между отдельными предметами, проводимыми в одном семестре, менее желательны и требуют детальной корректировки двух календарных учебных планов, чтобы они могли дополнять друг друга, а не происходило изолированное изучение отдельных предметов, несвязанных между собой.

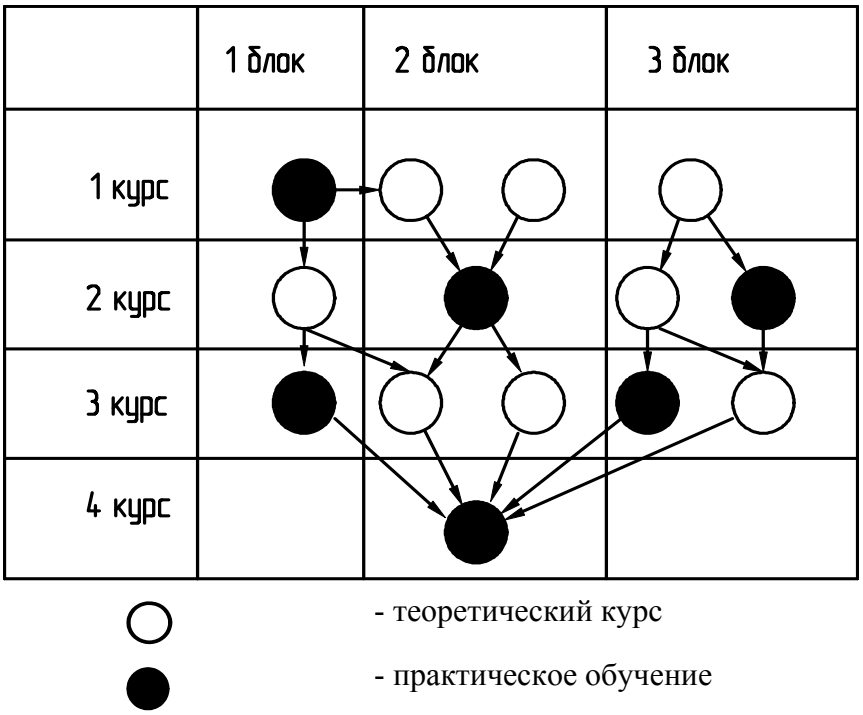

Рисунок 1 – Пример сетевого графика взаимосвязи блоков дисциплин

Большое внимание в настоящее время отводится самостоятельной работе студентов. Однако все еще отсутствует механизм обеспечения эффективности самостоятельной работы. Эффективная самостоятельная работа студентов возможна только при сочетании теоретического изучения и практического применения результата.

Разработка программно-аппаратного интерфейса для использования его в учебном процессе при комплексном изучении языков программирования различных уровней

доц. Холодов Г.М., Солопова О.И., Поповкин А.В.
МГТУ «МАМИ»
(495) 223-05-23, доб. 1305

Аннотация. В статье рассматривается разработка интерфейса для проектов MATLAB и его реализация на языке программирования C, схемы и методы взаимодействия приложения C и среды MATLAB, в том числе по локальной вычислительной сети (ЛВС). Приведена методика использования такого интерфейса. Сделаны выводы о необходимости создания интерфейса между приложением C и пакетом прикладных программ для решения задач технических вычислений, и возможности комплексного изучения студентами языков программирования различных уровней.

Ключевые слова: программирование; MATLAB; язык программирования C; программно-аппаратный интерфейс; создание интерфейса; высокоуровневый язык программирования; низкоуровневый язык программирования; взаимодействие разноязыковых программ; взаимодействие MATLAB с ANSI C; среда программирования Microsoft Visual C++.

Введение

Программное обеспечение систем управления техническими объектами, состоит из программных средств, выполненных на языках низкого и высокого уровней, которые обладают различными возможностями в решении задач управления. Для выработки управляющих воздействий объектом управления необходимо решение вычислительных задач, основанных на идее искусственного интеллекта, и других сложных задач, предполагающих обработку больших массивов данных и знаний. Эти задачи могут быть решены только программными средствами на языке высокого уровня. Как свидетельствует учебная практика, между программным обеспечением низкого и высокого уровня существует необходимость создания эффективного аппаратно-программного интерфейса для решения всего комплекса задач в реальном режиме времени.

