопке Exit, чтобы завершить работу.

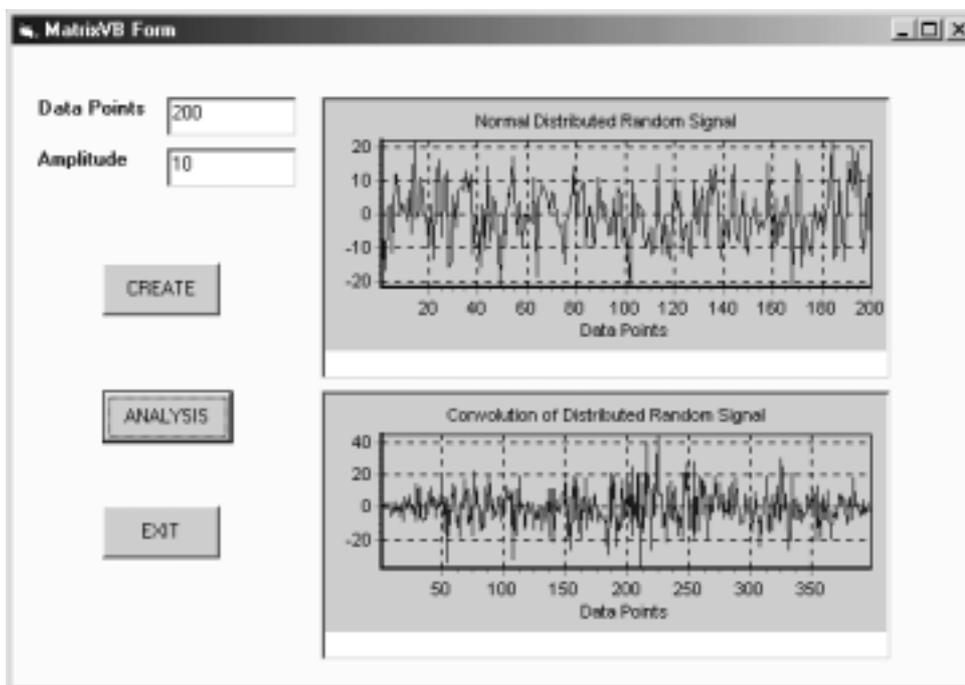


Рис. 3.24. Окно выполняющегося приложения

Из программы на Visual Basic можно обращаться к различным функциям MatrixVB. В данном примере мы использовали лишь некоторые из них. Разработанная программа очень проста и предназначена лишь для того, чтобы проиллюстрировать некоторые возможности MatrixVB. Используя библиотеку MatrixVB, можно создавать гораздо более сложные приложения.

Полностью код проекта MatrixVB расположен в каталоге Chapter 3\MatrixVB на прилагаемом к книге компакт-диске. Данный проект можно найти в каталоге Chapter 3\test. Полностью его исходный текст приведен в листинге 3.33.

Листинг 3.33. Полный код проекта MatrixVB

```
Option Explicit

Dim intSize As Integer, intAmp As Integer
Dim mRand As Variant, cRand As Variant
Dim ax As Variant, z As Variant

Private Sub cmdCreate_Click()

If Val(txtSize.Text) = 0 Or Val(txtAmp.Text) = 0 Then
    MsgBox "Enter the Values for Data Points and Amplitude "
    Exit Sub
End If

intSize = Val(txtSize.Text)
intAmp = Val(txtAmp.Text)
mRand = randn(1, intSize)

z = times(intAmp, mRand)
ax = vbaxes(picRandom.hWnd)
plot (z)
grid ("on")
Title ("Normal Distributed Random Signal")
xlabel ("Data Points")

End Sub

Private Sub cmdAnalysis_Click()

If Val(txtSize.Text) = 0 Or Val(txtAmp.Text) = 0 Then
    MsgBox "Enter the Values for Data Points and Amplitude "
    Exit Sub
End If

cRand = conv(mRand, mRand)
ax = vbaxes(picAnaly.hWnd)

plot (cRand)
grid ("on")
Title ("Convolution of Distributed Random Signal")
xlabel ("Data Points")

End Sub

Private Sub cmdExit_Click()
End
End Sub
```

3.8.5.3. Доставка готовой программы

MatrixVB предоставляет средства, упрощающие распространение скомпилированных приложений. Инсталляционная программа, представляющая собой исполняемый файл `C:\MatrixVB\bin\matrixvb4520rt.exe`, автоматически устанавливает и настраивает файлы MatrixVB на целевых машинах; при этом библиотека MatrixVB не требуется.

Для того чтобы доставить скомпилированное приложение с помощью инсталляционной программы, выполните следующие действия.

- Скопируйте файл `C:\MatrixVB\bin\matrixvb4520rt.exe` на целевую машину и запустите его.
- Введите PLP для завершения инсталляции.

После этого вы можете запускать программу, созданную на базе MatrixVB, на целевом компьютере.

3.8.6. Разработка приложения, выполняющего операции над матрицами

3.8.6.1. Программа калибровки робота

В данном разделе мы рассмотрим более сложный проект, созданный с использованием библиотеки MatrixVB. В этом примере операции над матрицами, реализуемые с помощью библиотеки MatrixVB, используются для вычисления ошибки позиционирования и составления графика распределения ошибки. В данном случае использование библиотеки MatrixVB демонстрируется на примере калибровки робота, так как при решении подобных задач широко используются операции над матрицами.

В процессе измерений эффлектор робота перемещается в трехмерном пространстве; при этом программа фиксирует его координаты. Сведения о позиции записываются в файл данных для дальнейшего использования. При выполнении измерений используются две системы координат: одна из них — система координат робота {B}, а вторая — мировая система координат {W}. Обычно мировой системой считается система координат, связанная с измерительным устройством.

Процесс идентификации базируется на результатах измерений. При этом используется алгоритм определения реальных параметров робота. После измерений и идентификации осуществляется верификация. Для ее выполнения используются два набора данных. Один из них содержит сведения о позиции, измеренной относительно мировой системы координат. Эти данные хранятся в файле `mes_wpos.txt` и включают в себя ошибку позиционирования. Второй набор данных получен при идентификации. Она представлена в системе координат робота {B} и хранится в файле `rel_bpos.txt`. Для того чтобы осуществить верификацию, необходимо сравнить результаты измерений с калибровочными данными в одной системе координат. В нашем распоряжении есть два набора данных, пред-

ставленных в разных системах. Мы используем операции над матрицами, чтобы преобразовать набор данных из одной системы координат в другую. Выполнив такое преобразование, можно приступить к сравнению.

Как вы уже знаете, одной из операций над матрицей является инверсия. Для того чтобы упростить код данного примера, мы будем считать, что два набора данных уже имеются в нашем распоряжении и хранятся в файлах C:\data\mes_Wpos.txt и C:\data\rel_Bpos. В файле C:\data\mes_Wpos.txt находятся данные о позиции в мировой системе координат после калибровки, а в файле C:\data\rel_Bpos содержатся результаты, представленные в базовой системе координат робота. Нам необходимо преобразовать первый набор данных из мировой системы координат в базовую систему координат робота и сравнить их, чтобы оценить результаты калибровки.

Создадим проект на Visual Basic, предназначенный для решения данной задачи.

3.8.6.2. Создание пользовательского интерфейса

Запустите Visual Basic и создайте новый проект MatrixVB. Выберите пункт меню Project⇒References и установите флажок в списке возле пункта MMatrix. Этим вы включите библиотеку MatrixVB в проект MatrixVB.

Добавьте в форму две кнопки, START и EXIT, и один объект Picture (рис. 3.25). Объекты и их имена указаны ниже.

