RogueWave SOFTWARE

Accelerating Great Code

# IMSL® Library for Java
# Evaluation Guide
# Version 7.3


# The IMSL® Library for Java Evaluation Guide

Rogue Wave Software


Printed in the United States of America

## Trademark Information

# TABLE OF CONTENTS

Installation of the IMSL® Library for Java ..... 5
Documentation ..... 6
Building an Example ..... 8
Demo ..... 9
Demo Gallery Profile Addendum ..... 10
Closing ..... 13

Thank you for your interest in evaluating the IMSL® Library for Java offered by Rogue Wave Software. Since 1970, IMSL Library products have served as the gold standard for comprehensive and robust mathematical, statistical, and financial libraries. It provides advanced analytics to the technical, scientific, and business communities across a wide range of fields. And, as with all our customers worldwide, we trust you will have a positive experience with the JMSL Library written in 100% Java™.

This Evaluation Guide is offered to assist you in quickly evaluating the IMSL® Library for Java. The guide will first discuss the comprehensive documentation supplied with the IMSL® Library for Java. Then it will build an example program from the documentation and finally, we will briefly discuss the JMSL Library Demo Gallery.

As a prospective customer, you may already be working with a Technical Support Engineer to whom you can address any questions during the evaluation period. If you do not have a technical contact, please do not hesitate to contact us at EvalSupport@roguewave.com.

# Installation of the IMSL® Library for Java

**Please follow these step-by-step instructions below.**

Note: Keep in mind that the JMSL Library requires a Java development environment such as Oracle's JDK 7, which is available from Oracle Corporation. We have provided a link in the Readme file that leads to more information about this environment as well as information on download locations.

1. Using a standalone Java jar file, it is easy to install the product. Unzip the download file into a temporary folder and run setup.jar to start the installation process. If your file browser is properly configured, you can double-click on the setup.jar file to start the execution. Otherwise, use a command window and issue the command: java –jar setup.jar to run the application.

2. Once you have started the installation process, as an evaluator simply enter 999999 for your license number when prompted. You may set the install directory to the recommended location or choose another directory. The examples below use `/usr/local/RogueWave/` for Unix/Linux systems and `C:\RogueWave` for Microsoft Windows.

3. After installing the product, please open the `readme.html` file. You can find this file by browsing to the product installation directory.

4. Assuming you have JDK 7 or 8 installed, the next step is to click "Installation" link to open the Installation page.

5. As part of the JMSL Numerical Library, you will receive an evaluation license key that should be placed in the `license` subdirectory in a file named `imsl_eval.dat`. Refer to the README file for specific instructions related to setup of the evaluation license The `imsl_eval.dat` file will be referenced when using the JMSL Numerical Library by the –D command line switch; for example:

