Publication of Interactive Visualizations with JavaView

Konrad Polthier, Samy Khadem, Eike Preuß and Ulrich Reitebuch

Technische Universität, Berlin

Abstract. JavaView is a 3D geometry viewer and a numerical software library written in Java which allows one to publish interactive geometries and mathematical experiments in online web pages. Its numerical software library provides solutions and tools for problems in differential geometry and mathematical visualization. This allows the creation of one's own geometric experiments, while always profiting from the advanced visualization capabilities and the web integration. JavaView easily integrates with third-party software like Mathematica and Maple, and enables direct publication of experimental results online.

1 Introduction

JavaView [18] is a software package for doing geometry, numerical experiments and scientific visualization online in a web browser as well as in a local application. Students, teachers and researchers can use JavaView to create and add experiments to electronic research publications or to distant learning environments. The future of mathematical communication and publication is strongly related to the internet, and JavaView is a tool to enhance classical textual descriptions not only with images and videos but additionally with interactive geometries and online mathematical experiments.

JavaView consists of a 3D geometry viewer and a numerical software library written in Java. It allows one to add interactive 3D geometry to any HTML document and to perform experiments and modelling online. JavaView has been developed to solve the following tasks:

1. Visualization of 3D geometries and numerical experiments online.
2. Publication of mathematical experiments in online electronic journals.
3. Development of visualization and numerical algorithms in an open class library
4. Specific algorithms and file formats to prepare geometry models for online publication
5. Smooth integration into third-party software via the JavaView API.

The first version of JavaView was released in November 1999 after development versions had been used in research projects at the Technische Universität Berlin for over a year. JavaView is now used and extended at different places world-wide. There exist a number of mathematical

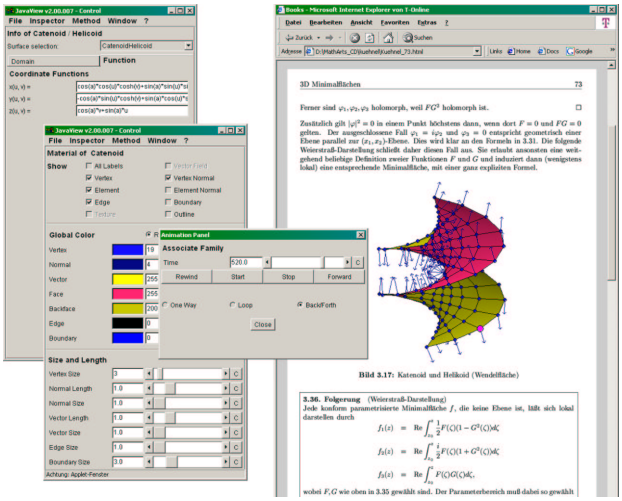

Fig. 1. Classic textbook on differential geometry enhanced with an interactive applet. On first sight, the applet appears like a static image. The interactivity is hidden but available upon the user's choosing.

demonstrations which show the range of new applications possible with web-based experimental software.

The simplest use of JavaView is the online display of interactive geometry models, while a more complicated applet may be a full mathematical experiment solving a variational problem. JavaView provides many tools and algorithms accessible from menus and dialogs without programming, as well as elaborate Java class libraries with an open API for programming custom mathematical experiments.

The frontend of JavaView is a 3D viewer with a number of control panels to inspect a geometry, modify material properties, change the camera and display settings, or drive an animation. A user has different options, as listed below, to use JavaView where each requires different levels of knowledge of JavaView or programming experience. JavaView comes with a set of applets to show a precomputed geometry model or do a number of mathematical experiments, like solving an ordinary differential equation.

- Use applets and experiments included in JavaView or written by third-party authors.
- Adjust existing applets, for example, to view one's own precomputed geometry model.
- Use sophisticated JavaView tools to analyse and modify geometry models online.
- Develop one's own experiments in Java using the open API of the class library of JavaView.
- Attach or integrate the application version of JavaView to one's own software, to Mathematica, or to Maple.

JavaView operates on 2D, 3D, and higher dimensional geometries, and it implements distinguished algorithms from discrete differential geometry [19][20]. It offers a full web integration and an operating system independence, and, in this sense, compares to the dynamic elementary geometry software programs Cabri [13] and Cinderella [11]. Both of these tools are also discussed in this book. JavaView's visualization tools and many of its algorithms compare to those found in visualization programs like GeomView [17], Grape [22], Oorange [4], AVS and others. The major difference to these visualization programs, and one of the initial reasons for the development of JavaView, is the possibility to publish interactive experiments written in JavaView directly in online research articles, books, or educational course notes.

In this article we introduce some properties of JavaView and give a number of examples of smooth integration of interactive experiments and visualizations in publications. An online version of this article is available at `http://www.javaview.de/` and includes interactive versions of the presented applets.

2 Components of JavaView

The most visible component of JavaView is the 3D display included in applets on web pages or appearing when JavaView is launched as an application. By default, the other functions and user interface elements are hidden from a first time user giving the impression that JavaView mainly provides interactive images. In fact, JavaView provides the functionality of displaying static in-text images along with fully interactive documents.

In each display, the right mouse button shows a popup menu to select different interaction modes for 3D-transformations, picking, or launching the control panel. The control panel of JavaView is the central place providing a variable set of additional panels and menus to analyse and modify the displayed data as well as to select and drive a complicated experiment. For example, each experiment in JavaView has a panel specific to the experiment to modify parameters and to drive the experiment.

A number of powerful tools, so-called workshops, are available from the menu bar to operate on the current geometry in the display. The functionality of these workshops ranges from the simple inversion of the surface orientation to the curvature-based simplification of large geometries.

2.1 Display Window

The display of JavaView provides most features of an advanced 3D viewer as well as a variety of interaction tools to pick and modify the displayed geometries.

Among these features are:

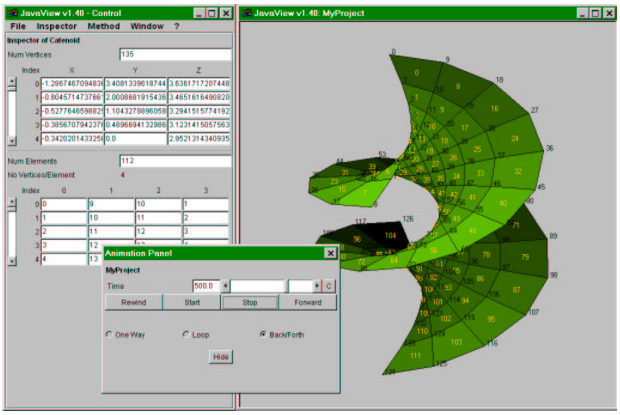

Fig. 2. JavaView display with inspector and animation panel.