Объект	Имя
Кнопка START	cmdStart
Кнопка EXIT	cmdExit
Объект Picture	picPos

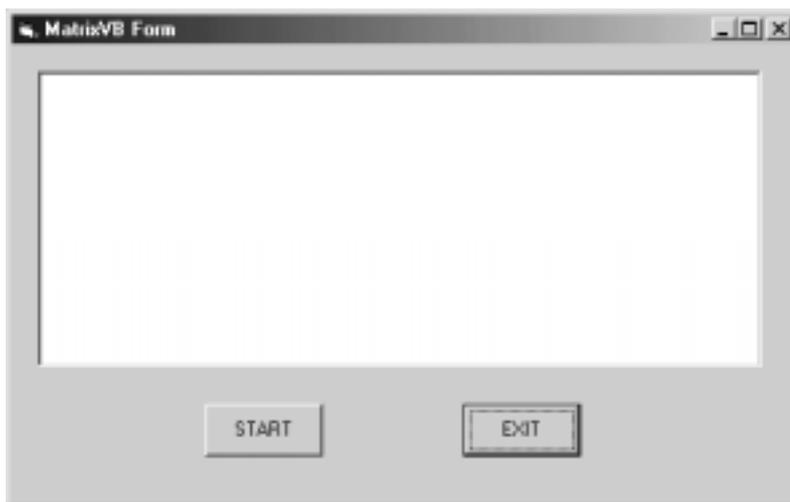


Рис. 3.25. Форма для программы калибровки робота

3.8.6.3. Создание кода программы

В первую очередь нам надо объявить переменные уровня формы. Откройте окно с кодом, выберите в списке Object пункт General, получив тем самым доступ к разделу General Declaration. Введите код, представленный в листинге 3.34.

Листинг 3.34. Код, включаемый в раздел General Declaration

```
Option Explicit

Dim m_Wpos(1 To 4, 1 To 20) As Single, i As Integer, j As Integer
Dim s(1 To 4) As Single, Trans(1 To 4, 1 To 4) As Single
Dim m_Bpos As Variant, r_Bpos As Variant, n_pos(1 To 20) As Single
Dim e_pos As Variant, ax As Variant
```

Большинство переменных имеет тип Variant, который широко используется в библиотеке MatrixVB. Назначение переменных описано в табл. 3.4. Включите в процедуру Form_Load код, показанный в листинге 3.35. Этот код инициализирует среду для выполнения операций над матрицами.

Таблица 3.4. Назначение переменных

Переменная	Назначение
m_Wpos(4, 20) — Matrix	Хранение данных об измерениях (файл mes_Wpos.txt)
m_Bpos — Variant	Хранение преобразованных данных в {B}
Trans(4, 4) — Matrix	Матрица преобразований
r_Bpos — Variant	Хранение данных калибровки (файл rel_Bpos.txt)
n_pos(20) — Vector	Хранение данных об ошибках позиционирования
e_pos — Variant	Хранение промежуточных данных об ошибках позиционирования
ax — Variant	Хранение вектора осей в рабочей области VB
i, j — Integer	Счетчики циклов

Листинг 3.35. Процедура Form_Load()

```
Private Sub Form_Load()
    j = 1
    Open "C:\data\TMatrix.txt" For Input As #1
    Do While Not EOF(1)
        For i = 1 To 4
            Input #1, Trans(j, i)
        Next i
        j = j + 1
    Loop
    Close #1
    j = 1
    Open "C:\data\mes_WPos.txt" For Input As #1
    Do While Not EOF(1)
        Input #1, s(1), s(2), s(3), s(4)
        For i = 1 To 4
```

```

    m_Wpos(i, j) = s(i)
  Next i
  m_Bpos = mtimes(inv(Trans), m_Wpos)
  j = j + 1
Loop
Close #1
m_Bpos = transpose(m_Bpos)

End Sub

```

Листинг 3.36. Матрица преобразования

```

0.9787      -0.197      0.056      2.517
0.1996      0.9791     -0.036     -0.444
-0.0477     0.0467     0.9977     0.2551
0           0           0           1

```

- В первую очередь при выполнении процедуры открывается файл `TMatrix.txt`, в котором хранится однородная матрица преобразования. Функция `Input` используется для чтения матрицы и записи ее в переменную `Trans`. Содержимое матрицы показано в листинге 3.36.
- Далее в процедуре открывается файл `mes_WPos.txt`, в котором находятся данные об измеренной позиции. Содержимое этого файла записывается в переменную `m_Wpos`. Переменная `s` используется для извлечения элементов из файла и помещения их в переменную `m_Wpos`. Эти действия осуществляются в циклах `for` и `Do While`. Выражение `EOF(1)` соответствует концу файла. Ввод продолжается до тех пор, пока выражение `Not EOF(1)` равно `True`.

Содержимое файла `mes_WPos.txt` представляет собой матрицу 20×4 . Значения, содержащиеся в файле, приведены в листинге 3.37. Каждая строка представляет собой вектор позиции. В файле находятся 20 таких векторов.

Листинг 3.37. Матрица, содержащаяся в файле `mes_WPos.txt`

```

11.0766      -0.0125     -10.8830      1
10.2723     -18.4004     -13.5890      1
-5.2630      -0.3080      -8.5059       1
-57.0573     15.9189     -25.9234      1
67.2476      -3.3661      2.3345       1
-25.5551     13.7436     -11.2634      1
75.9805     -15.1269      8.9952       1
26.7710     -16.3022     -4.8791       1
-58.4270     33.9845     -0.6599       1
-3.8053      27.4997     -12.7042      1
0.2054       7.8813     -17.8417      1
 38.5744     -5.0343     -9.0327       1
16.8961      3.5064      7.7533       1
12.2177     -1.6409     13.1715      1
 9.2936     -21.6886     -6.2385      1
14.2537     -1.0375     16.4671      1
42.2523     -8.5897      3.0063       1
12.7651      2.2541      2.7269       1
-6.4378      1.4510     -3.2350      1
-2.5668     -36.0615     -6.7076      1

```

Обратите внимание на одну особенность присвоения значения переменной `m_Wpos`. В цикле `for` присутствует выражение `m_Wpos(i, j) = s(i)`. В роли переменной цикла выступает `i`, а `m_Wpos` представляет собой матрицу 4×20 . Мы транспонируем вектор позиции, находящийся в файле `mes_Wpos.txt`, и присваиваем значение позиции матрице `m_Wpos`. Поскольку нам надо умножить эту матрицу на однородную матрицу преобразования `Trans`, размером 4×4 , необходимо обеспечить идентичность внутреннего представления двух матриц.

- В цикле мы также вызываем функции `inv()` и `mtimes()`. Функция `inv()` вычисляет матрицу, обратную матрице `Trans`, а функция `mtimes()` умножает обратную матрицу на вектор позиции. В результате получаем позицию в системе координат $\{B\}$. Рассмотрим причины, по которым необходимо использовать обратную матрицу.

Обычная операция перемножения матриц приведена ниже.

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_{z, \Theta} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Это выражение можно записать следующим образом:

$$W = T \times B$$

Символом W обозначается вектор позиции в мировой системе координат, а символ B соответствует тому же вектору в базовой системе координат робота. Используя приведенное выше выражение, мы, имея вектор позиции в базовой системе координат, можем определить тот же вектор в мировой системе координат.

Однако в данном примере мы измеряем позицию в мировой системе координат и должны сравнить вектор позиции с вектором в базовой системе координат. Для этого надо использовать следующее уравнение:

$$B = T^{-1} \times W$$

Таким способом мы можем преобразовать вектор позиции из мировых координат в базовые координаты робота и сравнить его с другим вектором.

- Нам также необходимо транспонировать матрицу `m_Wpos`, поскольку после вычисления предыдущего выражения мы получим матрицу размерностью 4×20 . Нам же для сравнения с матрицей, хранящейся в файле `rel_Wpos.txt`, нужна матрица размерностью 20×4 . По этой причине мы используем операцию `transpose`.
- Далее необходимо создать код процедуры, соответствующей кнопке `START`. После щелчка на этой кнопке вектор калибровочных значений, хранящийся в файле `C:\data\rel_WPos.txt`, должен быть загружен и

данные должны быть записаны в переменную `r_Vpos`. Затем надо сравнить преобразованные векторы измеренных позиций, находящиеся в переменной `m_Vpos`, с векторами калибровочных позиций, содержащимися в переменной `r_Vpos`. Для определения ошибок позиционирования используется функция `MatrixVB minus()`. Вектор ошибок присваивается переменной `e_pos`.