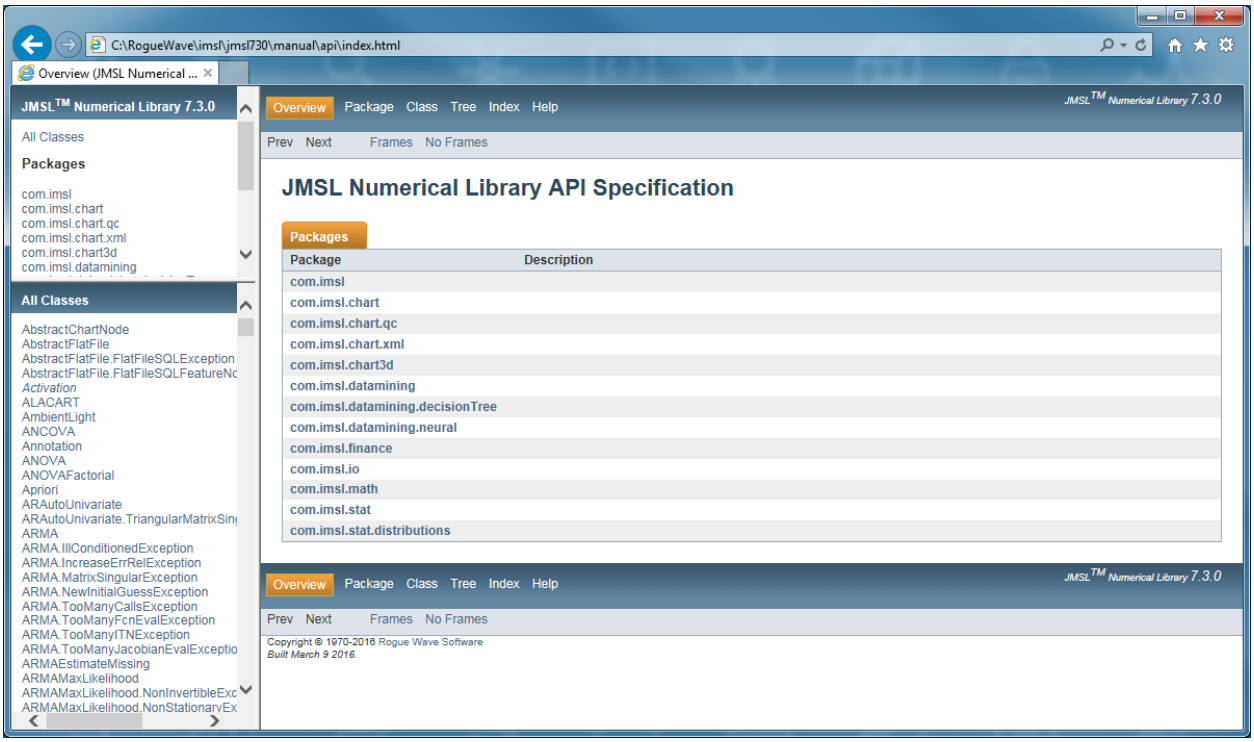
   java –Dcom.imsl.license.path=/usr/local/RogueWave/license/imsl_eval.dat MyApp
   or
   java –Dcom.imsl.license.path=C:\RogueWave\license\imsl_eval.dat MyApp

   Note: On the `Installation` page, you will find more information on how to easily configure and set up the JMSL Library.

Congratulations! You have successfully installed the Rogue Wave JMSL™ Numerical Library.

# Documentation

After successfully installing the JMSL Library, we recommend you briefly view the documentation. Our IMSL Library documentation is renowned for its ease of use and included code examples for clear illustration. As you can see by the screenshot below, it follows the standard javadoc views for pure Java products.




To access the JMSL Library documentation, please follow these steps:

- From the Readme, select the HTML "API Documentation" link. This is a framed HTML document in the standard javadoc format. You will notice that the algorithms are grouped into several packages, including:
  - com.imsl.math
  - com.imsl.stat
  - com.imsl.finance
  - com.imsl.chart
  - com.imsl.datamining

- To get a feel for the content of the documentation, open the `com.imsl.math` tree and then the `DenseLP` class. Here you will find a brief summary of the problem type to be solved and the algorithm(s) used. The length of these sections is proportional to the complexity of the problem or algorithm. For longer examples, you can view the sections covering `com.imsl.math.MinConNLP` or `com.imsl.stat.DiscriminantAnalysis`.

- Scrolling down the page of the `DenseLP` class yields sections that list the Nested Classes, Constructors, and Methods that are associated with this class. Clicking on any of these members gives more information for its use by linking below to the detailed sections; for example, you can see that the only constructor for this object requires a double matrix and two double arrays.

Congratulations! We hope that you have found the documentation to be helpful and easy to use. JMSL documentation includes comprehensive algorithm explanations and references, and customers around the world view it as a very valuable resource.

# Building an Example

Now that you have successfully installed the JMSL Library and become familiar with the documentation, we can now put the JMSL Library to some use. For that purpose, we will use a regular text editor and the command line tools provided with the Java JDK. Please note that you can build this application with any IDE such as Eclipse, NetBeans, JCreator, or many commercial products. The primary task in setting up a project using JMSL in an IDE is to add the `jmsl.jar` archive to the project.

A Straightforward Example:

1. For our example application, we will use some code from the help file, so open the API documentation. Again, you will need to browse to the `DenseLP` class. Follow the link to Example 1 that is in the See Also section. At the bottom of that page, follow the link to Java source code. Click on the link and save the file, `DenseLPEx1.java` to a working directory (we'll assume `/usr/local/java`). This example solves a straightforward linear programming problem: to minimize the function $f = -x_1 - 3x_2$ subject to several constraints.