- Rotation, translation, zoom, space ball camera control
- Picking and interactive modification of geometries
- Display of vertices, edges, faces, vector fields with variable sizes and colors
- Animations, keyframe interpolation, auto-rotation
- Textured surfaces, z-buffered display, depth cueing
- Sending camera and pick events
- Coordinate axes and rulers
- Perspective, orthogonal and non-euclidean camera projections
- Import and export of geometries in multiple geometry file formats
- PostScript and image file export for inclusion into TEX and paper publications
- Frontend for other applications, for example, to view *Mathematica* or *Maple* graphics

The JavaView display naturally has a drawback in rendering speed against native graphics libraries like OpenGL and DirectX. This speed penalty is mainly located at the rendering stage since the current browsers do not support native 3D rendering. Therefore, in situations where the real-time visualization of really large data sets with millions of triangles is required JavaView offers a simple export to third-party programs and allows the integration of native viewers. In general, the execution speed of Java code and of the algorithms implemented in JavaView heavily depends on the type of the virtual machine. Modern just-in-time compilers compile the Java byte during the loading process and dramatically reduce the speed penalty against native methods.

2.2 Workshops and Projects

A *project* in JavaView is a full-fledged application and very similar to a Java applet. For example, a project often provides the setup of a mathematical experiment including different Java classes, panels and dialogs,

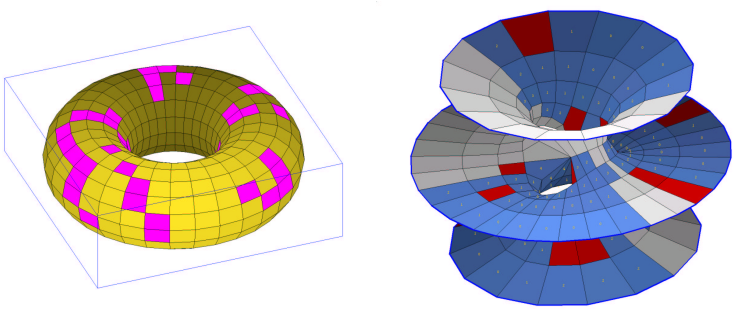

Fig. 3. Mathematical games on 3d-surfaces for educational purpose. Left: Game of Life. Right: Game *Minesweeper* on a minimal surface of Costa.

and Html descriptions. The major difference between a JavaView project and a Java applet is the fact that, a project is considered a module which is not only invoked by an applet but may be invoked by other projects too. This module character is hardly accomplished by using applets since an applet derives from the specific class *java.awt.applet.Applet* which is solely designed to run inside an Html page.

JavaView projects provide a more flexible functionality by deriving from the separate superclass *jv.project.PjProject*. For example, an instance of the project class may be created in an applet and configured according to special needs. This allows one to create instances of the same project class in different applets, each time the instance is configured differently. Each project has a well-defined interface for its configuration.

For programmers, projects simplify the integration of modules with the JavaView environment by providing an easy access to display windows, animation support and handling of display and camera listeners. Further, projects may be registered in the viewer manager and invoked from a menu during runtime.

Projects are heavy-weight concepts when considering their functionality while a *workshop* is a light-weight class basically consisting of a set of methods and a dialog for configuration. For example, the geodesic surveyor to measure geodesic distances on polyhedral surfaces is a workshop consisting of sophisticated algorithms and a dialog, say, for entering the two endpoints of a geodesic. Basically, the geometry is given to the workshop, then the workshop optionally opens a dialog or immediately starts its operation, and finally, the analysed or modified geometry is returned. A workshop is like a service station for a car, you pass the car to the station and get it back, hopefully renewed.

Workshops are either invoked from the method menu of the JavaView control panel to analyse or modify geometries, and to perform mathematical algorithms, or the workshop classes may be called by a programmer. Note that, workshops do not derive from a specific superclass nor do they implement a certain interface. They are a theoretical concept to enclose functionality.

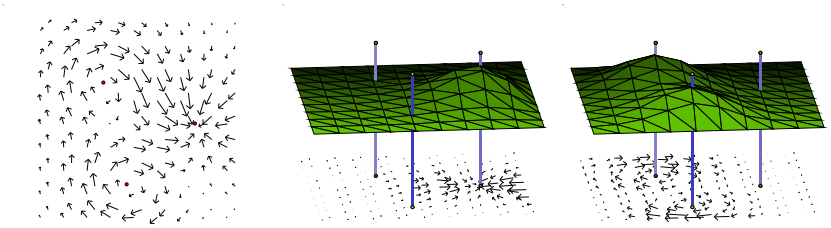

Fig. 4. Hodge-Helmholtz decomposition of a vector field for feature extraction. Left: original field, right: rotation and divergence free components with potentials.

2.3 Class Libraries and JavaView Archives

JavaView is delivered as a set of Java archives which simply are compressed collections of Java classes. The libraries are well documented, thereby allowing the development of one's own applets and applications based on the high-level classes and methods available in JavaView. The user's guide provides an introduction to programming in Java and using the classes of JavaView. The reference documentation provides detailed comments for all classes and methods. It was automatically generated from the JavaView source code with the *javadoc* utility [7].

The most important archive is *javaview.jar* which contains the basic packages for the 3D display, the geometry classes, a selection of the import and export loaders, the linear algebra classes, and some other sub packages. This archive must be available to all programmers. The optional archive *jvx.jar* provides extended geometry classes, a number of powerful workshop classes, and additional loaders. The optional archive *vgpapp.jar* contains a set of applications of JavaView and introductory tutorials for programmers. For details, we refer to the online reference documentation and the user's guide of JavaView.

It is possible to generate an individual archive for each application of JavaView which contains only those class files referenced and needed in the applet. This would reduce the size of the required JavaView classes even further. For example, many applets do not require some loaders to parse special geometry files and will perform correctly even if these loaders are not available in the Java archive.

Nevertheless, in general a programmer is not advised to create new archives and we do not recommend recombining any of the JavaView archives. The most important reason is that the application will hardly be maintainable when future revisions and extensions are made to JavaView. Any time new JavaView archives are available the application programmer must rebuild his special purpose archives.

