Ульяновский Государственный Университет

Механико-математический факультет

Кафедра математической кибернетики и информатики

“Web – технологии в обучении и преподавании”

курсовая работа

Выполнил :
Студент гр. ПМ – 31 Насибуллин Т. Г.

Научный руководитель :
Проф. Семушин Иннокентий Васильевич

Ульяновск, 2004 год

1.Введение

Целью этого небольшого исследования является изучение возможностей формата **PDF** для создания интерактивных электронных документов, в частности учебных пособий по математическим дисциплинам. Рассмотрена технология создания таких **PDF**-документов с использованием языка **JavaScript**, приведены также некоторые сведения о самом формате **PDF** и средствах для работы с ним.

Для демонстрационной реализации математических алгоритмов часто создаются обычные программные продукты на различных языках программирования. **PDF**-формат имеет в этом отношении некоторые преимущества :

- **PDF** – независящий от платформы формат, а программы пишутся под определённую операционную систему.
- Многие научные труды распространяются в виде **PDF**-документов, и реализация алгоритмов с использованием того же формата может быть добавлена к основному документу как приложение.

Для демонстрации было разработано пособие по курсу “Численные методы” с реализацией алгоритмов исключения для решения систем линейных алгебраических уравнений.

2.Общие сведения о формате PDF

В 1982 году в фирме **Adobe Systems**, **Inc** был разработан язык **PostScript**. **PostScript** создавался в качестве простого стандартного языка для описания вида чертежей и простых изображений на печатаемой странице, что наложило определенные штампы на используемые в языке конструкции. **PostScript** достаточно богат, он содержит более 250 операторов, что позволяет одни и те же действия описывать самыми различными способами.

Треть языка посвящена графике, остальное - это обычный процедурный язык программирования, который включает в себя элементы из многих других языков, но по структурному построению он ближе всего к языку Фортан. Одним из важнейших достижений языка является его независимость от устройства, на котором созданная страница будет воспроизведена. Файлы в формате **PostScript** записываются в виде 7-битовых ASCII-символов и могут быть созданы с помощью обычного текстового редактора.

PDF (Portable Document Format) - является отдельным направлением в развитии языка **PostScript** . Целью его разработки явилась необходимость

обеспечения пользователей простым и надежным форматом для обмена и просмотра электронных документов независимо от создающих их приложений, то есть основным инструментом для «безбумажного офиса». В основе PDF лежит координатная и цветовая кроссплатформенная модель языка **PostScript**. Но для эффективности диалоговой работы **PDF** имеет более структурированный формат, чем большинство **PostScript**-программ, и некоторые семантические особенности. PDF может содержать такие объекты, как аннотации и гипертекстовые связи, при этом текст, векторные и растровые изображения, формирующие содержание страницы, предоставляются операторами языка **PostScript**.

С момента появления формат рассматривался лишь применительно к электронному документообороту фирм. Поэтому отличительной чертой от **PostScript** является возможность печати **PDF**-файла в устройствах, не поддерживающих **PostScript**. Только в 1993 году с появлением второй его редакции **PDF** стал рассматриваться в качестве будущего серьезного конкурента **PostScript**. Спецификация языка 1.1 предусматривала отображение цвета независимо от устройства, функции сглаживания шрифтов, вставки аннотаций, защиту документов паролем и т.д.

Последняя на сегодняшний день редакция — спецификация **PDF** 1.3. К новым возможностям этой редакции можно отнести конвертирование и сохранение файлов из World Wide Web в **PDF**-файл, возможность представления логической структуры документа отдельно от ее графической структуры, внедрение в документ файлов любых форматов, поддержка **JavaScript**.

PDF является файловым форматом, а не языком программирования, хотя он и описывает элементы страницы, используя **PostScript**. Он не может быть прямо интерпретирован PostScript-интерпретатором. Однако страницы из **PDF**-файла могут быть преобразованы в программу на **PostScript**. В свою очередь, **PostScript**-программу также можно преобразовать в **PDF**-формат.