- Для отображения вектора ошибок на графике используется функция `abs()`. Двадцать значений ошибки присваиваются переменной `n_pos`, которая представляет матрицу размерностью 1×20 .
- При реализации данной операции мы используем средство `rN` библиотеки `MatrixVB`, которое позволяет выбирать элементы матрицы. В данном случае `N` означает размерность матрицы, поскольку мы имеем дело с двумерной матрицей, после `r` указано число 2. Символ `r` означает действительную часть комплексной матрицы. Таким образом, выражение `e_pos.r2(i, j)` позволяет извлечь элемент `e_pos(i, j)`. Кроме символа `r` может использоваться символ `i`. Так, `iN` означает мнимую часть комплексной матрицы размерностью `N`. В цикле мы суммируем четыре компонента ошибки для каждой строки и присваиваем результат элементу вектора `n_pos`.
- Затем мы используем функции `plot()` и `vbaxes()` для графического представления вектора ошибки в рабочей области Visual Basic.

Сохраните код программы и запустите ее на выполнение. Щелкните на кнопке `START`, и на экране появится изображение, подобное представленному на рис. 3.26. Полностью исходный текст проекта приведен в листинге 3.38.

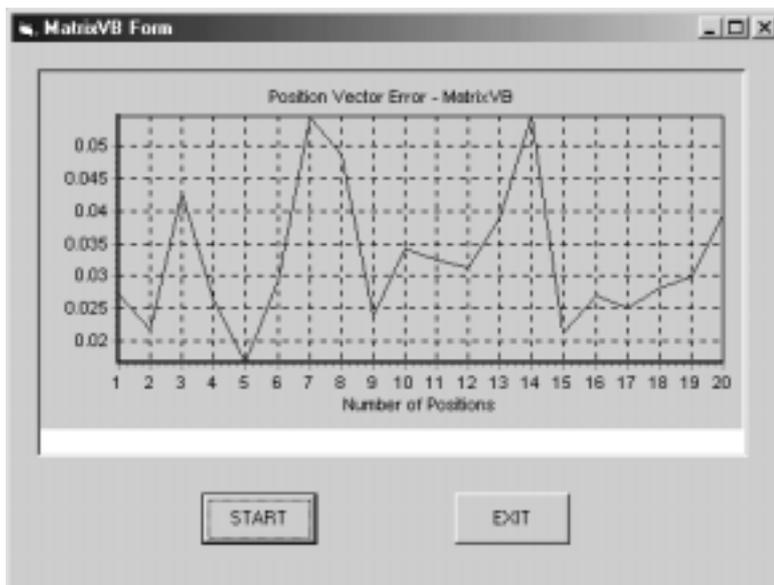


Рис. 3.26. Внешний вид окна выполняющегося приложения

Листинг 3.38. Исходный код готового проекта

```
Option Explicit
Dim m_Wpos(1 To 4, 1 To 20) As Single, i As Integer, j As Integer
Dim s(1 To 4) As Single, Trans(1 To 4, 1 To 4) As Single
Dim m_Bpos As Variant, r_Bpos As Variant, n_pos(1 To 20) As Single
Dim e_pos As Variant, ax As Variant

Private Sub cmdExit_Click()
End
End Sub

Private Sub cmdStart_Click()
r_Bpos = vload("C:\data\rel_BPos.txt")
e_pos = minus(r_Bpos, m_Bpos)
e_pos.Show
For i = 1 To 20
  For j = 1 To 4
    n_pos(i) = n_pos(i) + Abs(e_pos.r2(i, j))
  Next j
Next i
ax = vbaxes(picPos.hWnd)
plot (n_pos)
grid ("on")
Title ("Position Vector Error - MatrixVB")
xlabel ("Number of Positions")

End Sub

Private Sub Form_Load()
j = 1
Open "C:\data\TMatrix.txt" For Input As #1
Do While Not EOF(1)
  For i = 1 To 4
    Input #1, Trans(j, i)
  Next i
  j = j + 1
Loop
Close #1
j = 1
Open "C:\data\mes_WPos.txt" For Input As #1
Do While Not EOF(1)
  Input #1, s(1), s(2), s(3), s(4)
  For i = 1 To 4
    m_Wpos(i, j) = s(i)
  Next i
  m_Bpos = mtimes(inv(Trans), m_Wpos)
  j = j + 1
Loop
Close #1
m_Bpos = transpose(m_Bpos)

End Sub
```

Материалы проекта хранятся в каталоге `Chapter 3\VBMatrix` на прилагаемом к книге компакт-диске. Файлы, содержащиеся в этом каталоге, перечислены ниже.

- Файл формы, `Matrix.frm`.
- Файл проекта, `Matrix.vbp`.
- Исполняемый файл, `Matrix.exe`.
- Файл, содержащий матрицу преобразований, `TMatrix.txt`.
- Файл, содержащий сведения об измеренных позициях, `mes_WPos.txt`.
- Файл, содержащий сведения о калибровочных позициях, `rel_BPos.txt`.

Для того чтобы запустить программу на своем компьютере, скопируйте в каталог `c:\data` три файла данных: `TMatrix.txt`, `mes_WPos.txt` и `rel_BPos.txt`.

3.9. Взаимодействие программ, написанных на Java, с Matlab

3.9.1. Введение

Язык программирования Java пользуется большой популярностью среди разработчиков Internet-приложений. Java постоянно развивается, и новые версии данного языка предоставляют разработчикам Web-документов все более богатые возможности. Вначале Java создавался как средство для реализации небольших программ и встроенных приложений. Разработчики данного языка попытались упростить сложные структуры данных и алгоритмы, типичные для языка C++. В настоящее время Java используется в самых различных областях, например в производстве, банковской сфере, коммерческой деятельности и научных исследованиях.

Следует заметить, что Java плохо подходит для создания систем в реальном времени и сбора данных, в которых предполагается непосредственное обращение программ к специализированным аппаратным средствам. Однако в области Internet-программирования, взаимодействия с базами данных и в коммерческих приложениях данный язык существенно упрощает работу разработчиков. Java хорошо подходит для создания графических пользовательских интерфейсов и, что более важно, является объектно-ориентированным языком. Быстрое развитие Java сделало возможным использование данного языка для обмена данными между различными клиентами сети. Необходимость передавать по сети результаты измерений и обработки данных возникает при решении очень многих задач.

Как вы уже знаете, Matlab представляет собой мощный инструмент анализа данных в сложных вычислительных системах. Таким образом, организовав взаимодействие Java и Matlab, мы можем реализовывать приложения, предназначенные для выполнения сложных действий.

В разделе 3.9.2 описаны средства JMatLink, реализующие интерфейс между Java и Matlab. Дополнительную информацию по этому вопросу можно найти по адресу <http://www.held-mueller.de/JMatLink/>. Данный продукт бесплатно распространяется в виде исходных кодов и динамической библиотеки. Для организации взаимодействия Java и Matlab используется динамическая библиотека `JmatLink.dll`.

3.9.2. JMatLink

JMatLink представляет собой динамическую библиотеку, обеспечивающую взаимодействие Java и Matlab. Все функции этой библиотеки используют метод Matlab Engine или метод ActiveX. В программе на Java функции, предназначенные для взаимодействия с Matlab, вызываются в отдельных потоках. Синхронизация между потоками осуществляется посредством механизма ожидания.

Все функции JMatLink представляют собой платформенно-ориентированные методы. Они поддерживаются в системах Windows 95/98/NT/2000 и на различных платформах Unix. При переносе с одной платформы на другую изменять исходный код не требуется.

Средства JMatLink реализованы в виде подкласса класса `java.lang.Thread` (листинг 3.39).

Листинг 3.39. Суперклассы JMatLink

```

java.lang.Object
|
+---- java.lang.Thread
      |
      +---- JMatLink
  
```

3.9.2.1. Инсталляция библиотеки JMatLink

Файл библиотеки JMatLink можно скопировать, обратившись по адресу <http://www.held-mueller.de/JMatLink/>. Активизируйте ссылку Download и выберите исходные и двоичные коды версии 1.00 JMatLink. Сохраните zip-файл во временном каталоге своего компьютера и распакуйте архив, используя, например, в качестве целевого каталог `C:\JMatLink`. Если вы хотите выполнять математические вычисления и построение графиков в среде Matlab, вам необходимо также установить программное обеспечение Matlab и библиотеки, в частности Matlab Compiler, Matlab C/C++ Math Library и Matlab C/C++ Graphics Library. Основные этапы процесса инсталляции описаны ниже.