Одним из мощных пакетов, предназначенных для решения научно-технических задач, является *MATLAB*. Необходимо реализовать пользовательский интерфейс приложения для работы с *MATLAB*, осуществить взаимосвязь с пользователем удаленно, разработать схемы, посредством которых будет возможно осуществлять удаленное использование пакета прикладных программ для решения сложных задач. Используя приложение на *C*, разработкой которого мы занимаемся, пользователь (студент) может осуществлять взаимодействие со средой *MATLAB* посредством, создаваемого этим приложением интерфейса.

При решении некоторых задач часто возникает необходимость обработки матриц. Помимо обработки матриц часто бывает необходимо составлять различные графики. Организовать обработку матриц средствами лишь *C* или *C++* достаточно трудно, а программист-разработчик, пытающийся решить задачу представления графики средствами *C++,* вынужден самостоятельно реализовывать сложные алгоритмы. *MATLAB* – удобный инструмент для визуализации данных. Он обеспечивает создание графиков 2D и 3D. Реализовать трехмерную графику средствами *C* или *C++* чрезвычайно трудно. Интерфейс между приложением на *C* и *MATLAB* позволяет осуществить данные операции.

Обычно программист работает с системой *MATLAB* непосредственно, кодируя необходимые алгоритмы на встроенном языке *MATLAB*. Встроенный язык весьма удобен для написания математических алгоритмов - писать и отлаживать на нем программу занимает значительно меньше времени, чем на обычных языках программирования.

Однако довольно часто эффективность подобных программ оставляет желать лучшего. Как правило, подобная ситуация возникает, когда алгоритм плохо векторизуется, например, при обработке матриц нельзя выразить этот алгоритм, пользуясь векторными операторами языка *MATLAB* и приходится писать вложенные циклы, перераспределять память и т.п. В этом случае программа на языке *C* будет исполняться во много раз быстрее аналогичной программы на языке *MATLAB*. А ведь довольно часто время счета математической задачи может исчисляться сутками.

Кроме того, иногда возникают ситуации, когда те или иные сложные алгоритмы уже были реализованы на других языках программирования. В этом случае также будет быстрее не переписывать весь алгоритм на языке *MATLAB*, а написать аппаратно-программный интерфейс для *MATLAB* к уже существующему на языке *C* модулю и вызвать М-файл в приложение *C*.

Необходимо реализовать такую программу ввиду того, что *MATLAB* имеет свой язык программирования (функциональный язык программирования) и обычный пользователь зачастую может оказаться не способен решить, казалось бы, какую-либо тривиальную задачу из-за барьера в виде языка программирования, непонятного среднему пользователю. Поможем этому пользователю, создав удобный и, самое главное, понятный для него пользовательский интерфейс.

Но самой главной задачей всё же является создание интерфейса для взаимодействия низкого (*Assembler*, *C*) и высокого (*MATLAB*) уровней. К примеру, с какого-то технологиче-

ского оборудования при помощи контроллера будут сниматься данные и при помощи программы, написанной на высокоуровневом языке программирования, над этими данными будут производиться необходимые манипуляции. При помощи такого интерфейса можно организовать автоматизацию создания баз данных и сэкономить, таким образом, много времени и свести к минимуму вероятность ошибки. Без такого программно-аппаратного интерфейса, разработке которого и посвящена статья, приходилось бы привлекать дополнительный персонал, который бы осуществлял создание баз данных, и при обработке большого объема информации вероятность ошибки в таком случае имела бы место.

Схема применения такого взаимодействия: данные снимаются при помощи контроллера с какого-либо технологического оборудования программой на низкоуровневом языке программирования и передаются программе высокого уровня языка программирования, которая в свою очередь осуществляет обработку данных, полученных с контроллера. Далее осуществляется автоматизированное создание базы данных.

Предлагается применять такой подход в учебном процессе. То есть изучать языки программирования не в отдельности, а рассматривать создание приложений с учетом интеграции разноуровневых языков, принимая во внимание особенности каждого.

Анализ возможностей *MATLAB* в решении сложных задач управления объектами.

Приближение процесса обучения к реальным системам и процессам управления можно осуществить при помощи инструмента *MATLAB* для моделирования, который называется *Simulink*. Это интерактивный инструмент для моделирования, имитации и анализа динамических систем. Он дает возможность строить графические блок-диаграммы, имитировать динамические системы, исследовать работоспособность систем и совершенствовать проекты. *Simulink* полностью интегрирован с *MATLAB*, обеспечивая немедленным доступом к широкому спектру инструментов анализа и проектирования. Эти преимущества делают Simulink наиболее популярным инструментом для проектирования систем управления и коммуникации, цифровой обработки и других приложений моделирования. Мы будем использовать этот мощный инструмент для обучения, вызывая ту или иную библиотеку в приложение на *C*. Обучающийся сможет наглядно увидеть работу какого-либо механизма, используя для этого его модель.