On the other hand, the JavaView distribution contains a special purpose archive, namely, the *jvLite.zip* archive which is drastically optimized for fast download. Its small size of less than 100kb is comparable to a few gif images which appear frequently as beautifications on web pages.

The tiny lite version is about 20% of the size of the original archive javaview.jar. jvLite.zip was automatically generated using the JAX tool [6] by a sophisticated analysis of the Java byte code and by removal of all user interface classes from JavaView, i.e. panels, dialogs and menus. Therefore, the jvLite package is ideal for the efficient inclusion of precomputed geometry in interactive images respectively applets in online documents. Note, this lite version of JavaView is not useful for programming purposes since all class, method and instance names are obfuscated to very small names for data reduction. Further, many public methods, which are unused for viewing of precomputed geometries, were automatically removed.

Note, a browser must only download once either of the JavaView archives since it keeps the archive in its cache until it encounters a newer version of the JavaView archive.

2.4 JVX Geometry File Format

JavaView uses its own JVX file format for temporary and long term archiving of singles geometries and full-featured scenes. In contrast to the many other file formats supported by JavaView, the JVX format is XML-based. XML is the new language for a diversity of data types used in the internet. The specific format of JVX geometry files is specified in the document type definition (DTD) *jvx.dtd*. The existence of a DTD allows an automatic validation of the syntax and partially the semantics of every given JVX data file.

For example, the JVX format is used as one of the master file formats in the EG-Models server to a large extent because of its validation ability. The JVX format is also used to exchange data between JavaView and third-party software like Maple and Polymake. Each JVX geometry file may optionally be equipped with author information and detailed information about the mathematical properties of the model. This includes free-form keywords and MSC classifications.

Currently, the datafiles of JavaView use the following extensions where each of the formats is XML based and has an accompanying DTD:

JVX geometry file format for individual geometries and scenes with multiple geometries.
JVR configuration data of JavaView itself— like language support, window positions and menu contents.
JVD display and camera settings.

3 Creating One's Own Applets

3.1 A Hands-On Example

This hands-on example describes the necessary three steps to add the JavaView display to a web page. Here we show a precomputed geometry model in a JavaView window allowing interactive modifications.

1. Download the archive *javaview.jar* from the JavaView homepage.
2. Type the following web document *myPage.html* referring to a geometry model *brezel.obj*.
3. Upload all three files to a web server.

The sample applet tag inside the document *myPage.html* looks as follows:

```
<html><body><p>This is a JavaView applet on a web page.</p>
     <applet
          code=javaview.class
          archive=''javaview.jar''
          width=200 height=200>
          <param name=model value=''brezel.obj''>
     </applet>
</body></html>
```

This applet visualizes the geometry model inside a small window of 200*200 pixels on the web page. Note, in this example all three files reside in the same directory. The geometry model must be a file on the server which hosts the applet because of security restrictions in Java which forces each applet to run in a so-called sandbox. If JavaView runs as an application on a local machine outside a web browser then a geometry file from any url on the web may be launched. Applications do not run in a sandbox and have free access to system resources.

This example stresses the fact that the installation of the JavaView software is no longer an issue compared to the installation process of other software. The browser downloads the required Java archive when it encounters the archive parameter inside the applet tag. The browser also ensures that the archive is downloaded for the first request only, and later reuses the version it has stored in the browser's cache.

4 Sample Applications of JavaView

In this section we demonstrate a range of possible applications of JavaView on a number of existing examples. The following examples describe different aspects of the use of JavaView, among which are:

1. Visualization and evaluation of precomputed models which are stored somewhere on the internet.
2. Interactive tutorials explaining simple numeric or geometric facts to accompany classical lectures or online workshops.
3. Sophisticated numerical research projects which combine numerics and advanced visualization.
4. Joint research of authors at different universities doing numerical research experiments embedded into web pages.

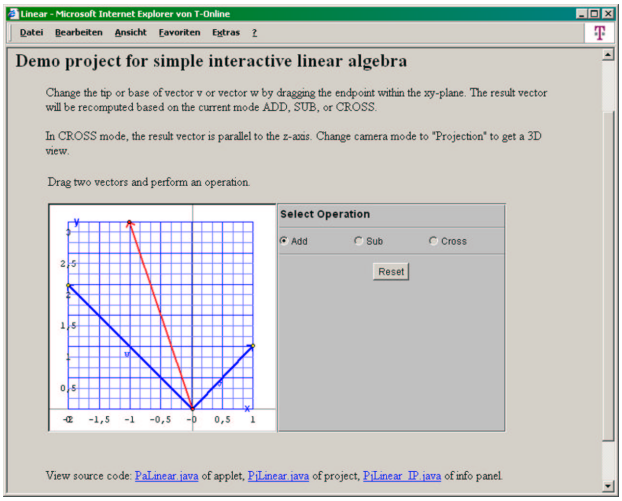

Fig. 5. Basic linear algebra applet created with 30 lines of new Java code.

4.1 Web-based Courses: Linear Algebra

Nowadays multimedia-enhanced undergraduate courses are developed and used at different places world-wide. For example, the interactive linear algebra course ILAW is one of the first successful projects, see the article [1] in this book. Nevertheless, there are still a number of technical barriers to overcome, for example, at the present time even rendering Mathematics in web pages is not settled in a satisfactory state. The MathML specification is already established by the World-Wide-Web consortium but still the major browsers do not support its rendering. In contrast, the technical problems for the inclusion of interactive experiments and demonstrations in online web courses do not provide conceptual problems and are presently solved apart from their rather large development cost.

A Java applet often is similar to a full-featured software program with a user interface, user documentation, and with a large programming effort during its development. JavaView is designed to simplify the development and maintenance of interactive applets as browsers develop and as operating systems change. Therefore, it basically reduces the development time of a new applet to the time needed for the new functionality rather than for the technical setup. For example, the use of JavaView's high level classes reduces the size of the simple applet shown in figure 5 to about thirty lines of new Java code.

