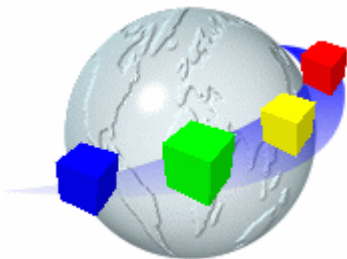


WebCab Portfolio v4.2

WebCab
Components



Preface

This documentation accompanies the WebCab Portfolio J2EE Application. The purpose of this documentation is to provide a clear and concise description of all aspects that are likely to be encountered in real life applications by developers and users of this J2EE Application. WebCab Portfolio contains methods which use the Markowitz and Capital Asset Pricing Model to construct a portfolio which for a given level of risk offers the greatest expected return.

The first chapter of this documentation contains a brief introduction to the most important implemented features and related system requirements. In chapter two we let the developer quickly get started with deploying the component by detailing deployment techniques on the most widely used application servers. We also detail how the functionality of this component can be tested through connecting to online web service demos hosted at [J2EE Home at WebCabComponents.com](http://www.webcabcomponents.com). The third chapter contains the mathematical documentation, which represents the theoretical background of this component's implemented features. Chapter four contains the programmer's guide containing a road map for developers to take advantage of every feature and capability. In chapter five we provide some examples to illustrate how features detailed within the programmer's guide can be applied in practice. In chapter six, UML diagrams are provided for each of the components interfaces. Finally, we introduce WebCab Components, its philosophy and approach to serving the JavaTM development community with robust and powerful J2EE applications.

If you have any questions or queries concerning the application or use of this component then please feel free to contact us via our support forum at:

<http://www.webcabcomponents.com/support/index.php>

Good luck with your project and thank you for your interest in our components.

The WebCab Components Team

Contents

Preface	i
1 Introduction	1
1.1 Product Description	1
1.1.1 Overview	1
1.1.2 Details	1
1.2 Package Details	3
1.3 Prerequisites and Compatibility	4
1.3.1 System Requirements	4
1.3.2 Compatibility	4
2 Where do I Start?	6
2.1 Self-Deploy Installer	6
2.2 Deploying this Component	7
2.3 IBM WebSphere Application Server	7
2.3.1 Deploying on IBM WebSphere V4.0 and V5.0	7
2.4 BEA WebLogic Server	9
2.4.1 BEA WebLogic Server 6.1	9
2.4.2 BEA WebLogic Server 7.0 (J2EE 1.3 compliant)	10
2.5 Oracle9i Application Server	12
2.5.1 Oracle9i v9.0.2	12
2.5.2 Oracle9i v9.0.3 (J2EE1.3 certified)	13
2.6 Sun ONE AppServer 7	14
2.7 Borland Enterprise AppServer 5	14
2.8 Sybase EAServer 3.6	16
2.9 Ironflare Orion Server	16
2.9.1 Orion 1.5.2 (and Orion 1.5.4)	16
2.9.2 Orion 1.6 (EJB2.0 compliant)	17
2.10 JBoss Open Source Application Server	18
2.10.1 JBoss 2.4.4 (and JBoss 2.4.3)	18
2.10.2 JBoss 3.0.0 (EJB2.0 compliant)	18

3	Mathematical Documentation	19
3.1	Assumptions, Questions, Functionality and Problems	19
3.1.1	Assumptions of Markowitz Theory	19
3.1.2	Assumptions of Capital Asset Pricing Model (CAPM)	20
3.1.3	What problems do these Models address	20
3.1.4	Range of Functionality contained within the :services:	21
3.1.5	Problems with the Application of the Markowitz Theory and CAPM	22
3.2	Preliminaries and Auxiliary EJB Component's	25
3.2.1	Choices in Approach and other issues	25
3.2.2	Basic Formulae for the Return, Covariance, Correlation and Risk	26
3.3	Expected Return and Risk from a Portfolio with two Assets	30
3.3.1	Motivation	30
3.3.2	Formulation for a Portfolio of two assets	31
3.3.3	Minimum risk of a portfolio with two assets	31
3.4	Markowitz Theory	32
3.4.1	Overview of Markowitz Theory Implemented	32
3.4.2	Construction of the Efficient Frontier	32
3.4.3	Constraints effect on the range of the Efficient Frontier	34
3.4.4	Consistency of the Assets and Effects on the Efficient Frontier	35
3.4.5	Selecting the Optimal Portfolio	41
3.4.6	Discovering the Investors risk - return profile	44
3.4.7	Examples of selecting the Optimal Portfolio from the Investors Utility Function	45
3.5	Capital Asset Pricing Model (CAPM)	48
3.5.1	Overview	48
3.5.2	Nature of the Capital Asset Pricing Model (CAPM)	49
3.5.3	Applying the CAPM	51
3.5.4	Summary of the CAPM	55
3.5.5	Putting constraints on the level of borrowing and lending	56
3.6	Performance Evaluation	58
3.6.1	Comparing the Sharpe and Treynor Performance Measures	59
3.7	Further and Supplementary Reading	59
3.7.1	Supplementary Reading	59
3.7.2	Further Reading	59
4	Programmer's guide	60
4.1	The Portfolio components methods	60
4.2	Exceptions	61
4.3	Interfacing with databases	61
4.4	Writing a typical client	62
4.5	Using JDBC Mediator with our EJBs	65
4.5.1	Overview	65
4.5.2	Connecting to your Database Server	66
4.5.3	JDBC Components	67

4.6	Support for Developers	70
4.6.1	Compilation and Custom Modification of Source Code	70
4.6.2	Online Support	71
5	Examples	72
5.1	Question and Answer (QA) Client Examples	72
5.1.1	Overview	72
5.1.2	Structure of QA Examples Directory	72
5.1.3	Top Four levels of the QA Directory Structure	73
5.1.4	Bottom Two levels of the QA Directory Structure	73
5.1.5	Quick Start Guide	73
5.1.6	Explanation of the QA Directory Structure and its files	74
5.2	Custom QA Examples	75
5.2.1	Markowitz Custom Clients	75
5.2.2	Capital Market Custom Clients	81
5.2.3	Asset Parameters Custom Clients	84
5.2.4	Two Asset Portfolio Custom Clients	85
5.2.5	Optimal portfolio	86
5.3	Database Example with JDBC Mediator	87
5.4	How to use Markowitz Theory?	88
5.4.1	How to find the portfolio from a collection of assets that exhibit the lowest risk for a given expected future return?	88
5.4.2	How to construct the Efficient Frontier?	89
5.4.3	How to estimate the expected return from the historical returns?	90
5.4.4	How can I measure the performance characteristics of a portfolio?	91
6	Guide to WebCab Components	93
6.1	The Company	93
6.2	Presentation of Products	93
6.3	Supported Clients, IDEs, Containers and DBMSs	93
6.4	Transparent Functionality	94
6.5	Code Conventions	94
6.6	Company Culture and Activity	94
6.7	Product Life cycle	94
6.8	Support, Warranty and Upgrades	95

Chapter 1

Introduction

1.1 Product Description

1.1.1 Overview

Apply Markowitz Theory and Capital Asset Pricing Model (CAPM) to analyze and construct the optimal portfolio with/without asset weight constraints with respect to Markowitz Theory by giving the risk, return or investors utility function; or with respect to CAPM by given the risk, return or Market Portfolio weighting. Also includes Performance Evaluation, extensive auxiliary classes/methods including equation solve and interpolation procedures, analysis of Efficient Frontier, Market Portfolio and CML.