Программа *Simulink* является приложением к пакету *MATLAB*. При моделировании в *Simulink* реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты. При этом, в отличие от классических способов моделирования, пользователю не нужно досконально изучать язык программирования и численные методы математики, а достаточно общих знаний, требующихся при работе на компьютере и, естественно, знаний той предметной области, в которой он работает.

Преимущество *Simulink* заключается также в том, что он позволяет пополнять библиотеки блоков с помощью подпрограмм, написанных как на языке *MATLAB*, так и на языках *C/C++, Fortran* и *Ada*. Это дает возможность программистам на этих языках использовать большие способности *Simulink* для решения задач моделирования.

Основные положения разработки интерфейса и его реализация.

Для создания интерфейса была использована технология *OLE (Object Linking and Embedding)*, разработанная корпорацией *Microsoft* для связывания и внедрения объектов. Эта технология позволяет осуществлять разделение объектов между прикладными программами.

MATLAB имеет язык высокого уровня и для решения задач управления необходимо разработать интерфейс. Чтобы создать интерфейс между *MATLAB* и приложением *C* воспользуемся технологией *OLE*. Эта технология позволяет передавать часть работы от одной программы редактирования к другой и возвращать результаты назад. Ключевым элементом технологии *OLE* является Автоматизация (*Automation*, *OLE Automation* – технология на осно-

ве модели компонентных объектов *Microsoft COM*). Автоматизация позволяет приложению, встроенному в другое приложение, активизировать себя и управлять своей частью пользовательского интерфейса, благодаря чему она может вносить изменения и затем завершать свою работу.

Осуществим связь *OLE*-контейнера (клиента) и *OLE*-сервера. Чтобы встроить один *OLE*-объект в другой *OLE*-объект, следует реализовать встроенный объект как *OLE*-сервер, а объект, содержащий первый объект, должен быть в этом случае *OLE*-контейнером. Пусть наше приложение на *C* будет *OLE*-контейнером, а *Matlab OLE*-сервером, выполняющим математические операции, вычисления, построения графиков и т.д. Будем работать из нашего приложения в *MATLAB*, вызвав его на выполнение в контейнер *OLE*. Обмен данными происходит через динамически отводимую память и не требует знания формата принимаемых данных. Компонент организует связь с объектами, которые в него помещаются косвенно, по ссылке на связываемый с *OLE*-объект.

В общем случае клиент и сервер находятся в разных адресных пространствах, и, соответственно, для управления сервером клиент должен обращаться к методам объектов, находящихся в другом адресном пространстве. Для этой цели используется технология *LRPC* (*Local Remote Procedure Calls* - локальные вызовы удаленных процедур).

Приложение на *C* реализовано методом использования *Matlab Engine*. Данный метод эквивалентен методу OLE и предполагает использование библиотеки *Matlab* Engine для взаимодействия с функциями Matlab. При использовании библиотеки *Matlab Engine* интерфейс между приложением *C* и *Matlab* существенно упрощается. Подобно управляющим элементам *OLE*, после обращения к среде *Matlab* и выполнения некоторых действий приложение *C* сохраняет контроль над выполнением программы. Еще одно преимущество использования библиотеки *Matlab Engine* состоит в том, что при этом нет необходимости хранить данные в файле в среде *VC++* и извлекать их, открывая файл в среде *Matlab*. Вместо этого появляется возможность сформировать данные в программе *C* и вызывать М-функции, передавая им информацию в виде массивов. [2]

Чтобы воспользоваться библиотекой *Matlab Engine*, необходимо включить ее при помощи директивы препроцессора в программу: *#include "engine.h*" /* локальный заголовочный файл ищется в том каталоге, в котором расположен исходный код */.