4.2 Distant Learning: Interactive Experiments

Distant learning courses have been around for a long time as supplements to the traditional educational system and to training courses in

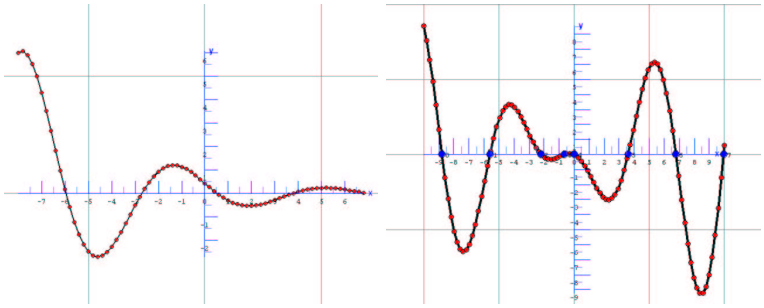

Fig. 6. The solver for ordinary differential equations (left) and the root finder applet (right) are online services.

industry. Distant learning projects must have well prepared course materials since there is less direct contact with students. These projects will be among the first to include interactive experiments, and may be even the driving forces for the development of whole packages of interactive online experiments.

For example, nowadays one would expect online courses to be much more interactive than classic textbooks on mathematics. In numerics, different numerical methods for solving an ordinary differential equation may be compared by a reader online following instructions in the written text. In geometry, the curvature of surfaces may be interactively investigated. In algebra, eigenvalues of a matrix may be interactively computed and their eigenvectors displayed in an applet integrated in the text.

The applets in figure 6 demonstrate the numerical solution of an ordinary differential equation (left) and the search for zeros of a user defined function (right). The root finder method subdivides the original interval and uses Brent's method to find the roots on each subinterval. JavaView has Java implementations of several algorithms of the Numerical Recipes library [21]. In all these examples, the user is assumed to follow a description of the algorithm while simultaneously studying the online example.

4.3 Numerics Online: Energy Minimizer

Java is a full-featured programming language allowing efficient development of numerics algorithms and visualization tools. Although it has some structural speed limitations, since by design it is an interpreted language, the language itself is based on modern programming paradigms. But this drawback is negligible in many applications since modern just-in-time compilers are able to provide an efficient compilation process on the fly while loading a Java program. Currently, Java may not be a language of choice for high-performance computing but it is already used in a large section of numerics. Among the two great benefits of Java are, first, its machine and operating system independence, and, equally

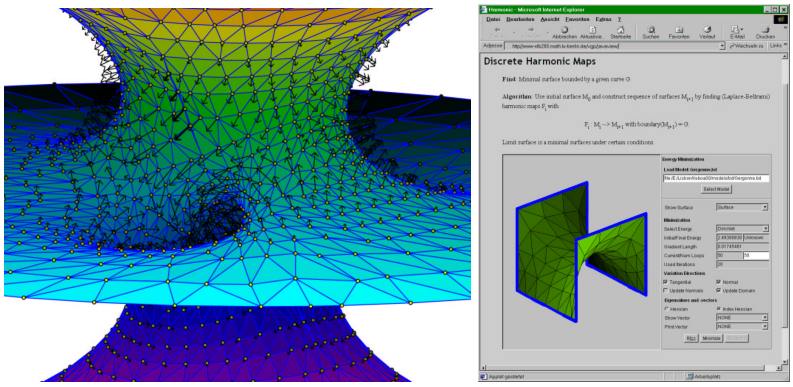

Fig. 7. First eigenfunction of the second variation of surface area (left) and applet for energy minimization (right).

important, the possibility to run interactive experiments online in web browsers.

Research applications like the eigenvalue computations in figure 7 demonstrate that numerical computations and scientific visualization are efficiently performed with JavaView. Further, the coherent interface of Java applets immediately makes these experiments accessible from locations world-wide as well as their inclusion in digital research publications.

4.4 Online Mathematics Services: Geodesic Surveyor

Nowadays, software basically relies on the paradigm of a local installation, but at the horizon we see the software industry already developing server based software. There is important potential in application service providers which offer computation and software resources over the internet, thereby freeing a user from doing a local installation and, even worse, keeping the local version updated over time. For example, Mathematica will soon release the product webMathematica which allows one to use a server based Mathematica kernel over the internet from a remote location.

The JavaView web site already hosts a few online services for doing mathematics. Some services like the AlgebraicSolver and the MathTyper invoke server side computations requiring a user to be online, while other services like the OdeSolver, RootFinder and GeodesicSurveyor are built into JavaView and rely purely on client side computations. All these services make use of the JavaView class library. The application programming interface of the class library is open and well-documented, enabling a rapid integration of third-party tools with the JavaView display and user interface.

For example, the *Algebraic Solver* computes algebraic surfaces in $\mathbb{R}^3$ based on user input. The user enters an equation, specifies various parameters, and starts the evaluation. The computation is done on the

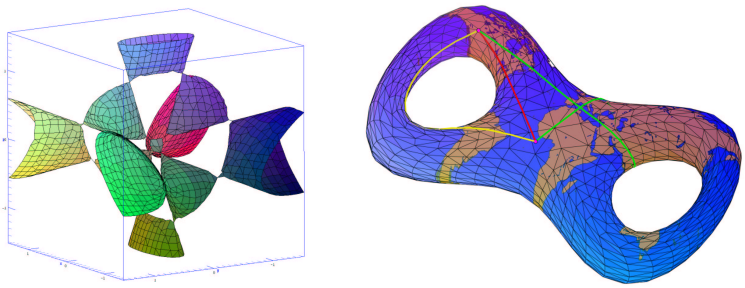

Fig. 8. Mathematical Online Services. Left: Algebraic Surface Solver, Right: Geodesic Surveyor.

web server which returns the computed surface to the user's JavaView applet. The surface is then easily studied and post-processed with standard JavaView tools, for example, the fine mesh is coarsened or artifacts removed. This service uses the server version of the Liverpool Surface Modelling Package [15] by Richard Morris with JavaView attached as interface for user input and frontend for 3D display of the computed surfaces.

The *Geodesic Surveyor* uses built-in methods of the extension package jvx.jar of JavaView to study discrete geodesics on any piecewise linear surface. The applet has two modes: either it computes locally shortest curves that connect two given points on the surface, or it computes the straightest curve which starts from a given point in a given tangential direction along the surface. Both methods work solely on the user's computer in contrast to the previous service on algebraic surfaces.

4.5 Archiving Interactive Documents in a Library

The project *Dissertation Online* funded by the Deutsche Forschungsgemeinschaft (DFG) is a cooperation of the Humbolt-Universität Berlin, Gerhard-Mercator-Universität Duisburg, Universität Erlangen-Nürnberg, Carl von Ossietzky-Universität Oldenburg, and of the main libraries in Germany. The main task of the project is the specification of criteria and guidelines for the digital publication of dissertations in Germany.