1.1.2 Details

This suite includes the following features:

- **Markowitz Model** - Construct optimally diversified portfolios.
 - **Efficient Frontier** - Construct the Efficient Frontier with or without constraints on the asset weights.
 - **Utility Function** - Discover and set the investors utility function.
 - **Optimal Portfolio** - Select the optimal portfolio or set of portfolios by providing the expected return desired, the maximum risk or the investors utility function.
- **Capital Asset Pricing Model (CAPM)** - Construct optimally diversified portfolios with can hold or borrow cash.
 - **Efficient Frontier** - Construct the Efficient Frontier with or without constraints on the asset weights.
 - **Market Portfolio** - Find the Market Portfolio which offer the greater expected return per unit of risk.

- **Capital Market Line (CML)** - Construct the CML with contains the optimal portfolio with respect to the CAPM.
- **Selecting Optimal Portfolio** - Select the optimal portfolio by given expected return, risk or the Market Portfolio weighting.
- **Analysis of Optimal Portfolio** - Evaluate the risk, expected return or Market Portfolio weighting of the optimal portfolio whenever one of these three properties is known.
- **Auxiliary Classes**
 - **Interpolation** - Cubic spline and general polynomial interpolation procedures to assist in the study and manipulation of curves such as the Efficient Frontier which are evaluated at a finite number of points.
 - **SolveFrontier** - Solve the Efficient Frontier with respect to the risk, return, or the investors utility function which may be given as a function of the risk or the expected return.
 - **TwoAssetPortfolio** - Evaluate of the optimal weighting of a portfolio with two assets. This functionality can be used to analyze the effect of a single purchase or sale from an arbitrary portfolio
 - **AssetParameters** - Evaluation of the covariance matrix, expected return, volatility, portfolio risk/variance, ARCH model for expected price.
 - **MaxRange** - Evaluates the maximum range of the values of the expected return for which Efficient Frontier should be considered when the historical data set does is not consistent within the assumptions of Markowitz Theory and CAPM.
 - **Performance Evaluation** - Offers a number of procedures for accessing the return and risk adjusted return (Treynors Measure, Sharpes Ratio).

This product also contains the following features:

- **GUI Bundle** - we bundle a suite of graphical user interface JavaBean components (with 1, 2, 4 or site-wide license) allowing the developer to plug-in a wide range of GUI functionality (including charts/graphs) into their client applications
- **EAR Files** - we provide individual customized EAR files for the most widely used application servers including IBM WebSphere 4.0/5.0, BEA WebLogic 6.1/7.0, Oracle 9iAS, Sun ONE AppServer 7, Ironflare Orion 1.5.2/1.6.0, Borland AppServer 5.0, Sybase EAServer 3.6 and JBoss 2.4.4/3.0.0
- **Self-Deploy** - the relevant servers EAR file will be self-deployed onto supported local application servers during the installation of the self-install package. The supported application servers include IBM WebSphere 4.0/5.0, BEA WebLogic 6.1/7.0, Oracle 9iAS, Borland AppServer 5.0, Ironflare Orion 1.5.2/1.6.0 and JBoss 2.4.4/3.0.0

- **JDBC Mediator** - A server side EJB component which mediates between an EJB component, clients and DBMS. The mediator moves most of a clients JDBC calls to the server and hence greatly increases the speed of JDBC client applications.
- **Web Application Example** - A JSP interactive HTML interface which enables you to test every component method directly from your browser.
- **Synthetic JDBC** - we use a JSP component to perform EJB calculations on SQL database columns from a remote DBMS. We apply an EJB function to certain rows from the database and list the output in HTML format. This is a powerful feature since it allows you to perform calculations in a JDBC manner without having to code the EJB-to-JDBC transaction yourself as it is all done by the JSP within the Application Server.
- **UML Models** - to assist system architects we provide UML diagrams of this component

1.2 Package Details

The Portfolio v4.2 J2EE Application contains the following:

- Introductory Text File (README.TXT)
- License Agreement
- Documentation in PDF Format
 - Product Description
 - System Requirements
 - Compatibility Issues
 - Deployment Guide (How to get started?)
 - Mathematical Documentation
 - Programmer's Guide
 - Examples
 - UML Models
 - Guide to WebCab Components
- JavaDocs Documentation
 - Class Descriptions
 - Methods Descriptions
- Deployment Archives (EAR)
- Examples and Related Source Code Files
- WebCab Components Brochure

1.3 Prerequisites and Compatibility

1.3.1 System Requirements

- An Operating System running Java™
- Pentium® III 733 MHz
- 256MB RAM
- A J2EE1.3 (EJB 2.0) compatible Application Server

1.3.2 Compatibility

Operating System for Deployment

- Windows 2003, XP, 2000, NT
- Sun Solaris
- Linux
- IBM AIX
- HP-UX

Component Type

- Enterprise JavaBeans™

Built Using

- Java™ 2 SDK Standard Edition 1.4.2
- Java™ 2 SDK Enterprise Edition 1.3

Compatible Application Servers

- IBM WebSphere Application Server V4.0/V5.0
- BEA WebLogic® Server 6.1/7.0
- Oracle® 9i Application Server
- Sun One Application Server 7
- Borland Enterprise AppServer 5.0
- Ironflare Orion Server 1.5.2/1.6.0
- Sybase EAServer 3.6

- JBoss 2.4.4/3.0.0

Chapter 2

Where do I Start?

Start using the Portfolio v4.2 J2EE Application right away by following the few quick and simple steps described in this chapter. We provide support for most Java compatible operating systems, such as Windows, Linux, IBM AIX, HP-UX and Solaris. Along with the most widely used Application Servers including IBM WebSphere, BEA WebLogic, Oracle9i, Sun ONE AppServer, Borland AppServer, Sybase EAServer, Ironflare Orion, and JBoss. If you require additional information regarding the use of this J2EE Application, then please don't hesitate to contact us via our support forum at: <http://www.webcabcomponents.com/support/index.php>.

2.1 Self-Deploy Installer

If you are locally using one of the following application servers:

- IBM WebSphere V4.0/V5.0
- BEA WebLogic 6.1/7.0
- Oracle9i v9.0.2/v9.0.3
- Borland Enterprise AppServer 5
- Ironflare Orion 1.5.2/1.6.0
- JBoss 2.4.4/3.0.0

During the installation of this Application, you were prompted to follow the deployment procedure of your application server. In this case the Application should already be deployed on your application server and you can start using it immediately.

If the Application is not installed on your local application server or you wish to deploy on a remote application server then you should follow the relevant steps below.

2.2 Deploying this Component

Depending on the Enterprise platform you have chosen for distributing your business components, you will follow different procedures when starting off. Within this chapter we describe individual Application Server deployment procedures. Please feel free to skip to the section that refers to the Application Server you are planning to use.

In case your Application Server is not listed in this chapter or the installation of your Application fails, please send us an email to Support@WebCabComponents.com attaching any logs or error messages.