Этот заголовочный файл размещается в каталоге `C:\MATLAB6p1\extern\include.`

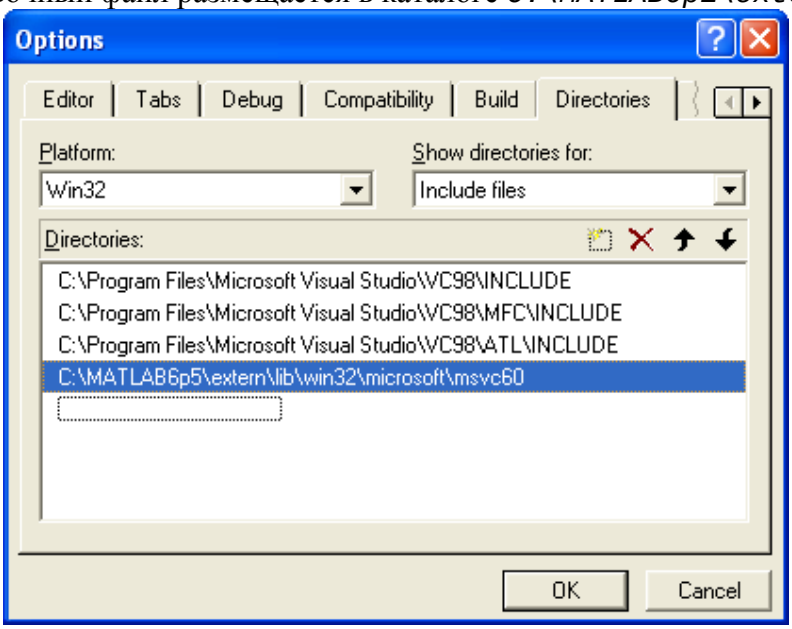

Рисунок 1 - Диалоговое окно Options, вкладка Directories

Его надо скопировать в локальный каталог, предназначенный для включаемых файлов,

и указать этот каталог на вкладке *Directories* в окне *Options,* что поясняется на рисунке 1.

Для того чтобы отобразить диалоговое окно *Options*, в окне *Visual C++* необходимо выбрать пункт *Options* меню *Tools*. Добавить файлы заголовков к проекту можно также, вызвав пункт меню *Add To Project -> Files*.

На рисунке 2 поясняется, что необходимо на вкладке *Link* окна *Project Settings* надо указать приведенные ниже библиотечные файлы: *libeng.lib*, *libmx.lib*. Оба указанных файла находятся в каталоге C:\MATLAB6p1\extern\lib\win32\Microsoft\msvc60.

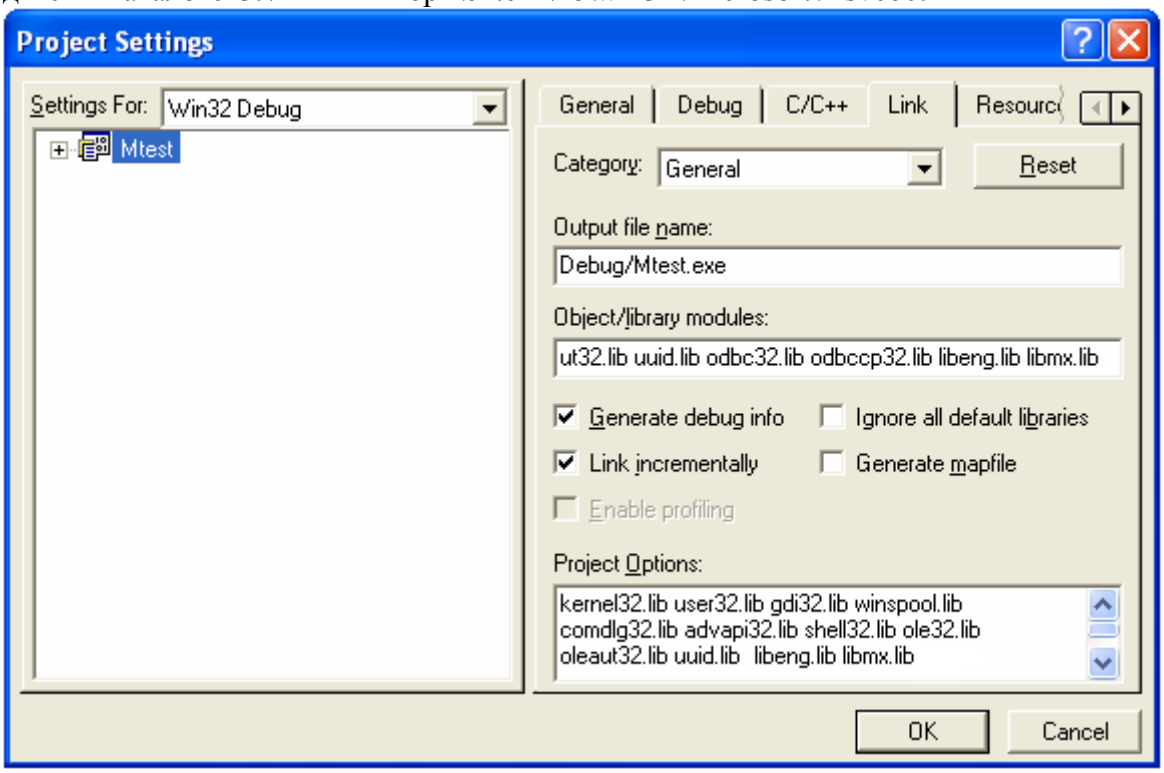

Рисунок 2 - Вкладка Link диалогового окна Project Settings

Эти файлы были скопированы в локальный каталог *lib*, был указан путь к библиотекам на вкладке *Directories* окна *Options*. Для этой цели можно также выбрать пункт *Project -> Add To Project -> Files.*