JavaView was selected by the project *Dissertation Online* of the German science foundation DFG to produce a reference online dissertation `http://www.javaview.de/applications/dissOnline.html` in mathematics including interactive visualizations and experiments. Figure 9 shows two sample pages of the dissertation of T. Hoffmann, which was enhanced with interactive experiments using JavaView and jDvi, see the article [5] in this book.

The category *dissertation* serves as a good playground for testing various technical issues related with the publication and archiving of

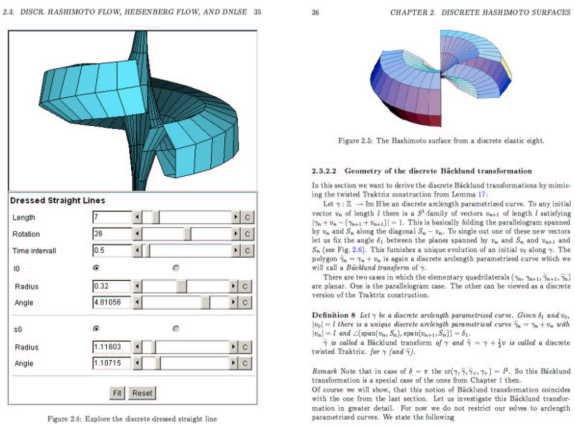

Fig. 9. Online dissertation with interactive JavaView applets.

scientific research documents. Among the many open problems of the project are:

- Different document structures for different scientific disciplines.
- Designated meta information for retrieval.
- Copyright and other legal aspects.
- Authentication and unchangeableness must be guaranteed.
- Inclusion of multimedia capabilities in digital publications

JavaView enables the inclusion of interactive experiments and visualizations in online documents while it respects the special needs of libraries. An important aspect for libraries are the maintenance costs for providing documents with interactive experiments online on a server. For example, the classic distribution of software is the delivery of binary programs. These binaries heavily depend on the operating system and the binary format often changes between operating system versions. A library usually cannot afford to maintain such a software. Therefore, such a dissertation would require a link to the original author for supplying new versions - a need which is even worse in the eyes of a librarian.

The use of Java applets completely simplifies the archiving and maintenance problems of multimedia documents:

1. The author of a dissertation uses a Java archive file, say, the JavaView archive file for display of 3D geometries, or he writes interactive experiments directly in Java. Then he references the Java based experiments in his article, say, by including an applet tag in an Html document which refers to the archive file. Now the full interactive online publication consists of the text documents, possibly some images and videos, and the Java archive files required for the

experiments. All these documents are bundled by the author and submitted to the library for archiving.
2. The library puts all documents of the publication in a web directory and provides public access. Note that beside indexing and archiving of the documents, the library has no additional task or maintenance duty, compared to non-multimedia documents. In particular, the library need not provide any special software for running the experiments.
3. A user of the library accesses the online publication with a web browser. By default, the web browser is Java enabled and therefore able to run the experiments. The browser automatically downloads the text documents as well as the Java archives referenced in the documents, and provides interactive access to the experiments. Note that even the user has no additional task beside finding the document. In fact, the user need not install any special software beside the web browser.

Summarizing, we have the following responsibilities for interactive experiments in publications:

Author: Create Java based experiment and upload
Library: Provide web space, no software installation
Reader: Java enabled web browser.

First, the author can store all experiments compactly in a Java archive file, which is independent of the operating system, and, second, the web browser of the user comes with a full Java installation by default. The library just stores the digital documents without any need of maintaining software packages required by publications.

The main reason for the simple deployment of Java programs lies in the fact that the operating system is responsible for providing a virtual machine in which Java programs run. Since the virtual machine is system dependent, any Java program can be system independent.

4.6 Making Textbooks Interactive

A large number of classical textbooks for students are available in the market as print versions. Often it requires minor efforts to create an interactive online version if one replaces static images and diagrams with interactive Java applets. Figure 1 shows a JavaView applet inside an online version of the textbook *Introduction to Differential Geometry* by Wolfgang Kühnel [12]. Here the static image of a helicoid in the print version of the book was replaced with a JavaView applet. Not only does the applet allow the reader to view the geometry from different perspectives, it also allows one to show the whole transformation of the helicoid to the catenoid as an animation. The book *Algebra Interactive* by Arjeh Cohen et. al. [2] is among the first convincing examples of this new generation of educational resources.

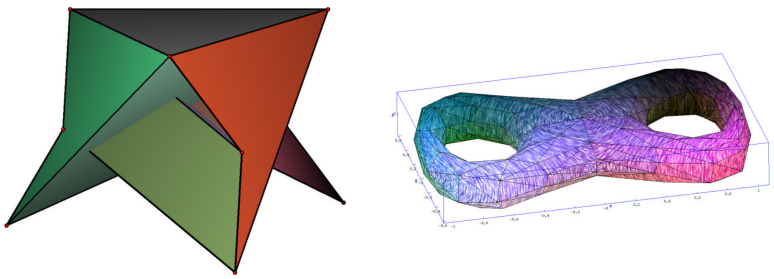

Fig. 10. The Boy surface is an immersion of the projective plane (left, polyhedral model by U. Brehm). Line integral convolution for flow visualization.

An applet may include exercises for students, or explain aspects of mathematical concepts. For example, the above transformation shows the associate family of minimal surfaces where all surfaces are minimal and isometric, and further, the surface normal vector at each vertex remains constant during the transformation. This latter property is easily viewed by students with default features of JavaView:

- Open the material panel and enable display of vertex normals (press 'ctrl-m' in the display)
- Optionally, mark a few vertices of the surface (while 'm' is pressed drag a small rectangle in the display)
- Optionally, color, length, and thickness of normal vectors may be adjusted in the material panel.
- Open the animation panel from menu Windows->Animation or by pressing 'ctrl-a', and start the transformation.

The mathematical properties of normal vectors are intuitively demonstrated by following the marked vertices during the animation.

Note that it is a smooth transition from a static classical book to a first version of an interactive book. Enhancing classic books with useful multimedia features requires much less technology on the authoring side than is widely assumed. Also, there is no need for a technological revolution on the reader's side since at first sight the multimedia book will look the same as a print version of the same book. The revolution is of a technical nature and resides inside the online software hidden to the reader, and often to the author as well.