Инсталляция в среде Windows 95/98/Me

Чтобы установить библиотеку JMatLink в операционной системе Windows 95/98/Me, выполните следующие действия.

- Скопируйте на свой компьютер и распакуйте библиотеку JMatLink. В данном примере мы выбираем для размещения распакованных файлов каталог C:\JMatLink.
- Откройте файл autoexec.bat и добавьте путь к одному из следующих каталогов.

C:\jdk1.2\bin, если используется Java 2 SDK 1.2.

C:\jdk1.3.1_01, если используется Java 2 SDK 1.3.

C:\j2sdk1.4.0, если используется Java 2 SDK 1.4.

- Продолжая работу с файлом autoexec.bat, добавьте к переменной окружения CLASSPATH следующий путь:

```
SET CLASSPATH = C:\JMatLink
```

Это необходимо потому, что вам придется запускать программы из этого каталога.

- Перед тем как закрыть файл autoexec.bat, надо убедиться, что в нем присутствует один из путей к Matlab.
C:\matlab\bin, если используется Matlab 5.2.
C:\MatlabR12\bin, если используется Matlab 6.0.
C:\MATLAB6p1\bin\win32, если используется Matlab 6.1.
- Скопируйте библиотечный файл JMatLink.dll в каталог C:\WINDOWS\SYSTEM.

Инсталляция в системе Windows 2000

Чтобы установить библиотеку JMatLink в операционной системе Windows 2000, выполните следующие действия.

- Скопируйте библиотеку JMatLink и распакуйте архив в один из каталогов на своей машине, например в каталог C:\JMatLink.
- Откройте диалоговое окно Control Panel и дважды щелкните на пиктограмме System, чтобы открыть окно System Properties.
- В появившемся окне выберите вкладку Advanced и щелкните на кнопке Environment Variables, чтобы открыть диалоговое окно Environment Variables.
- В результате выполненных вами действий отобразятся два окна списка. В нижнем списке, System variables, выберите переменную Path. Щелкните на кнопке Edit, расположенной ниже списка, чтобы открыть диалоговое окно Edit System Variable.

- В диалоговом окне обратитесь к интерфейсному элементу `Variable Values` и добавьте к списку следующее значение.
C:\jdk1.2\bin, если используется Java 2 SDK 1.2.
C:\jdk1.3.1_01, если используется Java 2 SDK 1.3.
C:\j2sdk1.4.0, если используется Java 2 SDK 1.4.
- Добавьте следующий путь:
C:\JMatLink
Для разделения путей используется точка с запятой (;).
- Убедитесь, что путь к Matlab установлен корректно. Это должно быть одно из следующих значений.
C:\matlab\bin, если используется Matlab 5.2.
C:\MatlabR12\bin, если используется Matlab 6.0.
C:\MATLAB6p1\bin\win32, если используется Matlab 6.1.
- Щелкните на кнопке ОК, чтобы закрыть диалоговое окно.
- Продолжая работу со списком `System variables`, воспользуйтесь полосой прокрутки, чтобы перейти к верхней строке списка. Проверьте, имеется ли в системе параметр `CLASSPATH`. Если он отсутствует, щелкните на кнопке `New`, расположенной ниже окна списка, и откройте диалоговое окно `New System Variable`.
- В появившемся диалоговом окне в поле редактирования `Variable Name`: введите `CLASSPATH`. Заметьте, что это слово полностью состоит из прописных букв. Введите в поле `Variable Values` следующее значение:
C:\JMatLink
- Щелкните на кнопке ОК, чтобы закрыть диалоговое окно.
- Если в списке уже присутствует переменная `CLASSPATH`, щелкните на кнопке `Edit` и добавьте к списку значений `C:\JMatLink`.

Необходимо помнить, что в системе в каждый момент времени может присутствовать только одна переменная — `CLASSPATH`. Если при запуске байтового кода Java будет обнаружено две или более переменных `CLASSPATH` (например, одна в списке `User variables for default`, а другая в списке `System variables`), появится сообщение об ошибке. Все пути к классам надо задавать в составе одной переменной `CLASSPATH` и располагать ее либо в списке `User variables for default`, либо `System variables`. Если данное требование нарушено, обнаружить такую ошибку достаточно трудно. Отображаемое при этом сообщение информирует о том, что интерпретатор Java не может найти функцию `main()` программы, предназначенной для запуска. Проверив код программы и обнаружив там функцию `main()` (речь идет о приложении, а не об апплете), разработчик обычно остается в неведении об истинных причинах ошибки.

Библиотека JMatLink содержит набор функций, которые выполняют роль интерфейса при взаимодействии с функциями Matlab Engine. Обзор функций библиотеки JMatLink приведен в приложении А.

3.9.3. Вызов функции plot() из программы на Java

В данном разделе рассматривается простой пример, демонстрирующий вызов функции Matlab plot(), которая предназначена для построения графика. В разделе 3.9.4 мы решим более сложную задачу. Данный простой пример демонстрирует возможности интерфейса между Java и Matlab и общий подход к его реализации.

Активизируйте редактор Notepad и введите исходный код Java, показанный в листинге 3.40. Сохраните исходный текст в файле Jmlab1.java. В данной программе суперклассом является класс Frame, т.е. мы создаем независимое оконное приложение. Пользовательский интерфейс включает две кнопки, Start и Close. Интерфейс ActionListener превращает создаваемый класс в обработчик событий, связанный с активизацией интерфейсных элементов.

Листинг 3.40. Содержимое файла Jmlab1.java

```

/*****
* Файл:      Jmlab1.java
* Описание:  проверка взаимодействия Java и Matlab
*            (Java 1.3.1, Matlab 6.1)
* Дата:      4/19/2002
* Автор:     Y. Bai
*****/
import java.awt.*;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.io.*;

public class Jmlab1 extends Frame implements ActionListener
{
    JMatLink engine;

    Panel paneC = new Panel();
    Button start = new Button("Start");
    Button close = new Button("Close");
    Label lMsg = new Label("Wait...");
    Font    lFont = new Font("SansSerif", Font.BOLD, 18);
    Font    cFont = new Font("SansSerif", Font.BOLD, 14);
    Label lName = new Label(
        "Welcome to Test Java Matlab Interface ");
    String msg =
        " Plotting, please wait for a couple of seconds...";
    double data_array[] = {1.2, 2.3, 3.4, 8.1, 7.3,
        3.3, 2.1, 10.05, 5.88, 5.77 };

    public Jmlab1(String title)    // Конструктор
    {

```

```

super(title);

engine = new JMatLink();
engine.setDebug( false );
Panel paneN = new Panel();
Panel paneS = new Panel();
start.addActionListener(this);
close.addActionListener(this);
paneN.setBackground(Color.gray);
paneS.setBackground(Color.gray);
lName.setFont(lFont);
lMsg.setFont(cFont);
paneN.add(lName);
paneS.add(start);
paneS.add(close);
paneC.add(lMsg);

add(paneN, "North");
add(paneS, "South");
add(paneC, "Center");
addWindowListener(new WindowAdapter()
{ public void windowClosing(WindowEvent e)
  { System.exit(0); } } );
}

public void actionPerformed(ActionEvent evt)
{
    int EpI;

    Object src = evt.getSource();

    if (src==start)
    {
        lMsg.setText(msg);
        paneC.add(lMsg);
        paneC.validate();
        add(paneC, "Center");

        System.out.println("engOpen()");
        EpI = engine.engOpen();
        engine.engPutArray(EpI, "data_array", data_array);
        engine.engEvalString(EpI, "plot(data_array)");
        engine.engEvalString(EpI, "grid");
        engine.engEvalString(EpI,
            "title('Welcome to Java Matlab Interface Lab');");
        engine.engEvalString(EpI, "xlabel('Data Points');");
        engine.engEvalString(EpI, "ylabel('Amplitude ');");
    }
    if (src==close)
    {
        engine.engClose();
        engine.engClose();
        engine.kill();
        System.exit(0);
    }
    repaint();
}

```

```

    } // actionPerformed

    public static void main(String[] args)
    {
        Jmlab1 jm = new Jmlab1("Jmlab1");
        jm.setSize(500, 250);
        jm.show();
    }
}

```

В программе на Java создается массив значений типа `double`. Вы также можете создать его в среде Matlab и передать программе на Java с помощью функции `engGetArray()`. Для того чтобы упростить пример, мы не используем данный подход и объявляем массив в среде Java. Впоследствии мы передадим его в среду Matlab, где осуществляется построение графика.