2.3 IBM WebSphere Application Server

The deployment procedures for IBM WebSphere V4.0 and V5.0 are the same. The only difference is that on WebSphere V4.0 you will be deploying the EJB1.1 version of Portfolio, whereas on WebSphere V5.0 you will be able to install the EJB2.0 version as well.

2.3.1 Deploying on IBM WebSphere V4.0 and V5.0

Starting the Server

The WebSphere Application Server (WAS) requires to be up and running at deployment time. Start the server under Windows by selecting **Programs→IBM WebSphere→Application Server V4.0** (resp. **V5.0**). Alternatively, the server can be started by running the `startServer.bat` script from the installation path. Under Linux log in as **root**, change to the `${WAS_HOME}/bin` directory, where `${WAS_HOME}` is the WebSphere installation path, and type the following line at command prompt¹:

```
./startServer.sh
```

Connecting to the Administration Console

Connect to the WAS Administrative Console, by opening a browser and typing in the following URL address:

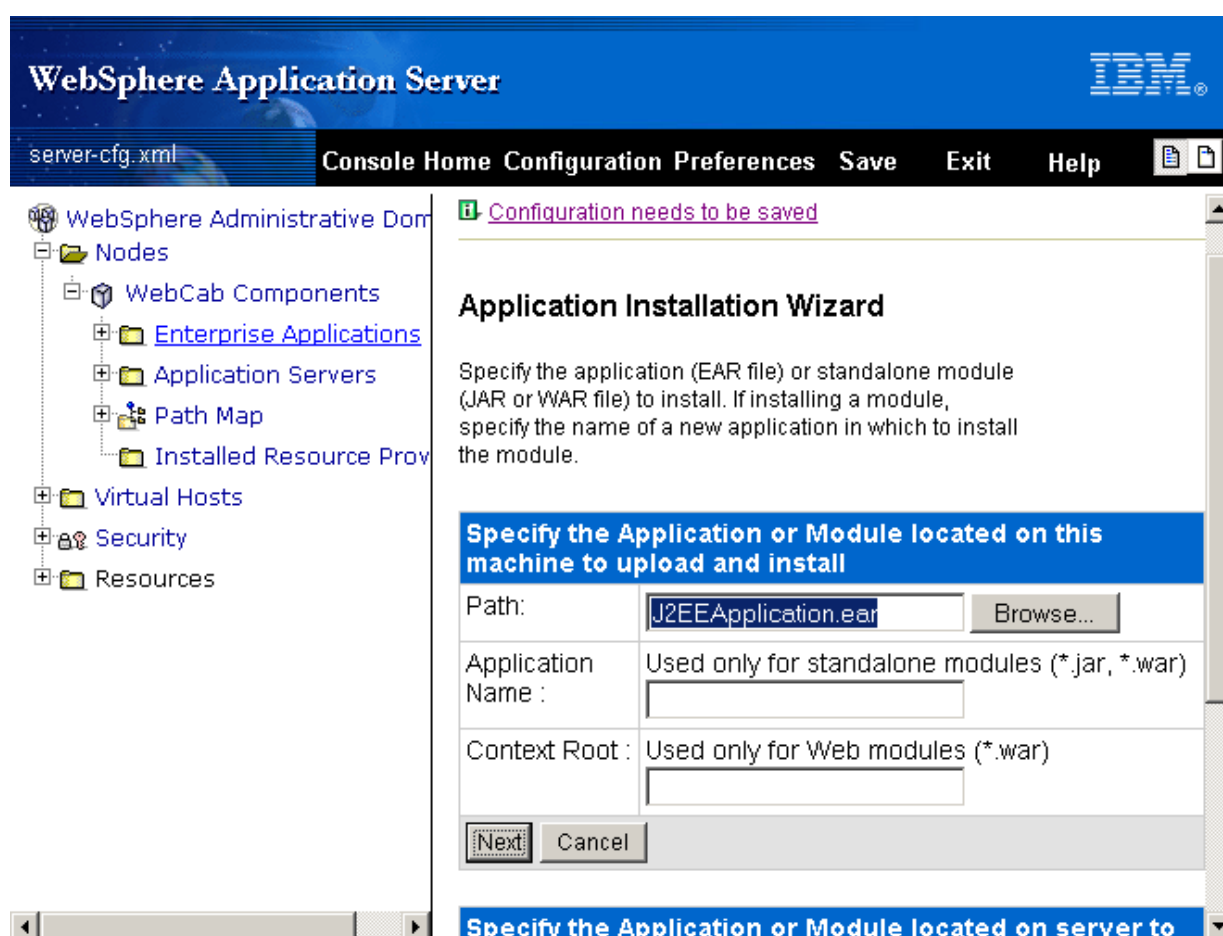
`http://localhost:9090/admin`

where `localhost` can be replaced by the hostname of the computer running WebSphere when connecting from a client machine. The browser will bring up a login page and ask you to fill in a name in order to track user-specific changes to configuration data. You may leave the field blank and press the **Submit** button. If an Alert page is displayed, press the **Cancel** button and continue.

¹ `${WAS_HOME}` usually defaults to `/opt/WebSphere/AppServer` under Unix.

Deployment

The Administrative console manages the Application Server into nodes and each node runs its own Enterprise Applications. You will be deploying inside one of the existing nodes which you can select from the left panel by expanding the **Nodes** folder. Expand the name of the installation node and select the **Enterprise Applications** item. The browser will bring up in the central frame a list of all currently installed Enterprise Applications. Click the **Install** button, browse the **Portfolio.ear** file, as installed under the *[ROOT_DIR]/Portfolio/Deployment/IBM WebSphere V4.x*² directory and click **Next**³.



IBM WebSphere V4.0/V5.0 Application Installation Wizard Page.

Click again **Next** through every screen until you reach the **Mapping Resource References to JNDI Names** screen. You will be required to map certain resource references to WebSphere DataSource JNDI names⁴. Keep clicking **Next** in every screen and press **Finish** to complete the deployment.

²If this is the V5.0 version of WebSphere, consider browsing from *[ROOT_DIR]/Portfolio/Deployment Ejb2.0/WebSphere V5.0*.

³*[ROOT_DIR]* represents the Portfolio installation directory.

⁴Read the IBM WebSphere documentation for information about how to define DataSources.

After waiting for a few minutes, the Administrative Console will display a page where Portfolio v4.2 will be listed among the currently installed Enterprise Applications.

Restarting the server

When installing Portfolio v4.2 for the first time inside WebSphere, you will be required to restart the IBM server before running it. You will also need to save the current configuration before doing so. Click on the *Configuration needs to be saved* link located at the top of the page and confirm the save by pressing OK.

In order to stop the server, click the **Application Servers** folder from the left pane under the current node and press the **Stop** button. In the next page click **OK** and wait for the console to confirm a successful shutdown. The next time you start the WebSphere server the Portfolio Application will be installed and made available to all clients.

2.4 BEA WebLogic Server

2.4.1 BEA WebLogic Server 6.1

Enterprise JavaBean™ components and J2EE Application are deployed inside a running WebLogic 6.1 server through a web browser interface. On this server you will be installing the EJB1.1 version of Portfolio.