PDF-файл не содержит процедур, переменных и управляющих конструкций. Операторы **PDF** реализуют формирование изображения безо всяких аналитических вычислений, что приводит к более быстрой работе.

PDF-формат является расширяемым и версионируемым, что делает приложения совместимыми снизу вверх.

Структурно **PDF**-файл состоит их четырех составных частей. Объекты с некоторыми исключениями соответствуют типам данных в языке **PostScript**. Структура файла определяет, как объекты хранятся в файле, какой к ним доступ и как они редактируются. Структура документа описывает как базовые объекты, которые используются для представления страниц, аннотаций, гипертекстовых связей и шрифтов. Четвертый компонент — описание страниц на **PostScript** с выше упомянутыми отличиями.

PDF-файл содержит таблицу перекрестных ссылок, используемую для произвольного доступа к страницам и объектам. Использование этой таблицы делает время доступа независимым от объема документа, что немаловажно при диалоговой работе. Реальные **PDF**-документы содержат

сотни страниц. **PDF**-формат дает возможность инкрементального редактирования с добавлениями нового содержимого без перезаписывания старого.

Для просмотра **PDF**-файлов используется бесплатная программа фирмы **Adobe** Acrobat Reader. Более сложные продукты Adobe Acrobat Standart и Adobe Acrobat Professional позволяют создавать и редактировать **PDF**-документы, последняя поддерживает также использование **JavaScript**. С **PDF**-форматом, кроме Acrobat и его расширений (PitStop, CrackJack, Quite Imposing Plus, Ari`s Power Bundle), на сегодняшний день работают такие программы, как: Illustrator, FreeHand, InDesign, Alap Imposer, Impostrip, Presswise, Corel DRAW, Flightcheck и некоторые другие.

Международный комитет по стандартизации полиграфических технологий GATS (Graphic Arts Technologies Standards) несколько лет назад определил PDF как приоритетный стандарт в программе цифрового распространения печатной продукции.

В настоящее время **PDF** используется в качестве единого формата хранения и обмена документов во всем мире, в том числе и для научных публикаций. Примером может служить его применение в США в Департаменте Коммерции, Национальном Институте Стандартов и Технологии, финансовой компании J.P. Morgan & Co. Документы в этом формате можно просматривать не только на основных операционных системах Windows и MacOS, но и BeOs, Linux, Solaris.

3. PDF + JavaScript = Acrobat JavaScript

Формат **PDF** широко используется для представления текстовой и графической информации. Также **PDF** удобен и для научных статей, содержащих различные математические символы и формулы. Однако новые продукты фирмы Adobe предоставляют гораздо больше, чем просто средства подготовки и просмотра текстов. **PDF**-документы могут содержать различные элементы управления (текстовые поля, кнопки, списки и т.д.) чтобы фиксировать введённые пользователем данные, обрабатывать их и выдавать результаты. В зависимости от действий пользователя может изменяться внешний вид документа, его функциональность, т.е. **статический** документ становится **интерактивным** (здесь можно отметить, что по такому же пути развивались web-страницы). Фактически теперь **PDF-**документ является самостоятельной программой, не зависящей от платформы на которой она выполняется. Для реализации всех этих возможностей был использован также платформно-независимый язык – **JavaScript**.

JavaScript – мощный объектно-ориентированный язык web-сценариев, разработанный фирмой **Netscape Communications**, чтобы расширить функциональные возможности web-страниц. Первоначально

предназначенный для браузеров **Netscape**, **JavaScrip**t быстро развивался и вскоре стал широко используемым, универсальным языком программирования. В программе Adobe Acrobat 6.0 Professional, предназначенной для создания и редактирования **PDF**-документов используется язык **Acrobat JavaScript**, разработанный на основе JavaScript версии 1.5 по спецификации ISO-16262. **Acrobat JavaScript** отличается от языка **JavaScript**, используемого при создании web-страниц, некоторыми дополнительными возможностями для работы с **PDF**-документом. Соответственно в **Acrobat JavaScript** нет встроенных классов **JavaScript**, предназначенных для работы с html-документами. Тем не менее, новый язык содержит многие стандартные классы **JavaScript,** среди которых для нас наиболее важен класс **Math,** предоставляющий набор различных математических функций. Что касается синтаксиса и семантика, в этом отношении язык **Acrobat JavaScript** полностью идентичен своему прародителю.