Процессор Matlab создается путем обращения к конструктору класса `JMatLink` и в дальнейшем применяется для организации доступа к среде Matlab. В данном приложении используются три панели, `paneS`, `paneN` и `paneC`. Каждая из них расположена в отдельной области главного окна. По умолчанию с главным окном связывается диспетчер компоновки `BoardLayout`. Текстовая метка `lName` используется в качестве заголовка и располагается в области `North` окна. Другая текстовая метка, `lMsg`, отображает состояние выполняющейся программы.

Кнопки `Start` и `Close` являются источниками событий, а главное окно выступает в роли обработчика, или объекта прослушивания. Для регистрации главного окна в качестве обработчика используется метод `addActionListener()`.

При выполнении программы в центральную панель `paneC` включается текстовая метка `lMsg = "Wait. . ."`. После щелчка на кнопке `Start` генерируется событие, в результате чего активизируется процедура обработки `actionPerformed()`. В этот момент метка `lMsg` должна изменить свой внешний вид, отобразив текст `"Plotting, please wait for a couple of seconds. . ."`. Метод `validate()` сообщает выполняющей системе Java о том, что главное окно должно быть обновлено. Не вызвав метод `validate()`, вы не увидите нового текста метки.

После генерации сообщения и вызова метода `actionPerformed()` необходимо обратиться к процессору Matlab и получить возвращаемые данные. При этом надо использовать ряд функций Matlab, выполняющих следующие задачи.

- Функция `engPutArray()` помещает массив данных в рабочую область Matlab.
- Функция `engEvalString()` передает каждую функцию в среду Matlab для выполнения требуемых действий.
- После того как данные и функции будут успешно переданы в среду Matlab и построение графика завершится, вызывается функция `engClose()`, которая закрывает процессор Matlab. Вызов этой функции осуществляется щелчком на кнопке `Close`.

Код программы достаточно прост. Основную нагрузку несут на себе функции `engOpen()` и `engEvalString()`. Функция `engOpen()` устанавливает канал связи между Java и Matlab, а с помощью функции `engEvalString()` мы передаем Matlab-функции в строковом формате в среду Matlab.

В окне DOS перейдите в каталог `C:\JMatLink`. Скомпилируйте и запустите на выполнение Java-программу. Главное окно работающей программы должно выглядеть так, как показано на рис. 3.27. Щелкните на кнопке `Start`, чтобы передать в среду Matlab команду на построение графика. Посредством метки `lmsg` отобразится текст `Plotting, please wait for a couple of seconds. . .`. Несколькими секундами позже появится окно Matlab с графиком, подобное показанному на рис. 3.28.



Рис. 3.27. Главное окно выполняющейся программы

Как видите, функции Matlab Engine вызываются сравнительно просто. Организуя интерфейс между Matlab и Java с помощью библиотеки JMatLink, можно решать задачи, которые предполагают сложные математические вычисления. Щелкните на кнопке `Close`, чтобы убрать с экрана график, и завершите выполнение программы, щелкнув на кнопке `Close` главного окна.

3.9.4. Передача параметров и получение результирующих данных

В данном разделе рассматривается пример передачи параметра функции Matlab и использования библиотечной функции `engGetScalar()` для получения ответа из среды Matlab. Создадим функцию Matlab. Код этой функции показан на рис. 3.41. Сохраните функцию в файле `JavaFunc.m` и поместите этот файл в каталог `C:\MATLAB6p1\work`.

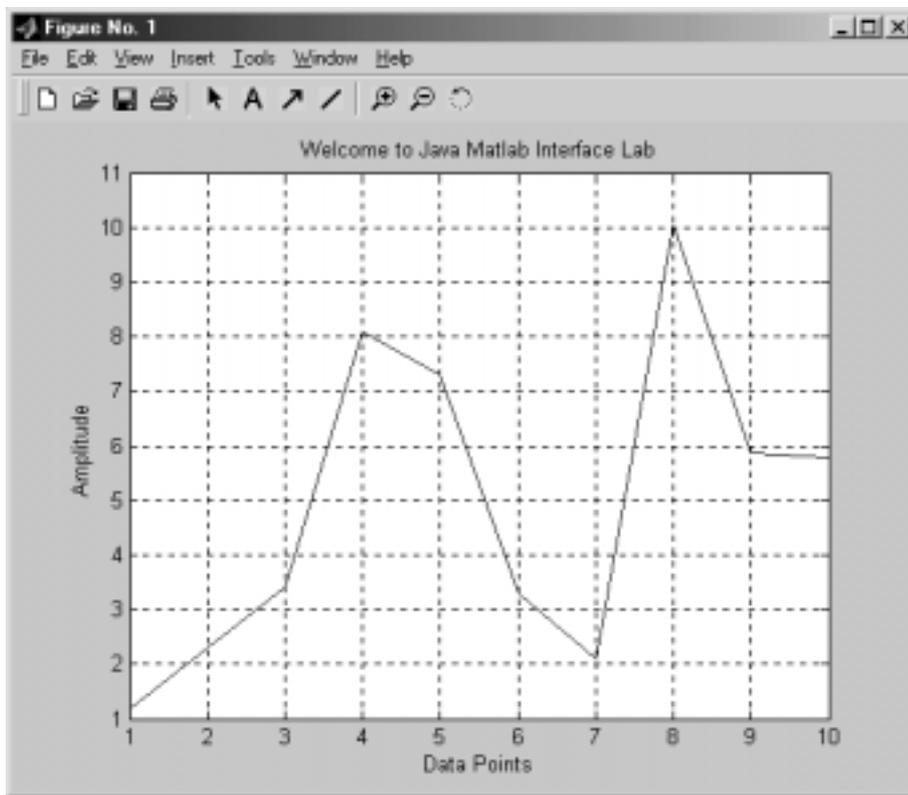


Рис. 3.28. Отображение графика в окне Matlab

Листинг 3.41. М-функция JavaFunc()

```

% Функция для проверки обращений из Java-программ
% 4/20/2002
% Y. Bai

function rc = JavaFunc(size)

data = randn(1, size);

plot(data);
grid;
title('Plot a Random Data Sequence for Java Program');

rc = 50;

```

Данная функция предназначена в основном для демонстрации возможностей программы на Java. Эта функция вызывается из программы на Java, и при вызове ей передается параметр `size`. Возвращаемое значение, `rc`, позволяет убедиться в том, что средства взаимодействия Java и Matlab работают корректно.

В теле созданной функции вызывается функция `randn()`. Этим создается набор псевдослучайных значений с нормальным распределением. Число элементов в наборе определяется параметром `size`, передаваемым из Java-программы. Функция `plot()` вызывается для построения графика, а по завершении выполнения функции программе на Java передается значение, равное 50.

Введите код программы на языке Java, показанный в листинге 3.42, и сохраните его в файле `JMtest.java`.