4.7 Electronic Archive of Geometry Models

The EG-Models archive [10][9][8] at `http://www.eg-models.de` is a digital journal for the publication of refereed geometry models and experimental data sets. Each published model consists of a data set describing the shape of the model and of a full textual description of the mathematical properties of the model, the experimental computation, keywords,

authors, references, and other information. Additionally, each model is accompanied with an interactive preview applet which enables a reader to interactively investigate the shape of the geometry model.

The EG-Models archive uses JavaView as the geometry viewer in the preview applets and it uses JavaView's JVX geometry file format as a possible choice of the master models. The JVX file format is an XML format which allows the automatic validation of the syntax, and a partial validation of the semantics of submitted geometry models.

Similar to the use of JavaView applets in the Dissertation Online project, the managers of the EG-Models archive have little maintenance overhead when offering such a high level interactive preview facility. The two necessary steps are to copy the JavaView lite archive *jvLite.zip* to the EG-Models server and to include an applet tag referring to jvLite.zip in each preview Html page. New versions of JavaView are easily upgraded by simply replacing the single JavaView archive. In practice, the rendering of pre-computed geometries is one of the most basic tasks in JavaView, making an update to newer versions a rather rare occasion.

Note further, the time required for downloading a preview model often is a factor of 3-10 times larger than downloading a JavaView archive because of the file size of geometry models. The preview models on the EG-Models server are already simplified versions of the master models to reduce the download time and to enable preview even on smaller computers. But still, for example, the preview model of the Penta surface `http://www.eg-models.de/2000.09.039/` is 381kb compared to the size of about 100kb of the JavaView lite archive. A further reduction in size is possible since JavaView is able to read zip and gnu-zip compressed geometry files. JavaView also offers a curvature controlled simplification tool to reduce the number of triangles of a geometry while keeping essential detail information. For example, the preview model of the Penta surface was obtained from the 5,325kb large master model.

4.8 Online Research Cooperation and Experiments

Cooperating researchers who are located at different places nowadays easily communicate via email and other electronic media. But performing a joint experiment still requires the exchange of newly developed software modules and their local installation. Here we encounter one of the major benefits of Java: first, the platform and operating system independence, and second, the invisible installation of Java software via automatic web-based mechanisms.

Biologists in South Africa use a JavaView based applet to compute shortest curves in a beehive to get information on the distance of the queen to selected worker bees, see `http://www.javaview.de/applications/`. Here the biologists load a model of the beehive into the geodesic applet, select the positions of two bees, and invoke the geodesics algorithm of JavaView's jvx package to obtain the shortest curve connecting both positions [16].

This experiment has been automatically invoked on a data base of about 65000 pairs of queen/worker positions. Since the applet is web-based this service can be provided online without any software installation on the client side beside the automatic download of the applet through the web browser. The beehive applet is available on the JavaView web site because of the easy access to the server. The experiments are performed by the biologists from South Africa and the actual computations are done in the web browser of their client computers.

Another example for online research cooperation was the computation of the second variation of area of unstable minimal and constant mean curvature surfaces in a joint project with Wayne Rossman in Kobe in Japan [20]. The cooperation started during Rossman's visit in Berlin and was continued via putting the experimental applets online to be accessible world-wide. The software development was continued in Berlin and the software archive on the web site was regularly updated with the newest version. Both parties from Berlin and Kobe were able to share experiments without a need for a laborious synchronization of software versions.

5 Integration with Other Software

An important application of JavaView is the integration as a viewer and as a geometry processing engine to commercial packages like Mathematica and Maple, or public domain university software. The possible integration with JavaView ranges from simple command line arguments over the pipe mechanism for exchange of geometry data, to a full scripting of JavaView. Currently, the scripting feature is available in Mathematica through J/Link, an optional Mathematica package freely available from Wolfram Research. J/Link allows bi-directional communication, that means JavaView not only provides a viewer for Mathematica graphics but JavaView events are able to invoke and steer Mathematica computations. For example, picking vertices on a geometry currently viewed in a JavaView display may be used as input for Mathematica calculations.

Currently, the following external mechanisms are available to configure, steer and query JavaView:

- Command line arguments.
- Applet parameters.
- Geometry models from file, stdin, applet parameter, and url.
- Configuration files from file, stdin, applet parameter, and url.
- Access to JavaView's class library from any Java class and from Mathematica via J/Link.
- Event listeners may register from any Java code and Mathematica via J/Link.

Special packages and libraries for the integration of JavaView with the following third-party software tools are available at the JavaView site.

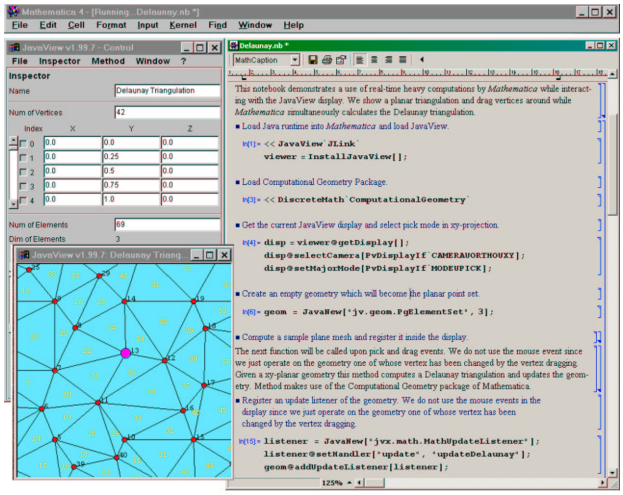

Fig. 11. Real-time computations by Mathematica initiated from pick events in a JavaView display. Vertices are dragged while Mathematica simultaneously calculates the Delaunay triangulation and updates the mesh combinatorics in the JavaView display.