4. Подробнее о Acrobat JavaScript

В реализации языка **JavaScript** для Acrobat отсутствуют объекты window, document и др., предназначенные для работы с окнами браузера. Вместо них создана собственная иерархия встроенных объектов для взаимодействия сценариев с **PDF**-документом. В этом разделе представлен краткий обзор наиболее важных из них, примеры использования при написании сценариев для **PDF**-документов.

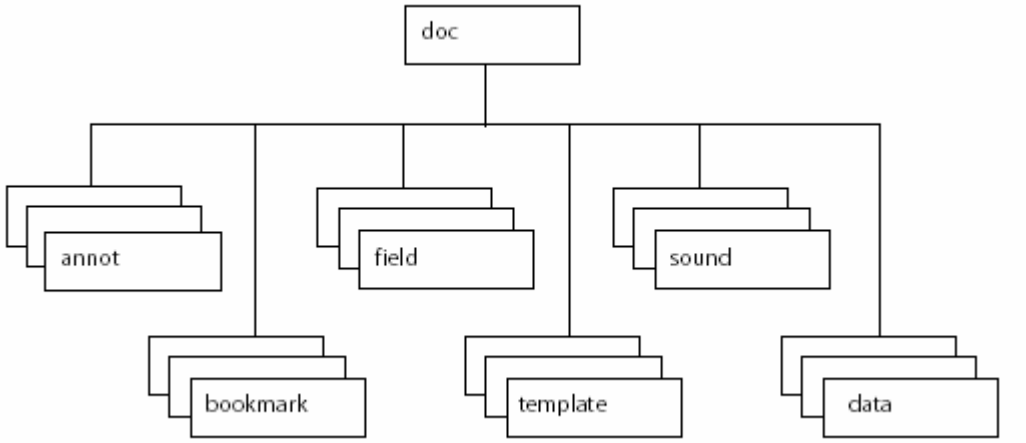
Объект **App**

App (от Application) – статический объект **Acrobat JavaScript**, который является представлением приложения Acrobat, используемого для просмотра документа. Объект **App** содержит множество методов и свойств, специфичных для Acrobat и различные сервисные подпрограммы. Используя этот объект, разработчик может получить доступ ко всем открытым в данный момент **PDF**-документам, а также к настройкам Acrobat, например, добавляя пункты меню, кнопки на панели инструментов и др. С помощью запросов объекта **App** можно узнать тип продукта **Adobe**, используемого для просмотра документа и его версию.

Объект **Doc**

Этот встроенный объект **Acrobat JavaScript** используется для представления самого PDF-документа. Объект **Doc** обеспечивает интерфейс между PDF-документом, открытом в программе просмотра, и интерпретатором JavaScript. Используя его свойства и методы, можно управлять всем содержимым PDF-документа и получать доступ к объектам, хранящимся внутри объекта Doc. Этими объектами могут быть поле (field),

закладка (bookmark), аннотация (annot), шаблон страницы (template) и др. Иерархия объектов, содержащихся в объекте Doc, приведена на рисунке ниже.



Следует заметить, что это иерархия содержания, а не иерархия наследования, т.е. объекты annot, field, и т.д. не наследуют свойства и методы объекта Doc, а лишь содержатся в нём. Из этих объектов остановимся лишь на одном наиболее важном – **field**.