Starting the server

Start BEA WebLogic 6.1 under Windows from the **Programs→BEA WebLogic E-Business Platform→WebLogic Server 6.1→Start Default Server** icon in the Start Menu. If you're running WebLogic 6.1 under Linux, change the current directory to `/home/user/bean/wlserver6.1/config/mydomain`, where *user* is your Unix account and *mydomain* points to the default domain name. Within this folder type the following line at command prompt:

```
./startWebLogic.sh
```

If asked for, enter the login password and wait a few seconds until the server to initializes.

Connecting to the WebLogic Console

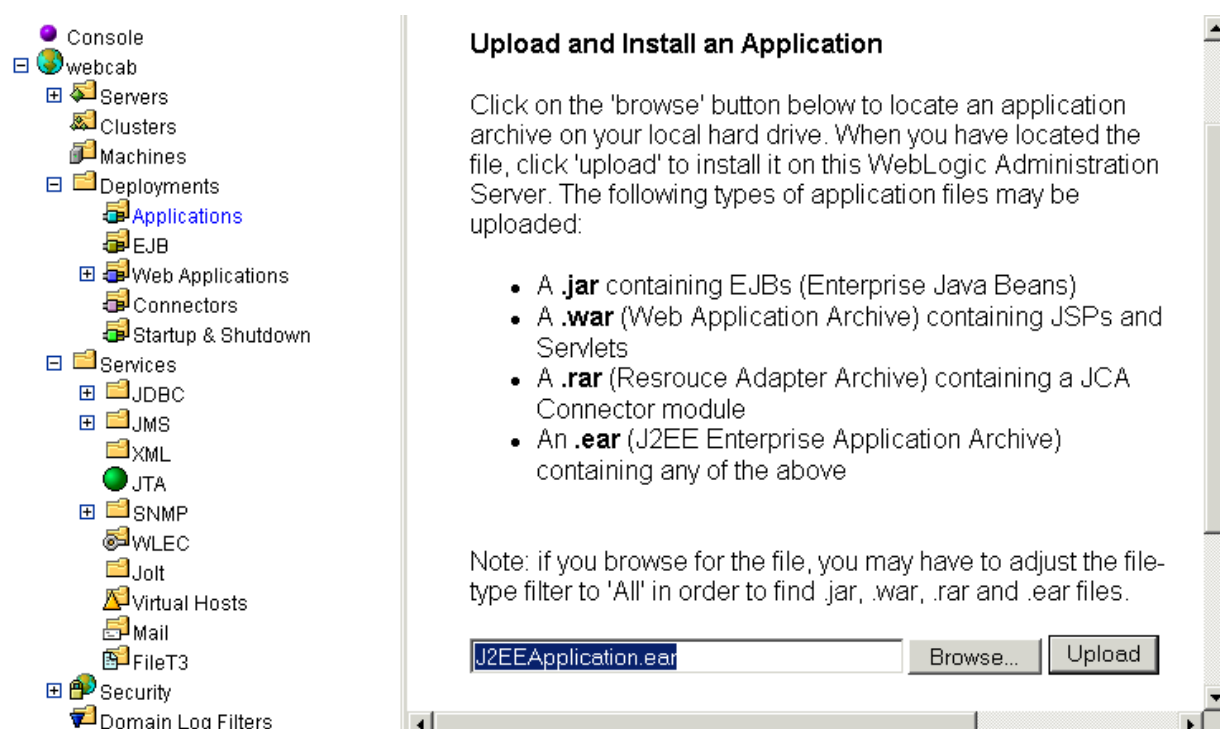
When the WebLogic Server is up and running, open an Internet browser and type in the following address:

```
http://localhost:7001/console
```

where `localhost` can be replaced by the hostname of your server running WebLogic, when connecting from another client machine. At the login screen enter the user name ("system" by default) and type in the login password. The browser will bring up the BEA WebLogic Server support interface.

Deployment

You will see in the left side of the screen your domain's name and a list of currently deployed Enterprise Applications. Click on the **Applications** item under the **Deployments** folder, then click the *Install a new Application...* link in the right panel.



WebLogic Server 6.1 Upload and Install Screen.

Browse the `Portfolio.ear` archive from `[ROOT_DIR]/Portfolio/Deployment/BEA WebLogic 6.1` directory and click the **Upload** button⁵. The J2EE Application will be uploaded and deployed inside the WebLogic Server 6.1.

2.4.2 BEA WebLogic Server 7.0 (J2EE 1.3 compliant)

Enterprise JavaBean™ components are deployed inside a running WebLogic 7.0 server through a web browser interface. Since WebLogic Server 7.0 is J2EE1.3 compatible, you will be installing the EJB2.0 version of Portfolio.

Starting the server

Start BEA WebLogic 7.0 under Windows from the `Programs→BEA WebLogic Platform 7.0→User Projects→mydomain` folder in the Start Menu, by clicking the “Start WebLogic Server” icon. Alternatively, you may start the server by running the `startWLS.cmd` script inside the `C:\bea\weblogic700\server\bin` folder.

⁵`[ROOT_DIR]` is the Portfolio installation directory, as specified at installation time.

If you're running WebLogic 7.0 under a Linux platform, change the current directory to `[WEBLOGIC_HOME]/weblogic700/server/bin`, where `[WEBLOGIC_HOME]` points to the BEA home directory, as installed under Linux, for example `/home/user/boa`. In this folder type the following line at command prompt:

```
./startWLS.sh
```

If asked for, enter the login password and wait a few seconds until the server initializes.

Connecting to the WebLogic Server Console

When the BEA WebLogic Server is up and running, open an Internet browser and type in the following address:

```
http://localhost:7001/console
```

where `localhost` can be replaced by the hostname of the server running WebLogic, when connecting from another client machine. In the login field, enter your user name ("system" by default) and type in the login password. The browser will bring up the BEA WebLogic Server 7.0 Console.