2. With the file saved, now open a Command Shell and change to the directory where the file is saved: cd /usr/local/java or cd c:\java. The code will be compiled with the `javac` command line compiler, which should be in your path. If not, you can add it or simply provide the full path to the compiler. The `jmsl.jar` will be pointed to as well:

   ```
   javac -cp /usr/local/RogueWave/imsl/jmsl730/lib/jmsl.jar DenseLPEx1.java
   ```

3. The compile step creates a `DenseLPEx1.class` file that we now want to execute. This time, we need to supply the path to the JMSL license file along with the path to `jmsl.jar`:

   ```
   java -Dcom.imsl.license.path=/usr/local/RogueWave/license/imsl_eval.dat
     -cp .:/usr/local/RogueWave/imsl/jmsl730/lib/jmsl.jar DenseLPEx1
   ```

   For Windows users, this line would look more like the following:

   ```
   java -Dcom.imsl.license.path=C:\RogueWave\license\imsl_eval.dat
     –cp .;C:\RogueWave\imsl\jmsl730\lib\jmsl.jar DenseLPEx1
   ```

4. The output of this command should be the following text:

```
Solution
  0
0 0.5
1 1
2 0
3 1
4 0.5
5 0
```

Congratulations! In only a few short minutes, you have used the JMSL Numerical Library to solve a Linear Programming optimization problem. This technique is widely applicable for financial engineering, operations research, and R&D product and process optimization, to name a few. Conservative estimates show the JMSL Numerical Library can save close to 95% of the development costs organizations would incur if they were to build algorithms from scratch. Numerical algorithms are very expensive to research, build, and maintain. Organizations can realize tremendous productivity gains by relying on fully tested and documented classes from IMSL.

Additionally, the IMSL Library for Java is the most comprehensive mathematical and statistical library available in pure Java, offering seamless operability with your existing Java applications.

# Demo

Included with the product is copy of the Demo Gallery of the IMSL Library for Java. The `gallery.jar` can be found in the `/lib` directory of the product installation. These demos will illustrate some of the functionality that is provided by the JMSL Library.

The Demo Gallery jar file can be run by double-clicking it if .jar files are configured as executables on your operating system. Alternatively, you may use the command line to get your demos running,

- Change to the `/lib` subdirectory.
- To start the demo gallery, execute the jar file with command line:

```
java -Dcom.imsl.license.path=/usr/local/RogueWave/license/imsl_eval.dat -jar gallery.jar
```

The JMSL Library product was used to build these functional demonstrations. The demos themselves are not the JMSL Library product -- they are brief, straightforward examples that illustrate parts of applications that can easily be built now that the JMSL Library is available. These functional demos showcase a selection of the broad mathematical and statistical analysis capabilities available in 100% Java. All of the source code for the demos is available with the JMSL product. For easy reference to help you quickly understand each of the demos, a brief profile of each functional demo is provided in the Demo Gallery Profile Addendum below.

## Demo Gallery Profile Addendum

1. Portfolio Optimization – This demo shows the ease in which one can use the JMSL Library for the computation, analysis and charting of the efficient frontier, and how to select the optimal solution from the set of possible solutions. In this demo, there are four indices that consist of generic security price data. If you have four different indices you will have an optimal portfolio that optimizes risk-adjusted return. The efficient frontier curve is the set of feasible portfolios that have the minimum possible risk along the range of feasible rates of return. To find the best portfolio of the set, one draws a line tangent to the Efficient Frontier through the risk-free rate of return.

2. Risk Analysis (Monte Carlo Simulation) – In this demo, using the historical performance of the indices, we simulate the performance of a portfolio that is made from varying levels of investment in each index. The amount invested in each index is listed in the "Portfolio Content" fields. Using the content fields you can change the amounts invested in each index as well as the number of samples used in the simulation. By clicking the "Update" button, IMSL will be used to simulate the performance of the new portfolio and display the results in the histogram. The results of this process allows for quick and easy "what-if" analysis of portfolios.

3. Data Fitting – This demo program illustrates how the JMSL Library can be used to develop Java applications that fit sampled data with a linear regression, a variety of cubic splines or to a nonlinear model, or to a contour model. In essence, this demo is four demonstrations in one. In each demonstration, the program computes a best fit based on the plotted data. The user may add additional or remove existing points to see how the fit changes. As data points are added or removed, the computed fits are updated automatically.