Детальная последовательность событий, происходящих при запуске приложения С поясняется в таблице 1. Рассмотрим некоторые моменты конкретнее.

Таблица 1.

№	**Последовательность событий при запуске приложения С**
1	Вызов вычислительного ядра *MATLAB*`а.
2	Вызов М-файла.
3	Появление и задержка на экране результата вычисления М-файла.
4	Появление окна "*PLOT IS SUCCESSFUL*!"
5	Выводится в консоли "*Hello, MATLAB*!" (строка, которая может быть заменена на любую другую. В данном случае она символизирует успешность взаимодействия приложения *C* со средой *MATLAB*).
6	После нажатия "*OK*" происходит закрытие М-файла, закрытие командного окна *MATLAB* `а и самого приложения.

После запуска вычислительного ядра происходит вызов М-файла, сформированного студентом для решения прикладной задачи. Далее студент получает результат вычислений М-файла, что поясняется на рисунке 3.

В рассмотренной программе для вызова М-функций используется строковый формат. Вызов окна сообщения несет двойную функциональную нагрузку. Во-первых, уведомляет

пользователя об успешности или неудаче вызова вычислительного ядра MATLAB, а во-вторых, осуществляет задержку результата вычислений на экране.

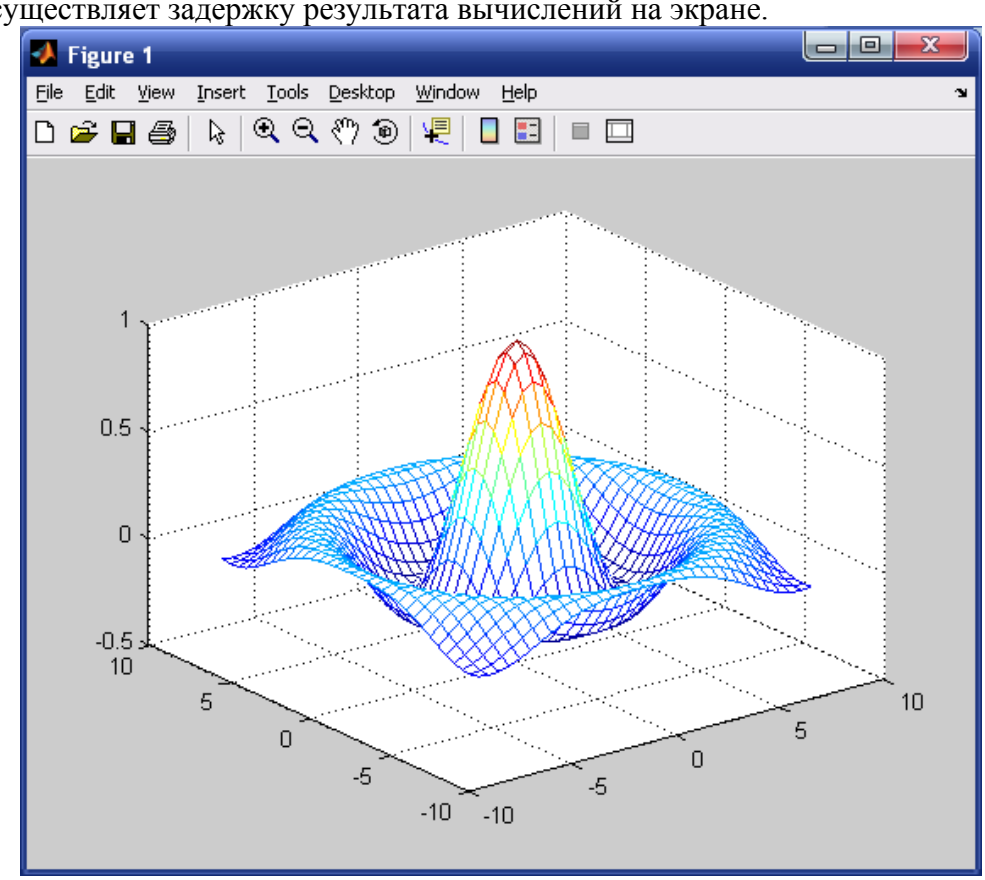

Рисунок 3 - Результат выполнения программы

Окно с сообщением (*Message Box*) запрограммировано на следующие события:
rc = MessageBox((HWND)NULL, (LPSTR)"PLOT IS SUCCESSFUL!", (LPSTR)"Mtest.c", MB_OK); /* эта инструкция обеспечивает появление окна с сообщением, свидетельствующим об удачном прохождении операции */;