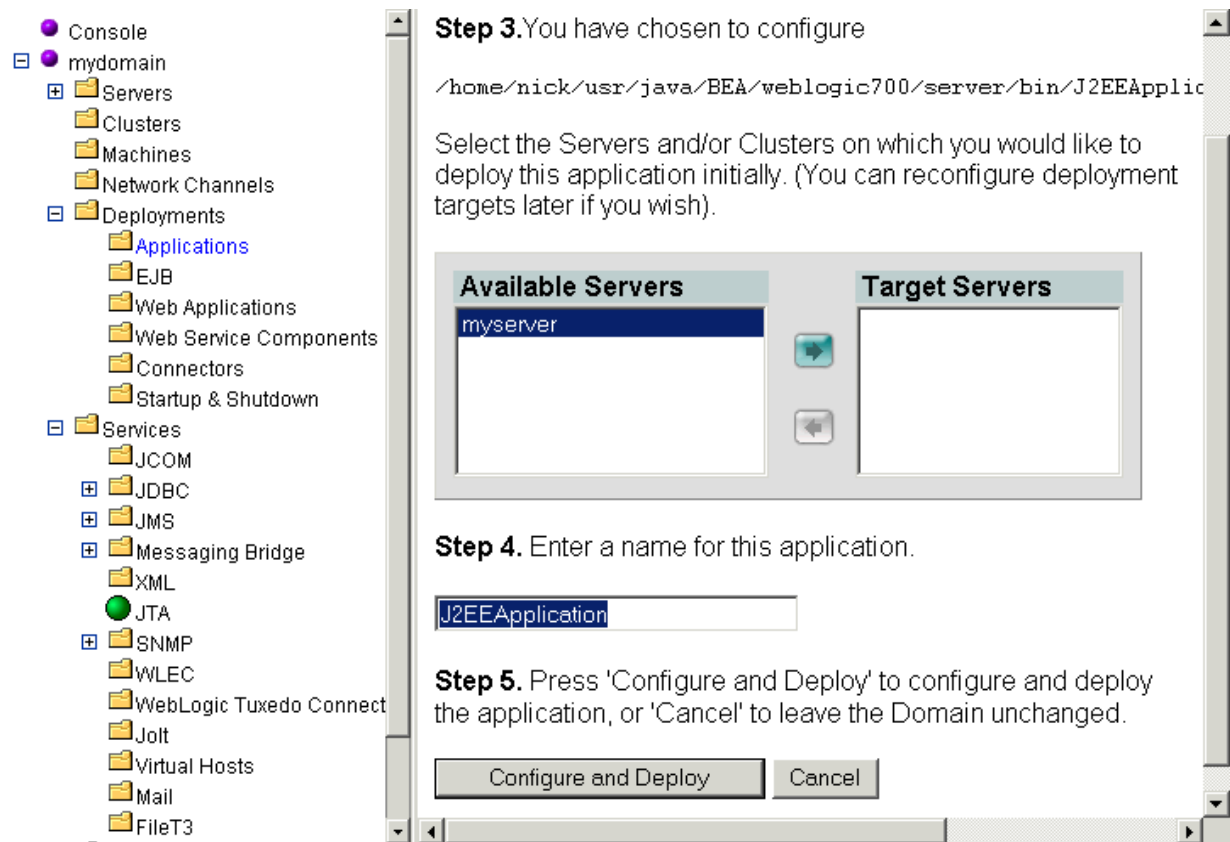
Uploading the Application

You will see in the left side of the screen your domain name and a list of currently deployed Enterprise Applications under the **Deployments/Applications** tree. Click on the **Applications** item under the **Deployments** folder, and press *Configure a new Application...* in the right panel. Click the link that says "upload it through your browser" and browse the `Portfolio.ear` archive from `[ROOT_DIR]/Portfolio/Deployment Ejb2.0/WebLogic 7.0` directory and click the **Upload** button⁶.

Deployment

After the upload is complete the page will display at the bottom of the screen the name of the uploaded file, `Portfolio.ear`. Click the "[select]" link at its left.

⁶`[ROOT_DIR]` is the Portfolio installation directory, as specified at installation time.



WebLogic Server 7.0 Deployment Page.

In the **Step 3.** section choose an available server and add it to the Target Servers box by pressing the right arrow. As a final step, press the **Configure and Deploy** button in the **Step 5.** section.

The next page will guarantee a successful deployment if the Status at the bottom of the screen displays a message that says "Running".

2.5 Oracle9i Application Server

We support installation under Oracle9i v9.0.2 and v9.0.3 Application Servers. The deployment procedures for Oracle9i 9.0.2 and 9.0.3 are quite similar. The major difference is that on Oracle9i 9.0.2 you will be deploying the EJB1.1 version of Portfolio, whereas on Oracle9i 9.0.3 you will be able to install the EJB2.0 version as well.

2.5.1 Oracle9i v9.0.2

In order to deploy the EJB1.1 version of Portfolio on your Oracle9i 9.0.2 Application Server, follow these three simple steps:

1. **Change current directory to the Oracle9i Application Server installation path.** Usually it is located in the *j2ee/home* subfolder of your Oracle9i DBMS home

directory. You can identify this folder by looking for a file named *oc4j.jar* (Oracle Containers for J2EE).

2. **Start Oracle9i v9.0.2.** In a command prompt window, type the following line:

```
java -jar oc4j.jar
```

3. **Deploy Portfolio.** In another command prompt window, type the following two lines:

Line 1

```
java -jar admin.jar ormi://localhost admin "password" -deploy -file  
  "[ROOT_DIR]/Portfolio/Deployment/Oracle9i/Portfolio.ear"  
  -deploymentName "Portfolio"
```

Line 2

```
java -jar admin.jar ormi://localhost admin "password"  
  -bindWebApp Portfolio Portfolio http-web-site Portfolio
```

Note that *password* is your administration password and *[ROOT_DIR]* is the installation directory of Portfolio.

2.5.2 Oracle9i v9.0.3 (J2EE1.3 certified)

In order to deploy the EJB2.0 version of Portfolio on your Oracle9i 9.0.3 Application Server, follow these three simple steps:

1. **Change current directory to the Oracle9i Application Server installation path.** Usually it is located in the *j2ee/home* subfolder of your Oracle9i DBMS home directory. You can identify this folder by looking for a file named *oc4j.jar* (Oracle Containers for J2EE).
2. **Start Oracle9i v9.0.3.** In a command prompt window, type the following line:

```
java -jar oc4j.jar
```

3. **Deploy Portfolio.** In another command prompt window, type the following two lines:

Line 1

```
java -jar admin.jar ormi://localhost admin "password" -deploy -file  
  "[ROOT_DIR]/Portfolio/Deployment Ejb2.0/Oracle9i v9.0.3/Portfolio.ear"  
  -deploymentName "Portfolio"
```

Line 2

```
java -jar admin.jar ormi://localhost admin "password"  
-bindWebApp Portfolio Portfolio http-web-site Portfolio
```

Note that *password* is your administration password and *[ROOT_DIR]* is the installation directory of Portfolio.

2.6 Sun ONE AppServer 7

Starting the Application Server

On Windows, click the Windows Start button, then choose **Programs→Sun ONE Application Server 7→Start Application Server**. If you are working under Unix, change the current directory to *\$SUN_HOME/bin*, where *\$SUN_HOME* is the Sun ONE installation path and type the following line at command prompt:

```
asadmin start-appserv
```

Starting the Administration Console

After the Sun ONE server initializes, open an Internet browser and type in the following address:

```
http://localhost:4848/admin
```

where *localhost* can be replaced by the hostname of your server running Sun ONE, if you are connecting from a different machine. At the login screen enter the administrator user name (“admin” by default) and type in the login password. The browser will bring up the Sun ONE Administration Console.

Deployment

Choose **Applications→Enterprise Apps** in the left-hand panel and click the **Deploy...** button in the page on the right. Browse the *[ROOT_DIR]/Portfolio/Deployment Ejb2.0/Sun ONE/Portfolio.ear* file where *[ROOT_DIR]* is the Portfolio installation directory and click OK. Press again OK in the next screen and allow a couple of minutes to complete the deployment process.