Листинг 3.42. Содержимое файла `JMtest.java`

```

/*****
 * Файл:      JMtest.java
 * Описание:  проверка механизма передачи данных Matlab
 * Дата:      4/20/2002
 * Автор:     Y. Bai
 *****/
import java.awt.*;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.io.*;
public class JMtest extends Frame implements ActionListener
{
    JMatLink engine = new JMatLink();
    Label answer = new Label("Answer = ? ");
    Label lsize = new Label("Number of Data");
    Button start = new Button("Start");
    Button close = new Button("Close");
    Font cFont = new Font("SansSerif", Font.BOLD, 14);
    Panel paneS = new Panel();
    Panel paneC = new Panel();
    Panel paneW = new Panel();
    TextField tsize = new TextField(10);
    public JMtest(String title)
    {
        super(title);
        answer.setFont(cFont);
        lsize.setFont(cFont);
        paneC.add(lsize);
        paneC.add(tsize);
        paneC.add(answer);
        paneS.add(start);
        paneS.add(close);
        paneS.setBackground(Color.gray);
        start.addActionListener(this);
        close.addActionListener(this);
        add(paneS, "South");
        add(paneC, "Center");
        add(paneW, "West");
    }
    public void actionPerformed(ActionEvent evt)
    {
        int a;
        double rc;

```

```

Object src = evt.getSource();

if (src==start)
{
    System.out.println("engOpen()");
    //Создание сеанса Matlab
    a = engine.engOpenSingleUse();
    engine.engEvalString(a,
        "rc = JavaFunc(" + tsize.getText() + ")");
    //Получение данных от Matlab
    rc = engine.engGetScalar(a, "rc");
    //Преобразование rc в строку
    answer.setText(Double.toString(rc));
    //Обновление главного окна
    paneC.validate();
}
if (src==close)
{
    engine.engClose(a);
    engine.kill();
    System.exit(0);
}
repaint();
} // actionPerformed

public static void main(String[] args)
{
    JMtest jm = new JMtest("JMtest");
    jm.setSize(400, 200);
    jm.show();
}
}

```

Когда пользователь щелкает на кнопке Start, вызывается функция `engOpenSingleUse()`, открывающая сеанс Matlab. С помощью этой функции можно открыть несколько сеансов. Функция `engEvalString()` используется для вызова Matlab-функции `JavaFunc()` с параметром `size`. Этот параметр определяет количество элементов псевдослучайного набора. Значение параметра пользователь вводит посредством объекта `textfield`. Для извлечения значения параметра и передачи его Matlab-функции `JavaFunc()` используется метод `getText()`. После этого производится обращение к функции `engGetScalar()`, которая позволяет получить данные, передаваемые из среды Matlab. Поскольку данные возвращаются в формате числа с двойной точностью, функция `Double.toString()` преобразует их в строковый вид, после чего они отображаются с помощью метки `answer`. Функция `validate()` информирует выполняющую систему о том, что появились новые данные, предназначенные для отображения, поэтому необходимо обновить окно.

Сохраните код программы в каталоге `C:\JMtest` в файле с именем `JMtest.java`. Откройте окно DOS и скомпилируйте в нем программу. Запустите программу на выполнение и введите значение 120, определяющее число элементов в наборе данных. Щелкните на кнопке Start, чтобы передать функ-

цию с параметром в среду Matlab. В результате будет выполнена функция `JavaFunc()`, создан массив псевдослучайных значений размером `1×size` и построен график. Окно работающей программы должно выглядеть так, как показано на рис. 3.29.

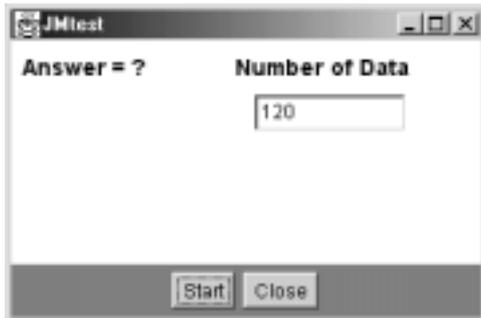


Рис. 3.29. Окно работающего приложения

Когда функция `JavaFunc()` завершает действия по созданию графика, метка на панели отображает значение (50.0). Это значение `JavaFunc()` возвращается программе на Java (рис. 3.30). Кроме того, на экране отображается график, показанный на рис. 3.31.

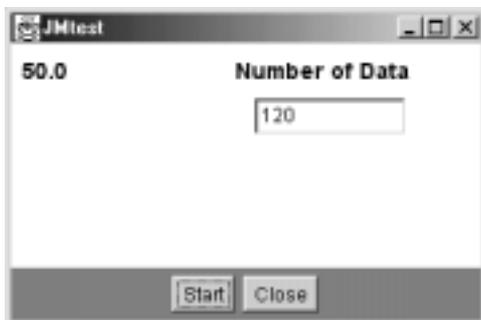


Рис. 3.30. `JavaFunc()` возвращает Java-программе значение, равное 50.0

По мере необходимости вы также можете использовать числовой формат параметров и, объединяя их со строкой путем конкатенации, передавать в среду Matlab. Например, если вы хотите передать функции Matlab переменную `dSize` типа `double`, вы можете использовать следующее выражение:

```
engine.evalString(a, "rc = JavaFunc(" + dSize + ")");
```

На рис. 3.31 видно, что число значений равно 120. Именно эта величина была введена в программе на Java. Как видите, приложение работает корректно. Закройте окно с графиком, щелкнув на кнопке `Close` в его верхнем правом углу, и завершите программу, щелкнув на кнопке `Close` в окне, созданном средствами Java.

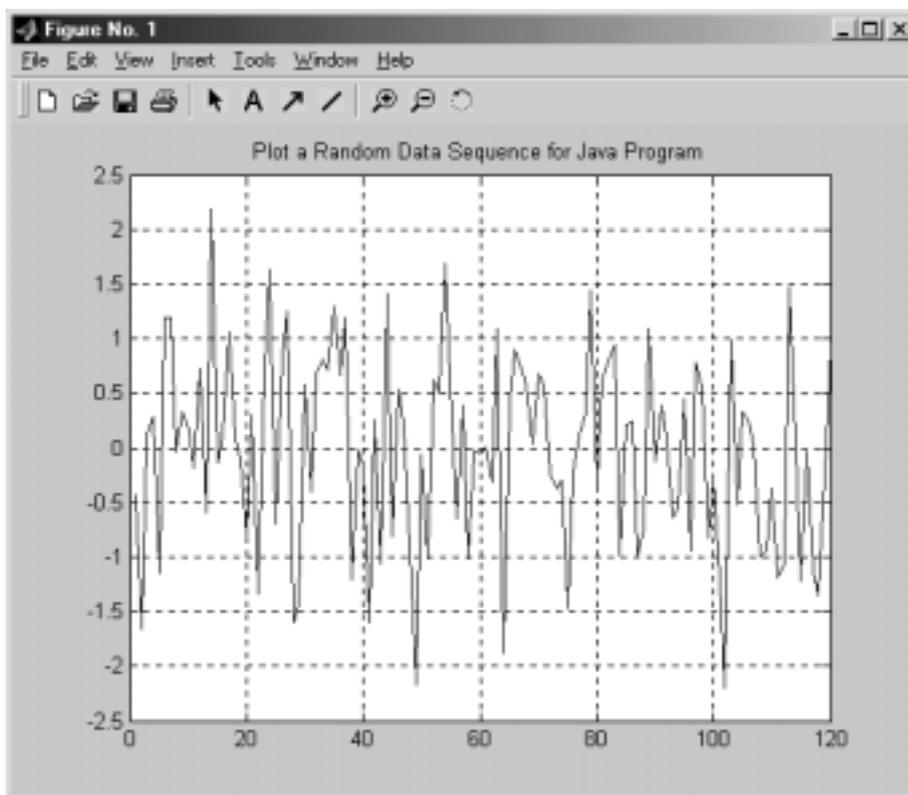


Рис. 3.31. График, сформированный в процессе выполнения программы

3.9.5. Взаимодействие апплета с Matlab

3.9.5.1. Создание Matlab-функции

В данном разделе рассматривается пример совместного использования Java-апплета и Matlab. Функции, выполняемой в среде Matlab, передается параметр `dim`, который задает размерность массивов, хранящихся в файлах данных. Функция `JavaMatrix.m` получает параметр `dim` и выполняет следующие задачи.

- Инициализирует матрицы.
- Загружает данные из файла `mes_wpos.txt`, содержащего векторы измеренных позиций. Эти векторы представлены в виде матрицы размерностью 20×4 .
- Загружает данные из файла `rel_wpos.txt`, содержащего векторы калибровочных позиций. Эти векторы представлены в виде матрицы размерностью 20×4 .
- Определяет матрицу преобразования `Trans` размерностью 4×4 .