4. Forecasting – This demo illustrates how the JMSL Library can be used to quickly build an application that uses ARMA statistical methodology to forecast time series based on historical data. Although the particular time series available in this demo is champagne sales per month, sun spot frequency per year, or the price of crude oil, such an analysis could apply to numerous types of cyclical time series data such as hurricane frequency, El niño intensity, or securities prices.

5. Neural Network Forecast – In this demo, the results of a Neural Network forecast are compared to an ARMA(2,1) model and the actual data for three data sets. The neural network forecasting technique can generate forecasts for data with very challenging and complex characteristics. Such characteristics can include noisy data, seasonal data, short time series, categorical data, as well as numerous possible, but unknown, interactions.

6. Statistical Examples – This demo illustrates how the JMSL Library can be used to develop Java applications based on various statistical methods. Those shown here are Random Number Generation, Correlation Visualization, Cohen's d and a Radar Plot.

7. Cluster Analysis – This demo illustrates how the JMSL Library can be used to perform both k-means and hierarchical cluster analysis. Cluster analysis aims to group data with similar quantities together. For this two-dimensional case, physical interpretation is straightforward, but this is rarely the case for complex data mining.

8. Predator Prey – In this demo, the predator-prey problem computes the population density of rabbits and foxes over time. As you can see here, as time goes on the foxes will eat the rabbits. After the rabbits are gone, the foxes die, the rabbits resurface and the process begins anew. This chart is essentially showing a graphical representation of "the circle of life". It shows how with the JMSL Library, problems requiring both numerical analysis and charting can be solved with a single, integrated Java solution.

9. Harmonic Analysis (FFT) – This demo illustrates how the JMSL Library can be used to perform spectral analysis of time series data. The signals are recordings of the four open strings of a cello being bowed. Each signal is of approximately constant pitch and thus has an easily interpretable spectral signal. You can select any portion of the wave form to analyze by dragging a rubber band box in the chart window. Then, click "Calculate FFT" to see the results in a separate chart frame.

10. Demographics – This is a basic demo showing how the JMSL Library can be used to easily create a Java application that extends the basic charting features. Historical demographic data

are plotted in a horizontal double-sided bar chart.

11. Heat Maps – This demo program illustrates how the JMSL Library can be used to develop Java applications that require data presented in a heat map graphic. There are three separate examples using different data and different visualization techniques. This chart type is used in a broad range of industries; however, it is commonly used in life sciences to display two-dimensional arrays of color values. One common application is to display protein expression results graphically.

12. Kalman Filter – This demo illustrates how the JMSL Library can be used to easily create a Java application that examines the uncertainty of financial data. The Kalman filter enables the user to address the question of how much of the volatility of the moment-to-moment stock price is due to the market's pricing uncertainty and how much is due to "true" changes in a company's value.

13. Stock Price Charting – This demo shows the ease and flexibility in which you can track the performance of many, many stocks in your portfolio. These particular stocks are from the SP500 index however they could just as easily be any other historical or real time data. Note that some stocks were not in the index for the full time period covered and so have fewer data points.

14. Bond Price Sensitivity – This application demonstrates how the JMSL Library financial algorithms and charting capabilities can be used to calculate and display the sensitivity of the value of bonds to changes in current interest rates. The window on the left includes a list of US Treasury bonds, with different maturity dates and different nominal interest rates. By selecting a bond from the window on the left, the JMSL Library is used to calculate the sensitivity of that particular bond to the market interest rate.

15. Select Points – This demo helps show the flexibility of chart interaction with the JMSL Library, as well as the seamless combination of chart interaction with analysis capabilities. This demo illustrates the ability to use the JMSL Library to easily create Java applications that allow visual data interaction and live statistical analysis.

16. Strip Chart – This demo illustrates how the JMSL Library can easily be used to build a Java application that incorporates live data. This demo shows first of all that strip charts are possible with the JMSL Library, and secondly it demonstrates that the frame around the chart can be extended to include additional menus.

## Closing

Thank you for your interest in Rogue Wave's JMSL Numerical Library. We hope that this brief evaluation guide has broadened your understanding and demonstrated a few key benefits about the JMSL Numerical Library. If you would like a technical expert to help you through a more complex code example or if you have any questions or need any technical assistance, please contact us at EvalSupport@roguewave.com.