rc = MessageBox((HWND)NULL, (LPSTR)"Can't start MATLAB engine", (LPSTR) "Mtest.c", MB_RETRYCANCEL); /* эта инструкция обеспечивает появление окна с сообщением, которое говорит нам о том, что по каким-либо причинам вычислительное ядро *MATLAB* не может быть вызвано. Это может произойти, если на клиенте не установлен *MATLAB* (в случае использования, приложение *C* на одном компьютере) или сервер с *MATLAB* не доступен (в случае использования приложения *C* по ЛВС */.

Данная методика обучения применяется в магистратуре на первом и втором году обучения по дисциплине «Компьютерные технологии в области автоматизации и управления» по направлению «Магистр управления и информатики». Всего отведено по учебному плану на эту дисциплину 110 часов, из которых 72 часа аудиторных занятий и 38 на самостоятельную работу. Данная программа используется в лабораторном практикуме. Разработаны программы на языке MATLAB по работе и созданию многомерных массивов ячеек с помощью функции cat. Функция cat позволяет формировать многомерные массивы ячеек. Темы лабораторных работ: «Трехмерные массивы ячеек», «Вложенные массивы», «Преобразование типов данных», «Тестирование имен ячеек», «Создание массива символьных ячеек из массива строк и визуализация массивов ячеек». Средствами *C* невозможно столь же легко создавать трехмерные массивы ячеек и визуализировать данные, поэтому используется лишь интерфейс, написанный на *C*, а для вычислений и визуализации используются мощности MATLAB посредством вызова вычислительного ядра. Студенты формируют М-файл с помощью любого удобного текстового редактора, что поясняется на рисунке 4.

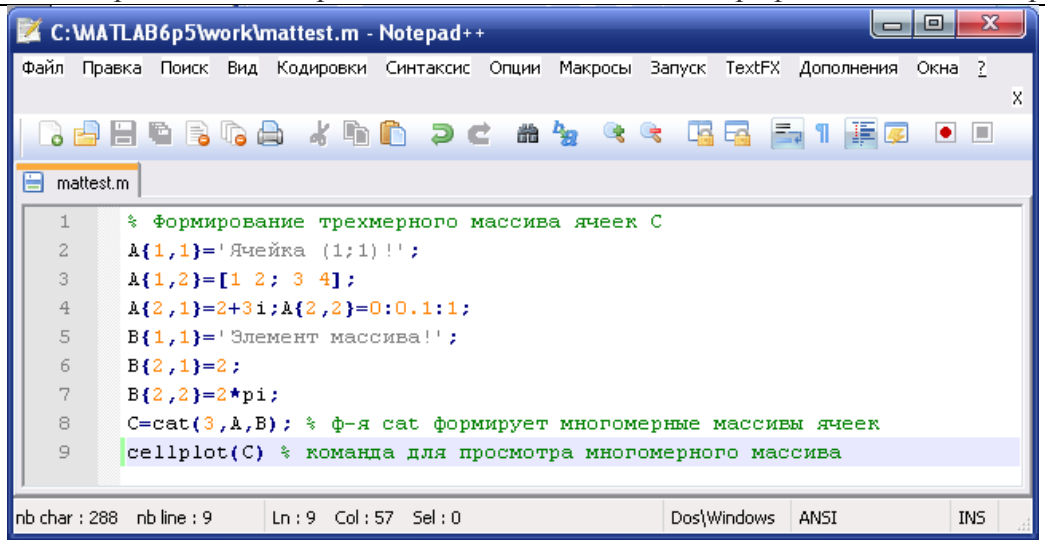

Рисунок 4 – Выполнение лабораторной работы «Создание многомерных массивов ячеек» с помощью свободного текстового редактора *Notepad* ++