2.7 Borland Enterprise AppServer 5

In order to deploy this Application inside Borland AppServer 5.0, you will need to perform the following tasks in the following order:

1. **Start the Borland Enterprise Server**

Under Windows you will click the **Borland Enterprise Server→Server** icon from the

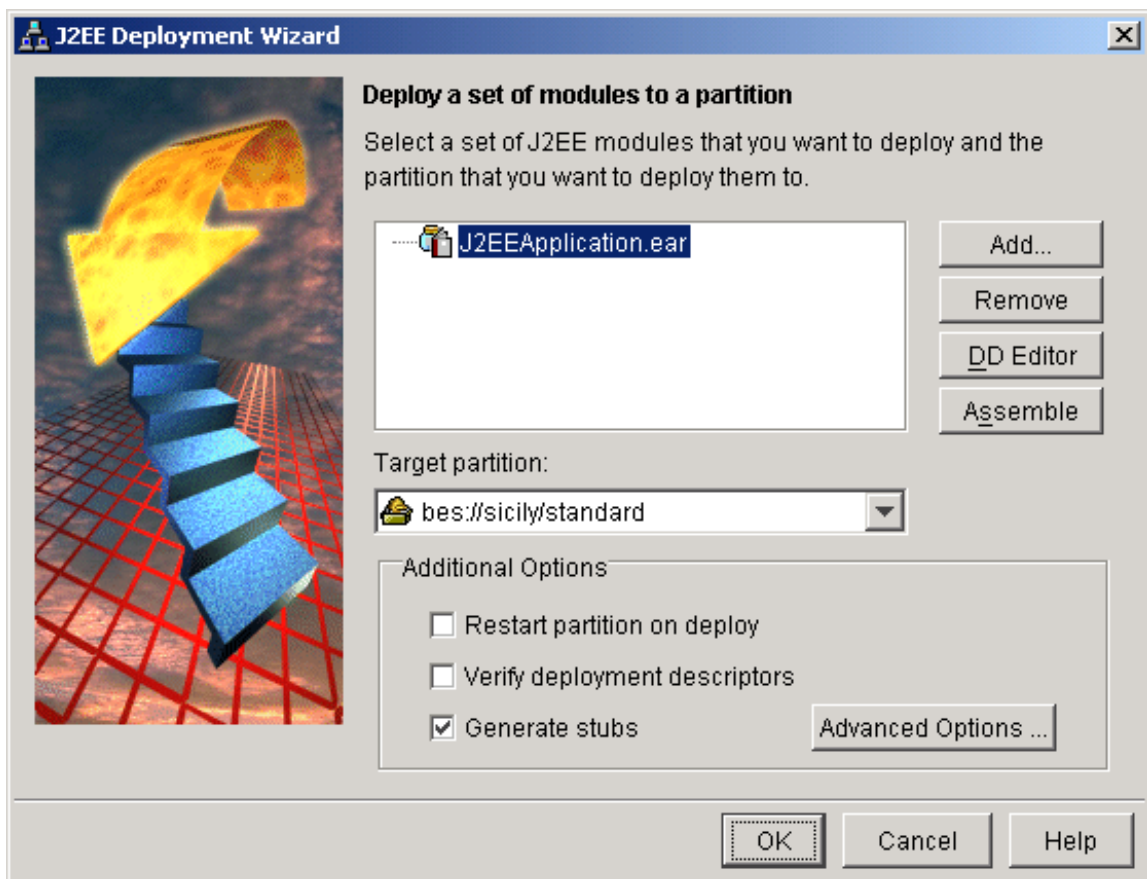
Start menu and wait a couple of seconds for the server to initialize. Under Unix type: `bin/ias` at the command prompt from the installation directory.

2. Open the Administration Console

You can open the Borland Enterprise Administration Console by clicking the **Borland Enterprise Server**→**Console** icon from the Start menu, if you are running under the Windows operating system. A login dialog will pop up, where you can click the **Login** button without making any modifications at all. Under Unix type: `bin/console` from the installation directory.

3. Deploy Your J2EE Modules

Select **Tasks**→**Deployment**→**Deploy J2EE modules to a partition...** to open the J2EE Deployment Wizard. Click the **Add** button and select the **Portfolio.ear** archive from the `[ROOT_DIR]/Portfolio/Deployment Ejb2.0/Borland AppServer 5` folder, where `[ROOT_DIR]` is the root directory where Portfolio was installed.



Borland AppServer 5 Deployment Dialog.

At the bottom of this dialog, turn on the **Generate Stubs** option and click **Next**. The deployment process will scroll inside a text area. When it completes, press the **OK** button.

2.8 Sybase EAServer 3.6

Deployment inside EAServer 3.6 is performed using the Jaguar Manager inside a running Jaguar Server.

1. **Start you Jaguar Server**

Under Windows, open Explorer, navigate to the EAServer installation folder and run the `serverstart_jdk12.bat` file by double-clicking it. If you are running the Jaguar Server under Unix, change your current directory to the EAServer installation path and type in the following line at command prompt:

```
./srvstart_jdk12
```

2. **Deploy to your Jaguar Repository**

Open the Jaguar Manager and connect to the previously started Jaguar server. Right-click with your mouse the **Packages** folder under **Sybase Central Java Edition**→**Jaguar Manager** and select **Deploy**→**EJB 1.1 Jar** from the popup menu. In the **Deploy Wizard** dialog, browse the `[ROOT_DIR]/Portfolio/Deployment/Sybase/Portfolio.ear` file⁷ and click **Next**. Wait for a few seconds in order to allow the deployment to take place and press the **Close** button when done.

3. **Deploy Inside your Jaguar Server**

From the Jaguar Manager, right-click the **Installed Packages** folder from the left panel and select **Install Package...** in order to bring up the **Package Wizard** dialog. Click on the **Install an Existing Package** button and select **Portfolio** from the list. Press **OK** to complete deployment inside EAServer.

2.9 Ironflare Orion Server

We support installation under Ironflare Orion 1.5.2, 1.5.4 and 1.6 Servers. The deployment procedures for Orion 1.5.2 and 1.6 are quite similar. The major difference is that on Orion 1.5.2, you will be deploying the EJB1.1 version of Portfolio, whereas on Orion 1.6 you will be able to install the EJB2.0 version as well.

2.9.1 Orion 1.5.2 (and Orion 1.5.4)

In order to deploy the EJB1.1 version of Portfolio on your Orion 1.5.2 Server, follow these three simple steps:

1. **Change current directory to the Orion Server installation directory.** You can identify it knowing it contains a file named *orion.jar*.
2. **Start Orion Server 1.5.2** In a command prompt window, type the following line:

⁷`[ROOT_DIR]` is the root directory where Portfolio was installed.

```
java -jar orion.jar
```

3. **Deploy Portfolio.** In another command prompt window, type the following two lines:

Line 1

```
java -jar admin.jar ormi://localhost admin "password" -deploy -file  
  "[ROOT_DIR]/Portfolio/Deployment/Ironflare Orion 1.5.x/Portfolio.ear"  
  -deploymentName "Portfolio"
```

Line 2

```
java -jar admin.jar ormi://localhost admin "password"  
  -bindWebApp Portfolio Portfolio default-web-site Portfolio
```

Note that *password* is your administration password and *[ROOT_DIR]* is the installation directory of Portfolio.