Объект **field** представляет поле формы документа. Типы полей : кнопка (button), текстовое поле (text), список (listbox), и др. Для динамического создания полей используется метод Doc.addField(cName,cFieldType), где cName – имя нового поля, cFieldType – его тип. Доступ к полям документа можно получить с помощью метода Doc.getField(cName), cName – имя поля. Для удаления поля из формы необходимо вызвать метод Doc.removeField(cName), указав имя поля cName. Из свойств объекта field следует отметить одно наиболее часто используемое – value. Это свойство есть у всех объектов field всех типов, кроме button, оно содержит данные, введённые пользователем. В зависимости от типа поля свойство value может принадлежать классам String, Date или Number.

Подробное описание всех объектов **Acrobat JavaScript** можно найти в статье “Acrobat JavaScript Scripting Reference” на сайте фирмы **Adobe** http://partners.adobe.com.

5. Создание интерактивных PDF-документов

В этом разделе будет описана технология создания интерактивных **PDF**-документов с использованием языка **Acrobat JavaScript**.

Использование JavaScript в **PDF**-документах предоставляет разработчику следующие возможности :

- Выполнение вычислений

Можно создавать поля ввода в документе формата PDF для сбора числовых данных. Разработчик может определить последовательность вычислений, которые будут производиться над этими данными, чтобы получить нужные результаты.

- Проверка допустимости данных, введённых пользователем
  Разработчик может осуществлять контроль соответствия типов и диапазона введённых данных допустимым значениям с выдачей сообщений пользователю в случае ошибок.
- Реакция на действия пользователя
  PDF-документ может отвечать на различные действия пользователя, например, щелчки мышью, текстовый ввод и т.п. Разработчик может связать код **JavaScript** с такими событиями, и при их наступлении Acrobat запустит соответствующий сценарий.
- Динамическая модификация документа
  Этот пункт включает в себя :
  - Изменение пользовательского интерфейса
    В ответ на действия пользователя можно удалять и создавать новые поля, кнопки, пункты меню, изменять их свойства и связанные с ними действия.
  - Изменение содержимого **PDF**-документа
    Разработчик может динамически создавать новые страницы документа на основе шаблона. Шаблон – это проект страницы, которую можно добавить к существующему **PDF**-документу.
- Работа с базами данных
  Язык **Acrobat JavaScript** имеет встроенные объекты для получения доступа к базам данных. Эти объекты образуют интерфейс для работы с базами данных Acrobat Database Connectivity (ADBC). Разработчик может использовать команды SQL для поиска и модификации данных, хранящейся в базе.

В общем случае процесс создания интерактивного PDF-документа включает в себя следующие стадии :

1. Первоначальное создание **PDF**-файла

   Программа Adobe Acrobat позволяет преобразовывать в **PDF**-формат различных типов файлов : форматы Microsoft Office, AutoCAD, документы PostScript, текстовые файлы, HTML-страницы, различные графические форматы (JPEG, GIF, BMP и др.). Большинство научных публикаций подготавливается с использованием системы TeX, которая также допускает преобразование в **PDF**. Это весьма удобно для последующего создания электронных учебных пособий.

2. Редактирование статического содержимого

Кроме поддержки сложных форматов текста в Adobe Acrobat есть возможности встраивания в PDF-документ графических изображений, а также аудио- и видео- фрагментов.

3. Создание элементов управления

На этом этапе в документе с помощью визуальных средств создаются различные элементы форм : кнопки, текстовые поля, и т.п. Определяются их внешний вид и свойства.

4. Написание **JavaScript**-сценариев

Adobe Acrobat позволяет работать как с **JavaScript**-кодом, встроенным в сам **PDF**-документ, так и с внешними сценариями. Внешний уровень использования предполагает создание отдельных файлов с расширением .js, содержащих код. При открытии документа Acrobat находит и загружает все такие сценарии.