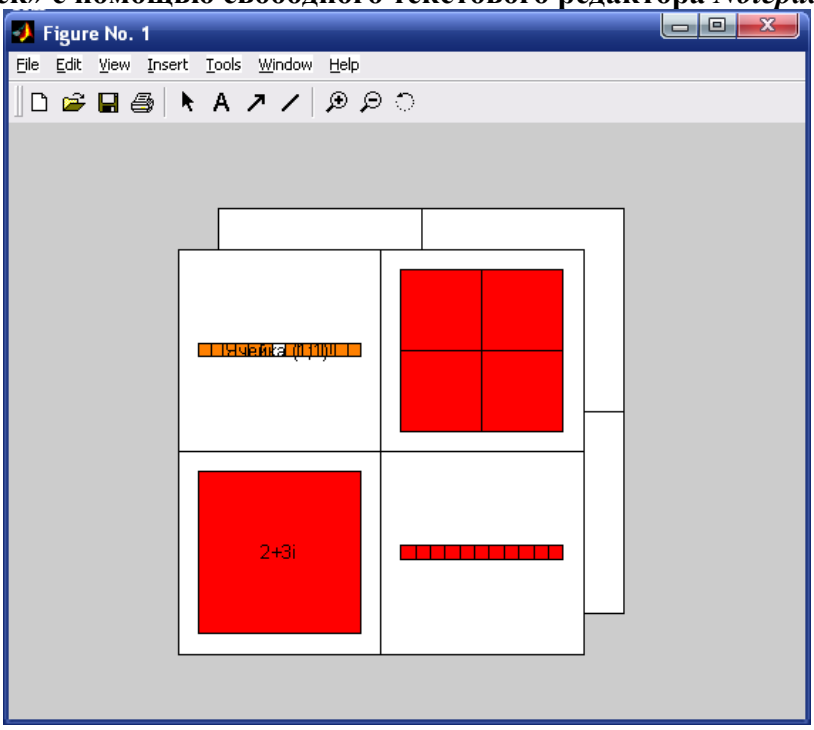

Рисунок 5 – Визуализация результата выполнения лабораторной работы «Создание многомерных массивов ячеек»

Они не прибегают к самому *MATLAB*, а запускают лишь приложение на *C*, после чего вызывается вычислительное ядро *MATLAB*, производятся необходимые вычисления, и студент может видеть результаты своей работы, что поясняется на рисунке 5. Новая методика проведения лабораторного практикума по дисциплине «Компьютерные технологии в области автоматизации и управления» прошла апробацию и в настоящее время внедрена в учебный процесс. Качественная реакция преподавателей и обучаемых на новую методику положительна.

В программе демонстрируется использование *Matlab Engine* для вызова М-функции, осуществляющей построение трехмерных графиков, создание многомерных массивов ячеек. Но при желании можно решать и любые другие задачи. Данная программа компилируется в исполняемый файл (.exe) и может выполняться в среде *Visual C++ 6.0* или непосредственно в системе *Windows*.

Рассмотрим схему взаимодействия приложения *C* и среды *MATLAB* (рисунок 6) по ЛВС. Эта схема применяется для обучения студентов в компьютерном классе, оснащенном локальной сетью. Организация решения задач происходит через сервер, оснащенный системой *MATLAB*, и подключенный в сеть с другими абонентами сети, которые в свою очередь выполняют приложение *C*. Приложение *C* выполняется абонентами, на которых не установлен *MATLAB*. Это могут быть маломощные компьютеры, так как для вычислений они используют ресурсы сервера. Используем для реализации такой схемы топологию «Звезда», при которой к одному центральному компьютеру присоединяются остальные периферийные компьютеры, причем каждый из них выполняет приложение *C*, использующее вычислительное ядро *MATLAB*. Конфликты в сети с топологией «звезда» невозможны, так как управление полностью централизовано. Топология может быть любая, это может быть и «шина», и «кольцо», «активное дерево» или «пассивное дерево» в зависимости от целей использования сети. В таком случае достаточно иметь лишь одну лицензионную копию *MATLAB*, которая будет установлена на сервере, а остальные абоненты, выполняющие приложение *C* будут использовать вычислительное ядро *MATLAB* посредством созданного интерфейса. При получении данных программа, написанная на языке программирования *C*, обращается к инструментальным средствам *MATLAB*, который, в свою очередь, выполняет необходимую обработку данных. Результаты обработки снова передаются программе на *C*, которая использует их для управления объектом. На сервере в системе *MATLAB* прописаны пути к М-файлам, в которых содержится решение задачи. Таким образом, преподаватель может просмотреть со своего рабочего места содержание М-файла конкретного студента и запустить его на исполнение. Интерфейс между приложением *C* и *MATLAB* позволяет существенно повысить уровень систем сбора и анализа данных.