2.9.2 Orion 1.6 (EJB2.0 compliant)

In order to deploy the EJB2.0 version of Portfolio on your Orion 1.6 Server, follow these three simple steps:

1. **Change current directory to the Orion 1.6 Server installation path.** You can identify this folder by looking for a file named *orion.jar*.
2. **Start Orion Server 1.6** In a command prompt window, type the following line:

```
java -jar orion.jar
```

3. **Deploy Portfolio.** In another command prompt window, type the following two lines:

Line 1

```
java -jar admin.jar ormi://localhost admin "password" -deploy -file  
  "[ROOT_DIR]/Portfolio/Deployment Ejb2.0/Orion 1.6/Portfolio.ear"  
  -deploymentName "Portfolio"
```

Line 2

```
java -jar admin.jar ormi://localhost admin "password"  
  -bindWebApp Portfolio Portfolio default-web-site Portfolio
```

Note that *password* is your administration password and *[ROOT_DIR]* is the installation directory of Portfolio.

2.10 JBoss Open Source Application Server

We provide deployment instructions and resources for JBoss 2.4.4 (with TomCat 4.0.1) and 3.0.0 (with TomCat 4.0.3). Under JBoss 3.0.0 or later you will be deploying the EJB2.0 version of Portfolio. All previous JBoss versions will only support the EJB1.1 Application.

2.10.1 JBoss 2.4.4 (and JBoss 2.4.3)

Start the JBoss Server by changing the current directory to *[JBOSS_HOME]/jboss/bin*, where *[JBOSS_HOME]* is the installation path for your JBoss 2.4.4 release. Under Windows, type at command prompt:

```
run.bat
```

If you are running JBoss under Linux, you will type in the following:

```
./run.sh
```

and wait for a few seconds until the JBoss server is up and running.

In order to deploy Portfolio, you will need to make a copy of the **Portfolio.ear** Enterprise archive located inside the *[ROOT_DIR]/Portfolio/Deployment/JBoss* directory into the *[JBOSS_HOME]/jboss/deploy* folder⁸.

2.10.2 JBoss 3.0.0 (EJB2.0 compliant)

Start the JBoss Server by changing the current directory to *[JBOSS_HOME]/bin*, where *[JBOSS_HOME]* is the installation path for your JBoss 3.0.0 release. Under Windows, type at command prompt:

```
run.bat
```

If you are running JBoss 3.0.0 under Linux, you will type in the following:

```
./run.sh
```

and wait for a few seconds until the JBoss server is up and running.

In order to deploy Portfolio, you will need to make a copy of the **Portfolio.ear** Enterprise archive located inside the *[ROOT_DIR]/Portfolio/Deployment Ejb2.0/JBoss 3.0.0* directory into the *[JBOSS_HOME]/server/default/deploy* folder⁹.

⁸*[ROOT_DIR]* is the root directory where you installed Portfolio.

⁹*[ROOT_DIR]* is the root directory where you installed Portfolio.

Chapter 3

Mathematical Documentation

Within this component we offer a though and flexible implementation of a number of ideas and model frameworks from from Portfolio¹ Theory. Portfolio Theory in general deals with the interrelation between risk² and return³ for Portfolios which are constructed from a given collection of assets. In particular, within our component we allow the user to apply Markowitz Theory and the Capital Asset Pricing Model (CAPM). Within this chapter of the PDF documentation we describe the theoretical content and application of these theories.

3.1 Assumptions, Questions, Functionality and Problems

The construction of Markowitz Theory and the Capital Asset Pricing Model (CAPM) is based upon a number of assumptions concerning the investment market, the behavior of the participants of these markets and the assets from which the investors portfolio can be constructed.

3.1.1 Assumptions of Markowitz Theory

The assumptions underlying Markowitz Theory from which the model is derived is as follows:

1. Investors seek to maximize the expected return of total wealth.
2. All investors have the same expected single period investment horizon.
3. All investors are risk-adverse, that is they will only accept greater risk if they are compensated with a higher expected return.

¹A Portfolio is just collection of assets.

²Also known as the volatility. It is a measure which determines the degree to which an asset's return (or sometimes price) fluctuates.

³The absolute or relative (i.e. percentage) change in the price of an asset, that is, we speak about the return of an asset.

4. Investors base their investment decisions on the expected return and risk (i.e. the standard deviation of an assets historical returns).
5. All markets are perfectly efficient (e.g. no taxes and no transaction costs).

3.1.2 Assumptions of Capital Asset Pricing Model (CAPM)

The CAPM is an extension of the Markowitz Theory and hence requires all of its assumptions. The addition to these assumptions it also requires that the investors is able to lend or borrowing (risk free) cash from the market in order to leverage or un-leverage a portfolio. In particular, we require the following provisions:

1. **Lend excess capital at the Market Rate:** The investor may lend money at the prevailing market rate. This constitutes the ability to hold within the portfolio a risk free asset which will provide the prevailing return available on cash. In practice, such assets are often referred to as a money market accounts and will yield a return in the region of LIBOR.
2. **Borrow capital at the Market Rate:** The investor may borrow money from the market at the prevailing market rate in order to invest within (risky) assets. This will increase the expected return of the original capital base and also increase its risk. In practice, the rate at which money can be lent from the market by a fund will be in the region of LIBOR (at least if the fund structure the loan as a REPO type agreement).

Remark The rate of which money can be borrowed or lend from or to the market is the same.

3.1.3 What problems do these Models address

Though it may not be obvious from the assumptions of these models but using the theories based upon these assumptions we are able to select the optimal portfolio which balances the risk/reward profile in accordance with the investors risk tolerance or reward requirements. The principle questions which we are able to address within each of the models are as follows:

- **Markowitz Theory** - Allows the optimal portfolio constructed from a collected of assets to be selected when the risk, expected return or investors utility function is known.
 1. What is the portfolio which can be constructed from a given set of assets which has the lowest risk for a given value of the expected return? (See the :Portfolio:Markowitz:efficientFrontier(double, int): :Portfolio:Markowitz:efficientFrontier(double, double[], double[], double): methods from the Markowitz EJB Component).

2. What is the portfolio which can be constructed from a given set of assets which has the greatest expected return for a given value of the total risk? (See the `:Portfolio:SolveFrontier`: EJB Component and in particular the methods `:Portfolio:SolveFrontier:findReturn(double, double[], double[])`: and `:Portfolio:SolveFrontier:findReturn(double[], double[], double[], double[], double)`: of that EJB Component)
 3. What is the portfolio(s) which can be constructed from a given set of assets which is optimal with respect to the investors risk/reward utility function? (See the `:Portfolio:Markowitz:optimalPortfolio`: and `:Portfolio:Markowitz:optimalPortfolioMaxEx` methods from the Markowitz EJB Component).
- **Capital Asset Pricing Model (CAPM)** - Allows the optimal portfolio constructed from a collection of assets with the option of either lending or borrowing cash, to be selected when the risk, expected return, weighting of the Market Portfolio are known.
 1. What is the portfolio which can be constructed from a given set of assets with the option of borrowing or lending cash at the prevailing market rate, such that the total risk is minimized for a given value of the expected return? (See `:Portfolio:CapitalMarket:weightCML`: to construct the optimal portfolio and `:Portfolio:CapitalMarket:riskCML`: to evaluate its total risk from the `:Portfolio:CapitalMarket`: EJB Component).
 2. What is the portfolio which can be constructed from a given set of assets with the option of borrowing or lending cash at the prevailing market rate, such that expected return is maximized for a given value of the total risk? (See `:Portfolio:CapitalMarket:returnCML`: to evaluate associated expected return and then `:Portfolio:CapitalMarket:weightCML`: to construct the optimal portfolio from the `:Portfolio:CapitalMarket`: EJB Component).
 3. What is the weighting of the Market Portfolio (i.e. non-cash part) within the optimal portfolio given by its expected return when the portfolio can be constructed from a given set of assets with the option of borrowing or lending cash at the prevailing market rate? (See `:Portfolio:CapitalMarket:weightCML`: in order to evaluate the weighting and then `:Portfolio:CapitalMarket:riskCML`: to evaluate its risk from the `:Portfolio:CapitalMarket`: EJB Component).