Внутренние сценарии делятся на **JavaScript**-код **уровня документа (**Document level JavaScripts**)** и код **уровня поля** формы **(**Field level JavaScripts**).** На первом уровне описываются глобальные переменные, подпрограммы и структуры данных, доступные по всему документу. Ко второму уровню относятся обработчики событий, связанные с элементами форм. Например, при нажатии пользователем кнопки Acrobat запускает связанный с ним скрипт, который считывает данные из полей формы, производит вычисления и выдаёт результат. Исходный текст обработчика может выглядеть так :

```
// считывание данных полей с именами “field_name_1” и “field_name_2”
        var Var1 = this.getField(“field_name_1”);
        var Var2 = this.getField(“field_name_2”);
// помещение результата сложения  в поле с именем “field_name_3”
        this.getField(“field_name_3”).value = Var1 + Var2;
```

Программа Adobe Acrobat предоставляет мощную среду для создания интерактивных **PDF**-документов. **JavaScript**-редактор позволяет создавать и редактировать обработчики событий и сценарии уровня документа. Предусмотрены также средства отладки **JavaScript**-кода с возможностью пошагового выполнения, просмотра значений переменных, установки контрольных точек.

6. Демонстрационный пример

Практическая часть данной работы включает в себя разработку интерактивного учебного пособия по курсу “Численные методы”. Пособие

представляет собой программную реализацию алгоритмов исключения Гаусса и Жордана для решения систем линейных уравнений и обращения матриц. Пример выполнен в виде **PDF**-документа созданного с помощью программного продукта фирмы **Adobe** – Acrobat 6.0 Professional.Для открытия файла можно использовать программу Adobe Reader версии 5.0 или более поздних.

Все алгоритмы реализованы на языке **Acrobat JavaScript**. Для каждого алгоритма предусмотрены следующие возможности :

- Заполнение матрицы случайными числами
- Ввод матрицы пользователем с клавиатуры
- Выполнение алгоритма целиком, либо в пошаговом режиме
- Вычисление обратной матрицы
- Решение системы линейных алгебраических уравнений

В пособии реализованы следующие алгоритмы :

1. LU–разложение по методу Гаусса
2. LU–разложение по методу Гаусса (по строкам)
3. LU–разложение по компактной схеме Краута
4. LU–разложение по компактной схеме “строка за строкой”
5. LU^{-1} –разложение по методу Жордана