- Вычисляет вектор измеренных позиций в системе координат робота. Для этого используется обратная матрица преобразований.
- Вычисляет ошибки, используя для этого векторы измеренных позиций и калибровочные векторы в системе координат робота.
- Отображает распределение ошибок в виде графика.

Для того чтобы продемонстрировать действие функции `JavaMatrix.m`, необходимо рассмотреть принципы преобразования из одной системы координат в другую. На рис. 3.32 показаны две системы координат: система, связанная с роботом, и мировая система координат, связанная с измерительным устройством. В процессе калибровки робота необходимо использовать устройство, позволяющее определить позицию эффектора в обеих системах координат. Для того чтобы сравнить векторы позиций, измеренные в двух системах координат, следует преобразовать данные измерений из одной системы в другую.

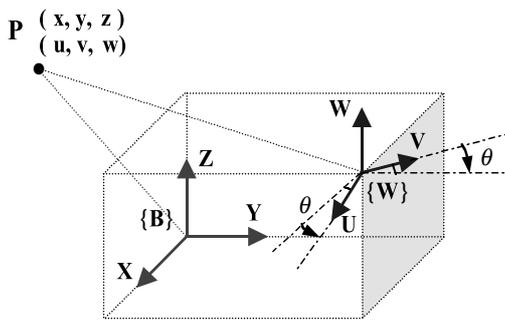


Рис. 3.32. Преобразование координат

При работе реального приложения невозможно определить системы координат так, чтобы их оси были строго параллельны, поэтому неизбежны ошибки рассогласования. Подобные ошибки возникают, во-первых, из-за того, что прочность физических устройств ограничена, а во-вторых, потому, что начала координат двух систем не совпадают. На рис. 3.32 показана ошибка, возникающая вследствие вращения мировой системы координат вдоль оси Z. Эта ошибка представлена углом θ . В реальном приложении рассогласование существует по всем осям. Чтобы упростить данный пример, мы будем предполагать, что рассогласование наблюдается только по одной оси.

Для того чтобы преобразовать вектор позиции из одной системы координат в другую, необходимо иметь информацию о соответствии между системами. Эта информация представляется в виде однородной матрицы преобразования `Trans` размерностью 4×4 элемента. Предположим, что в нашем распоряжении имеются матрица `Trans` и позиция `P`, измеренная в системе координат робота (рис. 3.32). Для преобразования данных из системы координат робота в мировую систему координат можно использовать следующее соотношение:

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 & 0 \\ \sin \Theta & \cos \Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_{z,\Theta} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

где $T_{z,\Theta}$ — однородная матрица преобразований Trans.

В данном примере в нашем распоряжении имеется 20 векторов позиций, измеренных в мировой системе координат, и столько же векторов, измеренных в системе координат робота. Нам необходимо определить ошибку. Для этого необходимо преобразовать данные о позициях из мировой системы координат в систему координат робота. Это означает, что для вычисления целевой позиции должна применяться матрица, обратная Trans:

$$B = T^{-1}W$$

Векторы позиций, измеренные в мировой системе координат, хранятся в файле `mes_WPos.txt`, который расположен в каталоге `C:\data`. Векторы позиций, соответствующие системе координат робота, записаны в файле `rel_BPos.txt`, находящемся в том же каталоге.

Как было сказано ранее, функция `JavaMatrix.m` должна загрузить два вектора позиций и вычислить векторы позиций в системе координат робота, используя операции с матрицами. После этого определяются ошибки и в среде Matlab строится график, отражающий их распределение.

Данный пример демонстрирует использование Java-апплета для вызова по сети в среде Matlab сложных операций с матрицами. Коды данного примера могут послужить базой для более сложных программ.

При обработке векторов позиций с помощью матрицы преобразования надо транспонировать матрицу, которая отражает данные, полученные в мировой системе координат. Как вы уже знаете, размерность матрицы преобразования составляет 4×4 , а в файле `mes_WPos.txt` содержатся векторы позиций в виде матрицы 20×4 .

Исходный код M-функции `JavaMatrix.m` показан в листинге 3.43. Параметр `dim` задает размер вектора. Данную функцию надо сохранить в рабочем каталоге Matlab `C:\MATLAB6p1\work`.

Листинг 3.43. M-функция `JavaMatrix()`

```
% Вычисление позиции с помощью однородной матрицы преобразования
% Данная функция вызывается из Java-апплета
function JavaMatrix(dim)

B_pos = zeros(dim,4);
Trans = zeros(4, 4);
W_pos = zeros(dim, 4);
B_pos1 = zeros(4, dim);
e_pos = zeros(dim, 4);
```

```

err = zeros(dim);

m_Wpos = load('C:\data\mes_WPos.txt');
r_Bpos = load('C:\data\rel_BPos.txt');

Trans = [ 0.9787 -0.197  0.056  2.517;
          0.1996  0.9791 -0.036 -0.444;
          -0.0477 0.0467  0.9977 0.2551;
          0       0       0       1];

for i = 1:dim
    B_pos1(:, i) = inv(Trans) * m_Wpos(i, :);
end

B_pos = B_pos1'
e_pos = B_pos - r_Bpos

for n = 1:dim
    err(n) = norm(e_pos(n));
end

plot(err, 'b-');
grid
title('Measured and Actual Position Error Distribution ');
xlabel('Number of Positions ');
ylabel('Position Error ');

```

3.9.5.2. Создание Java-аплета

Откройте редактор Notepad и введите в нем исходный код Java-аплета. В окне аплета отобразятся две текстовые метки. Одна из них представляет собой заголовок, а вторая выводит данные о состоянии аплета. Кроме того, в этом окне имеются две кнопки: Start и Close. Эти кнопки используются для запуска и завершения программы.

Интерфейс ActionListener, который реализует класс аплета, превращает этот класс в обработчик событий, связанный с двумя кнопками. Исходный код аплета показан в листинге 3.44. После щелчка на кнопке Start вызывается процедура обработки события, в теле которой содержимое текстовой метки lMsg изменяется с "Click Start Button to Start . . ." на "Plotting, please wait . . .". Метод validate() сообщает о том, что содержимое окна аплета следует обновить. Если не вызывать этот метод, текстовая метка будет отображать предыдущий текст.

Листинг 3.44. Код аплета, содержащийся в файле JMApplet.java

```

/*****
* Файл:      JMApplet.java
* Описание:  Java Applet program to interface to Matlab
* Дата:      4/21/2002
* Автор:     Y. Bai
*****/

```

```
import java.awt.*;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.applet.*;

public class JMApplet extends Applet implements ActionListener
{
    JMatLink engine;

    Button start = new Button("Start");
    Button close = new Button("Close");
    Label lMsg = new Label("Click Start Button to Start...");
    Font lFont = new Font("SansSerif", Font.BOLD, 18);
    Font cFont = new Font("SansSerif", Font.BOLD, 14);
    Label lName = new
        Label("Test Java-Matlab Interface by Applet ");
    String msg = " Plotting, please wait....";

    public void init()
    {
        start.addActionListener(this);
        close.addActionListener(this);
        lName.setFont(lFont);
        lMsg.setFont(cFont);
        add(lName);
        add(lMsg);
        add(start);
        add(close);
    }

    public void actionPerformed(ActionEvent evt)
    {
        int dim = 20;

        engine = new JMatLink();
        Object src = evt.getSource();

        if (src==start)
        {
            lMsg.setText(msg);
            validate();
            engine.engOpen();
            engine.engEvalString("JavaMatrix(" + dim + ")");
        }
        if (src==close)
        {
            engine.engClose();
            engine.kill();
            System.exit(0);
        }
    }
}
```

Переменная `dim` может быть непосредственно добавлена к строке. На платформе Java целочисленная переменная автоматически преобразуется в строку, которая вместе с остальными символами, заданными в виде литерала, передается функции Matlab.

Сохраните исходный код Java в каталоге `C:\JMatLink` в файле `JavaApplet.java`. В окне DOS откройте компилятор, который преобразует код апплета в файл класса `JavaApplet.class`.