5.1 Mathematica and J/Link

We now take a closer look at the prominent example of the tight integration of JavaView with Mathematica [23]. The J/Link package available for free from Wolfram Research `http://www.wolfram.com` allows the access to Java classes and methods from within a Mathematica notebook, including the handling of Java events. For example, one can type the following sequence of commands in a Mathematica notebook to show Mathematica graphics in a JavaView display and to configure the geometry directly from the notebook:

```
(* Load the Java runtime and initialize JavaView. *)
<<JavaView`JLink`
InstallJavaView[];

(* Create a sample Mathematica Graphics. *)
cube = Graphics3D[Cuboid[{1,1,1}]];

(* Show the geometry in a JavaView display *)
(* and catch the returned JavaView object. *)
jvCube = JavaView[cube];

(* Query and modify the JavaView geometry *)
(* from a notebook. *)
jvCube@getArea[];
jvCube@showVertexNormals[True];
```

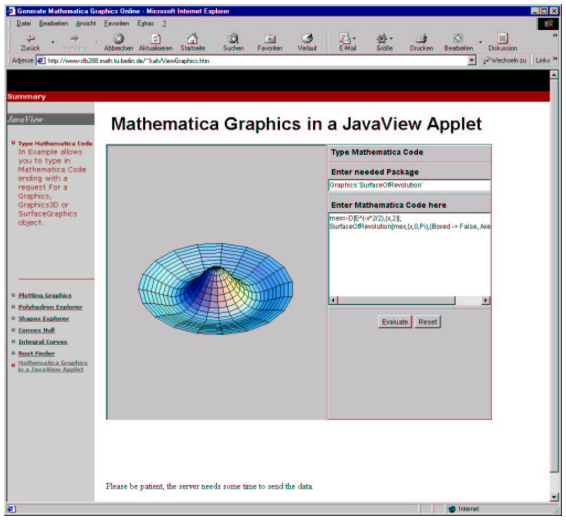

Fig. 12. Typing commands in a computer algebra system online in a web browser, and displaying the graphics result in a JavaView applet. Sample applets created by Klaus Hildebrandt.

Note: the returned object *jvCube* is a full instance of a JavaView class, that means that from this Mathematica notebook all public methods of the geometry class may be invoked. Using J/Link, Mathematica becomes a powerful scripting environment for JavaView programmers, and Mathematica programmers are able to catch 2 and 3-dimensional user pick events in the display.

5.2 WebMathematica

webMathematica [23] by Wolfram Research connects Mathematica to the web. It is a server-based technology built on top of Java servlets which are pieces of Java source code that add functionality to a web server in a manner similar to the way applets add functionality to a browser. Servlets are designed to support a request/response computing model that is commonly used in web servers, similar to cgi scripts. Servlet environments are freely available.

A webMathematica site can return content in many formats including HTML, various image formats, Mathematica notebooks, MathML, and TeX. It can work conveniently with many different web client technologies in browsers such as HTML forms, Java applets, JavaScript, Plug-ins, and ActiveX controls. Running webMathematica requires an HTTP server, including a servlet environment, an installation of Mathematica, and the Mathematica package J/Link .

There are basically two ways to communicate with a webMathematica server: either to write an HTML page that contains extra tags with Mathematica code, or to communicate with the Mathematica server from

a Java applet. In the first case, when an Html page is requested these commands are processed by a Mathematica kernel on the server and the resulting values are included. The usage of Html FORM elements offers possibilities for interactivity. We developed the additional web-Mathematica command MSPJavaView[g] where g means any Mathematica graphics object. This command inserts a JavaView applet to the HTML page which obtains the computed Mathematica graphics as applet parameter.

Figure 12 shows an example where a user types commands online in a text field whose result is then processed by a Mathematica kernel on the server. The result, here consisting of graphics content, is then displayed in a JavaView applet. The following example is for programmers and shows a simple Mathematica server page containing an MSP tag:

```
<html>
<%Mathlet MSPJavaView[Plot3D[Sin[x*y],{x,0,1},{y,0,1}]] %>
</html>
```

After processing, this page is replaced with a regular HTML page including a JavaView applet with the computed Mathematica graphics:

```
<html>
  <applet code="javaview.class" archive="javaview.jar"
    width=400 height=400>
    <param name=mathematica value="SurfaceGraphics[{
      {0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.},
      {0.,0.005102018681303901,0.010203904553163461, ...
      MeshRange -> {{0.,1.}, {0.,1.}}}]">
  </applet>
</html>
```

JavaView equips the web page with an advanced 3D viewer for interactive scientific visualization. Graphics imported from Mathematica are optimized in JavaView for visualization tasks: for example, equal vertices are identified and a topological mesh is created. JavaView provides tools to operate on this structure, e.g. interactive refinement or simplification of the mesh. Additionally, lists of graphics can optionally be shown as animations.

Using MSPJavaView[g] is simple and does not require any knowledge of Java, but it provides access to a variety of different applications. This is mainly achieved by the feature that JavaView projects can be optionally loaded into the viewer, e.g. MSPJavaView[g, Project->"Modeling"]. This offers the possibility of easily using existing applets and to integrate self-made projects.

The second type of communication with the Mathematica server is to write applets and use a URL connection. This equips the applet with the

possibility to send requests to and get responses from the Mathematica kernel. Here interactivity is provided by the applet. In comparison to FORM elements, this gives you more possibilities and is more convenient, e.g. you do not need to reload the whole page after each request. For convenience, the JavaView library contains a special class PsMSP for the communication with a Mathematica server.

5.3 Maple and JavaViewLib

JavaViewLib is an add-on package to Maple [14] developed by Steve Dugaro and one of the authors. The momentum for the JavaViewLib project was initiated in the summer of 2000 by Jonathan Borwein, director of the Center for Experimental and Constructive Mathematics, and Konrad Polthier following the Live Collaborative Mathematics conference at Simon Fraser University. Efforts were aimed at using JavaView to preserve the dynamic viewing capability of Maple for mathematically generated plot objects upon export to the web. Funding by the Telelearning NCE and MathResources allowed for a complete Maple package to be developed. With this package, any maple plot object can be exported through a variety of ways into a superior viewing environment. Furthermore, JavaViewLib allows for models created in a variety of other software packages to be effortlessly imported into Maple worksheets. Many examples of the linking of JavaView and Maple are demonstrated at `http://www.cecm.sfu.ca/projects/webDemo/htm/webdemo.htm`.

JavaView provides a superior viewing environment to augment and enhance the plot of geometrical objects in Maple. It provides several features that are non-existent in the Maple plotter, such as an arc-ball rotation, making object viewing smoother and less directionally constrained than in Maple. Furthermore, JavaView offers a point modeling feature that allows plots to be manually manipulated.

The predominant feature of the JavaViewLib is the capacity to export Maple- generated models into one of two applet-based viewers — one optimized for speed, the other for customizability. This greatly enhances the current state of plot object export in Maple — no longer do dynamic plots need to be converted to static images when creating html pages from Maple worksheets. One can also export plot data to a variety of other formats such as VRML or JavaView's own XML format, where data can be viewed as a markup tree or further developed upon. Efforts were made to maintain the aesthetic presentation of Maple geometries and their corresponding axes upon export. With JavaViewLib, models created in other applications, such as Maya and Mathematica, can easily be imported into Maple's viewing environment.