Исходный текст скриптов (с сокращениями) приведен ниже.

```
// Класс Matrix////////////////////////////////////////////////
// Конструктор
function Matrix(dim)
{
    this.dim = dim;
    this.m = new Array(this.dim);
    for(var i=0;i<this.dim;i++) this.m[i] = new Array(this.dim);

    this.SolveLU_ = SolveLU_;
    this.SolveL_U = SolveL_U;
    this.SolveLU_inv = SolveLU_inv;
    this.InvertLU_ = InvertLU_;
    this.InvertL_U = InvertL_U;
    this.InvertLU_inv = InvertLU_inv;
    this.SetCol = SetCol;
    this.GaussLU_ = GaussLU_;
    this.GaussL_U = GaussL_U;
    this.GaussLU_ByRows = GaussLU_ByRows;
    this.KrautLU_ = KrautLU_;
    this.KrautL_U = KrautL_U;
    this.RowByRowLU_ = RowByRowLU_;
    this.JordanLU_inv = JordanLU_inv;
```

```
}
// Решение СЛУ при различных видах LU-разложения
function SolveLU_(b)
{
    var x = new Vector(this.dim);
    var y = new Vector(this.dim);
    var val = 0.;
//прямой ход
    for(var i=0;i<this.dim;++i)
    {
        val = 0.;
        for(var j=0;j<i;++j) val+=this.m[i][j]*y.v[j];
        y.v[i] = (b.v[i] - val)/this.m[i][i];
    };
//обратный ход
    for(i=this.dim-1;i>=0;--i)
    {
        val = 0.;

          for(j=i+1;j<this.dim;++j) val+=this.m[i][j]*x.v[j];
          x.v[i] = y.v[i] - val;
    };
    return x;
}
function SolveL_U(b)
{
    var x = new Vector(this.dim);
    var y = new Vector(this.dim);
    var val = 0.;
//прямой ход
    for(var i=0;i<this.dim;++i)
    {
        val = 0.;
        for(var j=0;j<i;++j) val+=this.m[i][j]*y.v[j];
        y.v[i] = b.v[i] - val;
    };
//обратный ход
    for(i=this.dim-1;i>=0;--i)
    {
        val = 0.;
        for(var j=i+1;j<this.dim;++j) val+=this.m[i][j]*x.v[j];
        x.v[i] = (y.v[i] - val)/this.m[i][i];
    };
    return x;
}
function SolveLU_inv(b)
{
    var x = new Vector(this.dim);
    var y = new Vector(this.dim);
    var val = 0.;
//прямой ход
    for(var i=0;i<this.dim;++i)
    {
```

```
        val = 0.;
        for(var j=0;j<i;++j) val+=this.m[i][j]*y.v[j];
        y.v[i] = (b.v[i] - val)/this.m[i][i];
    };
//перемножение U^-1 и y
    for(i=0;i<this.dim;i++)
    {
        val = y.v[i];
        for(j=i+1;j<this.dim;j++) val+=this.m[i][j]*y.v[j];
        x.v[i] = val;
    };
    return x;
}
//Обращение матрицы при различных видах разложения
function InvertLU_()
{
    var b = new Vector(this.dim);
    var x = new Vector(this.dim);
    var invm = new Matrix(this.dim);
    b.NullGen();
    for(var j=0;j<this.dim;j++)
    {
        b.v[j] = 1;
        x = this.SolveLU_(b);
        invm.SetCol(j,x);
        b.v[j] = 0;
    };
    return invm;
}
function InvertL_U()
{
    var b = new Vector(this.dim);
    var x = new Vector(this.dim);
    var invm = new Matrix(this.dim);
    b.NullGen();
    for(var j=0;j<this.dim;j++)
    {
        b.v[j] = 1;
        x = this.SolveL_U(b);
        invm.SetCol(j,x);
        b.v[j] = 0;
    };
    return invm;
}
function InvertLU_inv()
{
    var b = new Vector(this.dim);
    var x = new Vector(this.dim);
    var invm = new Matrix(this.dim);
    b.NullGen();
    for(var j=0;j<this.dim;j++)
    {
        b.v[j] = 1;
```

```
        x = this.SolveLU_inv(b);
        invm.SetCol(j,x);
        b.v[j] = 0;
    };
    return invm;
}
function SetCol(k,col)
{
    for(var i=0;i<this.dim;i++) this.m[i][k] = col.v[i];
}
//Разложение метода Гаусса
function GaussLU_()
{
    for(var k=0;k<this.dim;k++)
    {
         if(this.m[k][k]==0) return false;
         for(var j=k+1;j<this.dim;j++)
             this.m[k][j]/=this.m[k][k];
         for(var i=k+1;i<this.dim;i++)
             for(j=k+1;j<this.dim;j++)
                 this.m[i][j]=this.m[i][k]*this.m[k][j];
     };
    return true;
}
function GaussL_U()
{
     for(var k=0;k<4;k++)
     {
     if(this.m[k][k]==0) return false;