Откройте редактор Notepad и введите следующий HTML-код:

```
<html>
<applet code = "JMatApplet.class" width = 400 height = 150>
</applet>
</html>
```

Сохраните его в файле `JavaApplet.html` в каталоге `C:\JMatLink`, т.е. в том же каталоге, что и `JavaApplet.java`. Откройте Web-браузер, например Internet Explorer. Для того чтобы открыть Internet Explorer, щелкните на соответствующей пиктограмме на рабочем столе. В диалоговом окне Connect To щелкните на кнопке Work Offline. Это необходимо, поскольку мы не собираемся работать в Web, а лишь проверим работоспособность апплета. Если на экране появится несколько диалоговых окон, продолжайте щелкать на Stay Offline. Откройте диалоговое окно Open, щелкнув на пункте меню File⇒Open. Активизируйте кнопку Browse и найдите HTML-файл `JavaApplet.html`, который находится в каталоге `C:\JMatLink`. Выберите этот файл, щелкните на кнопке Open, а затем на кнопке OK, чтобы открыть апплет. Окно апплета должно выглядеть так, как показано на рис. 3.33.

Щелкните на кнопке Start. Текстовая метка должна измениться на "Plotting, please wait. . ." (рис. 3.34). Через несколько секунд на экране отобразится график распределения ошибок (рис. 3.35), построенный в среде Matlab.

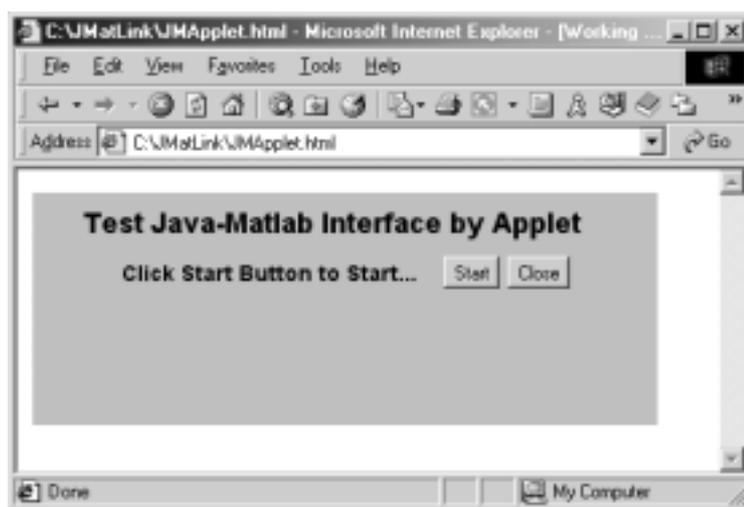


Рис. 3.33. Окно апплета, отображаемое с помощью браузера



Рис. 3.34. По щелчку на кнопке Start текстовая метка изменяется

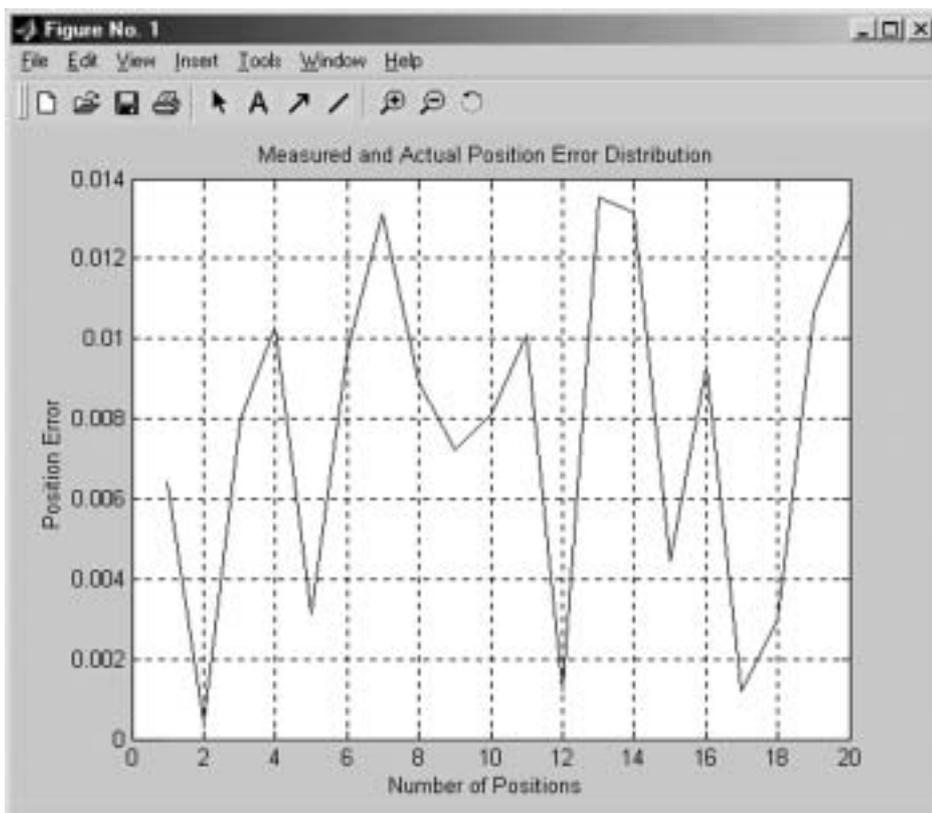


Рис. 3.35. График распределения ошибок, сформированный средствами Matlab

Щелкните на кнопке Close, чтобы завершить выполнение апплета. Для того чтобы апплет мог работать, следует хранить файлы данных `mes_WPos.txt` и `rel_BPos.txt` в каталоге `C:\data`. Вам также надо разместить функцию Matlab `JavaMatrix.m` в рабочем каталоге Matlab `C\MATLAB6p1\work`. Исходный код Java-апплета `JavaApplet.java` и HTML-файл `JavaApplet.html` также находятся в каталоге `C:\JMatLink`.

Все указанные файлы находятся в папке `Chapter 3\JMatLink` на прилагаемом к книге компакт-диске. Скопировав их на свой компьютер, вы можете запустить апплет на выполнение, но прежде следует убедиться, что система правильно сконфигурирована.

3.10. Разработка управляющих систем, работающих в реальном времени

Во многих приложениях, в особенности в управляющих программах, приходится создавать процедуры для решения уравнений. Часто алгоритмы решения уравнений со многими неизвестными бывают сложными, поэтому приходится затрачивать большие усилия, чтобы реализовать их в программе на VC++. При решении уравнений используются операции с матрицами. Создавая приложения в виде программ на VC++ и Matlab, взаимодействующих между собой, проблему удастся решить достаточно просто.

Блок-схема, которая демонстрирует вызов Matlab-функции, осуществляющей операции с матрицами, из программы на VC++, показана на рис. 3.36. Для организации взаимодействия VC++ и Matlab можно использовать любой из способов, обсуждавшихся в данной главе.

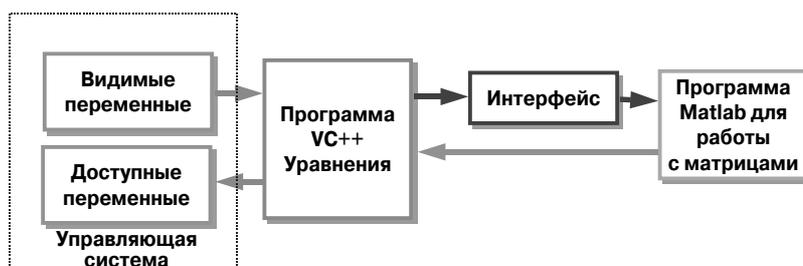


Рис. 3.36. Вызов функции Matlab из программы на VC++

3.10.1. Выводы

На основании материала, изложенного в данной главе, можно сделать вывод о том, что взаимодействие программ на VC++ и Matlab позволяет существенно расширить возможности управляющих приложений в реальном времени и обеспечить выполнение сложных математических вычислений в рамках подобных приложений. Благодаря применению данного подхода удастся объеди-

нить управляющие модули, базирующиеся на одной платформе, и средства, реализующие математические вычисления, предполагающие использование другой платформы.

Без участия Matlab в среде, ориентированной на управление в реальном времени, например C или VC++, чрезвычайно трудно реализовать сложные алгоритмы обработки данных. Описанные в данной главе подходы предоставляют новые возможности для создания управляющих систем.