Рисунок 6 - Схема взаимодействия приложения C и среды MATLAB по топологии «Звезда»

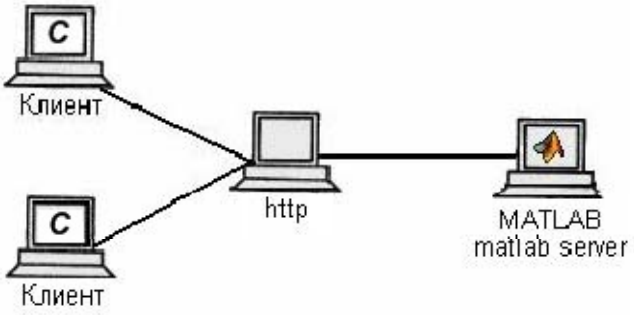

Рисунок 7 - Схема взаимодействия приложения *C* и среды *MATLAB* посредством глобальной сети

Можно также в дальнейшем на основе, созданного интерфейса реализовать схему взаимодействия приложения *C* и среды *MATLAB* для дистанционного обучения посредством гло-

бальной сети (рисунок 7). Эта схема может применяться для обучения студентов удаленно.

Как и любое современное приложение, *MATLAB* поддерживает возможности сетевых технологий посредством глобальной сети. Для этого в пакет включено приложение *MATLAB Web Server*, с помощью которого можно разрабатывать программы *MATLAB*, взаимодействующие с сетями Интернет. Используя *MATLAB Web Server*, можно организовать обработку данных, получаемых по сети, и отображать их в Web-обозревателе. В простейшем случае эта схема выглядит так: обозреватель, установленный в клиентской части сети, принимает информацию, идущую от серверной части, в которой установлен *MATLAB* и *MATLAB Web Server*. Через приложение, написанное на С, которое установлено на клиентской машине есть возможность использовать данные, которые будут отправляться на сервер с установленным на нем *MATLAB*. Тогда *MATLAB* решает задачу с этими данными, результат которой передается обратно клиенту. Внедрение метода использования такой схемы через глобальную сеть не производилось, ввиду того, что дисциплина «Компьютерные технологии в науке, технике и технологии» не проводится для студентов, обучаемых дистанционно, вечерников и заочников.

Заключение

В данной статье рассмотрено, разработанное приложение, которое позволяет осуществлять обучение студентов, рассматривая программно-аппаратные средства СУ (системы управления) в комплексе, начиная с ассемблера, *C* и до сложных приложений, написанных на языках высокого уровня. Разработан интерфейс на языке программирования *C*, с помощью которого стало возможным взаимодействие программы, реализованной на низкоуровневом языке и среды *MATLAB*, которая работает на языке высокого уровня. Появилась возможность использования вычислительных ресурсов и мощных средств визуализации данных из приложения *C*. Приведена методика и схемы использования такого интерфейса в учебном процессе, в том числе по ЛВС. В результате внедрения такой методики обучения заметно улучшилось представление студентов о взаимодействии языков программирования высокого и низкого уровней, технологии OLE, работе сетевых приложений, технологии «клиент-сервер».

Литература

1. Черных И.В. *Simulink*: Инструмент моделирования динамических систем, Консультационный центр MATLAB компании Softline, 2008. - Режим доступа: http://matlab.exponenta.ru/simulink/book1/index.php, свободный.
2. Инг Бей, Джифенг Ксу. Взаимодействие *MATLAB* с *ANSI C*, *Visual C++*, *Visual Basic* и *Java*, гл. 3 - М.: Вильямс, 2005.- 207 с.
3. Инг Бей. Взаимодействие разноязыковых программ в Microsoft Windows. Руководство программиста. / Инг Бей. – М.: Вильямс, 2005.-880 с.
4. Пахомов Б.И. С/С++ и Borland C++ Builder для начинающих**.** / Пахомов Б.И – СПб.: БХВ-Петербург, 2007.- 628 с.
5. Подкур М.П. Программирование в среде Borland C++ Builder с математическими библиотеками MATLAB C/C++. / Подкур М.П., Смоленцев Н.К., Подкур П.Н. – М.:ДМК Пресс, 2006. – 496 с.
6. Наталия Елманова. Создание серверов автоматизации с помощью C++ Builder, Центр Информационных Технологий, 1999. - Режим доступа: http://www.citforum.ru/programming/cpp/autosrv.shtml, свободный.