     for(var i=k+1;i<4;i++) this.m[i][k]/=this.m[k][k];
     for(i=k+1;i<4;i++)
          for(var j=k+1;j<4;j++)
               this.m[i][j]-=this.m[i][k]*this.m[k][j];
};
    return true;
}
function GaussLU_ByRows()
{
    for(var k=0;k<this.dim;k++)
    {
        for(var i=0;i<k;i++)
            for(var j=i+1;j<this.dim;j++)
                this.m[k][j]-=this.m[k][i]*this.m[i][j];
        if(this.m[k][k]==0) return false;
        for(j=k+1;j<this.dim;j++) this.m[k][j]/=this.m[k][k];
    };
    return true;
}
// Разложение по компактной схеме Краута
function KrautLU_()
{
    var val;
    for(var k=0;k<this.dim;k++)
```

```
    {
        for(var i=k;i<this.dim;i++)
        {
            val = 0.;
            for(var j=0;j<k;j++) val+=this.m[i][j]*this.m[j][k];
            this.m[i][k]-=val;
        };
        if(this.m[k][k]==0) return false;
        for(j=k+1;j<this.dim;j++)
        {
            val = 0.;
            for(i=0;i<k;i++) val+=this.m[k][i]*this.m[i][j];
            this.m[k][j]=(this.m[k][j] - val)/this.m[k][k];
        };
    };
    return true;
}
function KrautL_U()
{
    var val;
    for(var k=0;k<this.dim;k++)
    {
        for(var j=k;j<this.dim;j++)
        {
            val = 0.;
            for(var i=0;i<k;i++) val+=this.m[k][i]*this.m[i][j];
            this.m[k][j]-=val;
        };
        if(this.m[k][k]==0) return false;
        for(var i=k+1;i<this.dim;i++)
        {
            val = 0;
            for(j=0;j<k;j++) val+=this.m[i][j]*this.m[j][k];
            this.m[i][k]-=val;
            this.m[i][k]/=this.m[k][k];
        };
    };
    return true;
}
// Разложение по компактной схеме “строка за строкой”
function RowByRowLU_()
{
    var val;
    for(var k=0;k<this.dim;k++)
    {
        for(var j=0;j<=k;++j)
        {
            val = 0.;
            for(var i=0;i<j;i++) val+=this.m[k][i]*this.m[i][j];
            this.m[k][j]-=val;
        };
        for(j=k+1;j<this.dim;++j)
        {
```

```
            val = 0;
            for(var i=0;i<k;i++) val+=this.m[k][i]*this.m[i][j];
            this.m[k][j]-=val;
        };
        if(this.m[k][k] == 0) return false;
        for(j=i+1;j<this.dim;++j) this.m[k][j]/=this.m[k][k];
    };
    return true;
}
//Разложение по методу Жордана
function JordanLU_inv()
{
    for(var k=0;k<this.dim;k++)
    {
        if(this.m[k][k]==0) return false;
        for(var j=k+1;j<this.dim;j++)
            this.m[k][j]/=this.m[k][k];
        for(var i=0;i<k;i++)
            for(j=k+1;j<this.dim;j++)
                this.m[i][j]-=this.m[i][k]*this.m[k][j];
        for(i=k+1;i<this.dim;i++)
            for(j=k+1;j<this.dim;j++)
                this.m[i][j]-=this.m[i][k]*this.m[k][j];
    };
    for(i=0;i<this.dim;i++)
        for(j=i+1;j<this.dim;j++) this.m[i][j] = -this.m[i][j];
    return true;
}
```

7. Заключение

В конце следует отметить, что PDF-формат является не только удобным средством подготовки электронных публикаций, но и мощным инструментом создания интерактивных документов. Использование его в преподавании для разработки учебных пособий весьма полезно. Возможности, предоставляемые для этого языком Acrobat JavaScript, использованы далеко не в полном объёме.

Список литературы

1. “Acrobat JavaScript Scripting Guide”, technical note #5430, May 2003 – руководство по использованию языка **Acrobat JavaScript,** взято с сайта фирмы Adobe http://partners.adobe.com
2. “Acrobat JavaScript Scripting Reference”, technical note #5431, February 2004 – полное описание встроенных объектов языка **Acrobat JavaScript**, взято с сайта фирмы Adobe http://partners.adobe.com
3. И.В. Семушин, Г.Ю. Куликов, Лабораторный практикум по курсу “Вычислительная линейная алгебра” – Ульяновск, 1996 – использовано при создании демонстрационного примера

Содержание