5.4 Polymake

Polymake [3] by Michael Joswig and Ewgenij Gawrilow is a highly flexible software system which is used by researchers in geometry to investigate

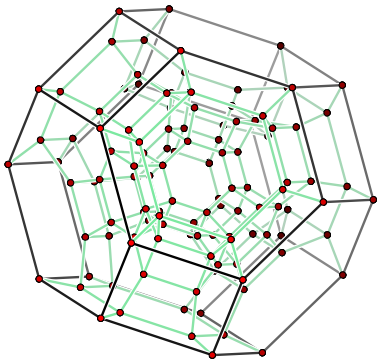

Fig. 13. Schlegel diagram of a 4-dimensional *permutahedron* generated with Polymake and rendered in JavaView. The JavaView display directly renders models in $\mathbb{R}^4$ and hyperbolic spaces.

geometric and combinatorial properties of convex polytopes. As a key feature it offers a wide variety of interfaces to other programs, which greatly enlarges the capabilities of the system. For visualization purposes the system primarily relies on JavaView.

The system allows one to work with polytopes on a rather abstract level, and the user can trigger visualization in an almost naïve way. We give an example:

```
> permutahedron perm4.poly 4
> polymake perm4.poly SCHLEGEL
```

The first command on a standard UNIX shell produces a 4-dimensional *permutahedron*, whose vertices correspond to the 120 elements of the symmetric group of degree 5. The second command asks for a certain 3-dimensional projection of it, a so-called *Schlegel diagram*, see figure 13.

Polymake is a hybrid system where C++ client components interact with a Perl server. Interfaces to external programs are usually done in Perl; JavaView is no exception. In our example, polymake produces a JVX file with a description of the Schlegel diagram which is then passed to JavaView via a `system()` command line call.

6 Conclusion

The internet dramatically changes the classical way of communicating and publishing mathematics. We have sketched some ideas of ongoing changes and the benefits which mathematics will gain from these new developments. The interactive, exploratory component of mathematics, which has been removed from mathematical publications for too long

a time, is now available in the form of Java-enabled software. We have given several demonstrations of JavaView to enhance online publications with multimedia experiments.

References

1. B. Bauslaugh, R. Cannings, C. Laflamme, and K. Nicholson. An intuitive approach to elementary geometry on the web. In J. Borwein, M. Morales, K. Polthier, and J. F. Rodrigues, editors, *Multimedia Tools for Communicating Mathematics*. Springer Verlag, 2001. `http://ilaw.math.ucalgary.ca`.
2. A. Cohen, H. Cuypers, and H. Sterk. *Algebra Interactive!* Springer Verlag, 1999.
3. E. Gawrilow and M. Joswig. polymake, version 1.4: a software package for analyzing convex polytopes, 1997–2001. `http://www.math.tu-berlin.de/diskregeom/polymake`.
4. C. Gunn, A. Ortmann, U. Pinkall, K. Polthier, and U. Schwarz. Oorange: A virtual laboratory for experimental mathematics. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics*, pages 249–265. Springer Verlag, Heidelberg, 1997.
5. T. Hoffmann. jDvi - a way to put interactive TeX on the web. In J. Borwein, M. Morales, K. Polthier, and J. F. Rodrigues, editors, *Multimedia Tools for Communicating Mathematics*. Springer Verlag, 2001.
6. IBM Alphaworks Homepage. `http://www.alphaworks.ibm.com/`.
7. Javasoft Homepage. `http://www.javasoft.com`.
8. M. Joswig and K. Polthier. Digital models and computer assisted proofs. *EMS Newsletter*, December, 2000.
9. M. Joswig and K. Polthier. Digitale geometrische Modelle. *DMV Mitteilungen*, 4:20 – 22, 2000.
10. M. Joswig and K. Polthier. EG-Models - a new journal for digital geometry models. In J. Borwein, M. Morales, K. Polthier, and J. F. Rodrigues, editors, *Multimedia Tools for Communicating Mathematics*. Springer Verlag, 2001. `http://www.eg-models.de`.
11. U. H. Kortenkamp and J. Richter-Gebert. A dynamic setup for elementary geometry. In J. Borwein, M. Morales, K. Polthier, and J. F. Rodrigues, editors, *Multimedia Tools for Communicating Mathematics*. Springer Verlag, 2001. `http://www.cinderella.de`.
12. W. Kühnel. *Differential Geometry, Curves - Surfaces - Manifolds*. American Math. Society, 2001.
13. G. Kuntz. Dynamic geometry on WWW. In J. Borwein, M. Morales, K. Polthier, and J. F. Rodrigues, editors, *Multimedia Tools for Communicating Mathematics*. Springer Verlag, 2001. `http://www-cabri.imag.fr/cabrijava`.
14. Maple Waterloo. Homepage. `http://www.maplesoft.com`.
15. R. J. Morris. The use of computer graphics for solving problems in singularity theory. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics*, pages 53–66. Springer Verlag, Heidelberg, 1997.
16. P. Neumann, C. W. W. Pirk, R. Hepburn, and S. E. Radloff. A scientific note on the natural merger of two honeybee colonies (apis mellifera capensis). *Apidologie*, 32:113–114, 2000.

17. M. Phillips. *Geomview Manual, Version 1.4*. The Geometry Center, University of Minnesota, Minneapolis, 1993.
18. K. Polthier, S. Khadem-Al-Charieh, E. Preuß, and U. Reitebuch. Homepage, 2001. `http://www.javaview.de/`.
19. K. Polthier and E. Preuß. Variational approach to vector field decomposition. In R. van Liere, F. Post, and et.al., editors, *Proc. of Eurographics Workshop on Scientific Visualization*. Springer Verlag, to appear.
20. K. Polthier and W. Rossman. Index of discrete constant mean curvature surfaces. Report 484, SFB 288, TU-Berlin, 2000.
21. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C: The Art of Scientific Computing*. Cambridge University Press, 1993. `http://www.nr.com/`.
22. Sfb256. GRAPE *Manual*. Sonderforschungsbereich 256, University of Bonn, Sept. 1995. `http://www-sfb256.iam.uni-bonn.de/grape/main.html`.
23. Wolfram Research. Homepage. `http://www.wolfram.com`.