3.1.4 Range of Functionality contained within the `:services`:

This Component contains the following business `:services`:`i/p`

- `:Portfolio:AssetParameters`: - Auxiliary class which offers methods to assist in the evaluation and estimation of various parameters which are then used within the methods of the main classes.
- `:Portfolio:CapitalMarket`: - Implements the Capital Asset Pricing Model (CAPM). The CAPM is an extension of the Markowitz theory in that in the construction of an

optimal portfolio along with (risky) assets you may borrow or lend (zero risk) cash at the prevailing market rate.

- `:Portfolio:Interpolation:` - Offers methods by which the Efficient Frontier can be constructed from a finite set of points.
- `:Portfolio:Markowitz:` - Implements the Markowitz model, that is, we offer method which allow the portfolio with the least return to be constructed from a collection of assets.
- `:Portfolio:PerformanceEvaluation:` - Offers a number of procedures for accessing the return and risk adjusted return (Treynors Measure, Sharpes Ratio).
- `:Portfolio:SolveFrontier:` - This class complements the methods found within the Markowitz class which allow you to find for a given expected return the corresponded portfolio on the Efficient Frontier. Here we also allow yo to provide the value of the total risk and we will find the corresponding values of the expected return of the portfolio on the efficient frontier.
- `:Portfolio:TwoAssetPortfolio:` - Evaluation of the optimal weighting of a portfolio with two assets which can be used to analysis the effect of a single purchase or sale from an arbitrary portfolio.
- `:Portfolio:Volatility:` - Auxiliary class which offers methods to assist in the evaluation of the volatility, variance and covariance of the assets.

3.1.5 Problems with the Application of the Markowitz Theory and CAPM

When applying the Markowitz Theory and CAPM to real world problems concerning the construction of portfolios, we will in general be faced with the following difficulties.

Estimation of the Parameters

Problem: To apply any of these models we requires knowledge of the expectation and variance of the return for every available investment and the covariance between every pair of investments. This information in practice is not obtainable. To get around this we use the historical rates as estimates, the idea being that in the near term at least, the rates will not vary significantly from there historical rates.

Our Approach: Within our Component we have offered a number of utility `:services:` which assist in the evaluation and estimation of these quantities. The major of these auxiliary methods will be found within the `:Portfolio:AssetParameters:` EJB Component. With this EJB Component we offer methods for the estimation of the expected returns, variance and covariance in accordance with a historical and scenario based approach.

Scaling as the Number of Assets considered increases

Problem: Both the Markowitz Theory and CAPM application becomes computationally unyielding when a large number of investments are considered. For instance, on a typical desktop machine when 1 or 2 hundred assets are used then the application of these models will take a number of minutes; however if thousands of assets are used then the application of the model will take hours or even days.

Our Approach: Both the Markowitz and CAPM require the maximum of a given function to be determined. This requires that a low level multi-dimensional optimization algorithm is applied. These algorithms by there very are computational intension and hence the computational intensity of the application of the Markowitz and the CAPM cannot to completely avoided. However, we have mitigated the computational requirements of the application of this Component by:

1. **Refined Algorithm** - The optimization algorithm used here is an adapted version of Rosen's Algorithm that has been optimized to handle the particular optimization problems which are generated from the Markowitz Theory and CAPM. This custom algorithm was developed from the basis of our experience and developed algorithms found within the WebCab Optimization Component.
2. **This Components Design** - The Component is designed so that the Efficient Frontier and the Market Portfolio are constructed at a finite number of points after which the interpolation methods are applied. By designing the Component in this way you are able to call the optimization algorithms maybe 5 or 10 times in order to be able to construct the optimal portfolio for a wide range of possible values of the risk, expected return or Investors risk/rewards preferences.

Moreover, the Component is actually designed so that the computation of the Efficient Frontier and the Market Portfolio which depend on the optimization algorithms can be completed at the beginning of the application of these models, and these algorithms will not need to be used during the subsequent application of the models. Therefore, within real life situations the closing prices and other market data can be collected from which the Efficient Frontier and Market Portfolio can be constructed using an over-night batch process. Once these objects have been constructed the application of the Markowitz and CAPM even for portfolios which can be constructed from 1,000's of assets can be carried out using a desktop machine in essentially real time.

Determining the Risk/Reward preferences of the Investor

Problem: Within the application of the Markowitz Theory when selecting the optimal portfolio with respect the Investor's risk tolerance it is necessary to provide the Investors Utility function which describes there riskpreferences. This Utility function allows a unique portfolio to be selected from the collection of portfolios which offer the least risk for a given

expected return. The difficulty here like with other non-observable quantities is in our ability to estimate this function from information which is provided by the investor.

Our Approach: The difficulty in obtaining the Investors Utility Function is in providing a framework which is both comprehensive and user friendly. Ideally, the Investors would be in a position to submit a function (in analytic form or otherwise) which relates the risk and expected return in accordance with his preferences. However, this requirement and resulting construction of the Utility function is not sufficiently intuitive in order to be widely applied. The approach we have taken here is to allow the Investor to give his risk/reward preferences at a finite number of sweet spots around which we will interpolate in order to obtain the Utility function itself. Clearly, one of the assumptions we are forced to make in this approach is that the Utility function is smoothly varying between the interpolation points given.

Remark The interval between and the location of the interpolation points should reflect nature of the investors preferences. In particular, in regions where the cubic spline interpolation function is likely to differ most from the (true) Utility function more interpolation points should be requested from the investor.

For further details concerning the application of this (and other) approaches please see the section [‘Discovering the Investors risk - return profile’](#).

Practicalities of Diversification

Problem: Though this problem is not particular to our (or others) implementations it is important to point out that in practice costs will be incurred when an asset is brought or sold, and the effects of diversification (in accordance with the Markowitz Theory and CAPM) will diminish as the number of assets considered within the collection of asset increases. Therefore in practice one should balance the benefits of diversification with the costs incurred in re-balancing a portfolio.

Solution: The level of costs will vary according to the type and quantity of assets being held within the portfolio. Generally speaking the more liquid the asset the lower the relative transaction costs will be for a given transaction size. Moreover, several studies have shown that it is possible to derive most of the benefits of diversification with a portfolio consisting of 12 to 18 holdings. That is, to be adequately diversified does not require 200 holdings in a portfolio which could incur significant trading costs.

Another approach to deducing the effects of trading costs is to use derivatives on the underlying assets rather than the assets themselves in order to rebalance the portfolios. The reason being that derivatives trading generally has significantly lower costs than trading in the underlying security.

3.2 Preliminaries and Auxiliary EJB Component's

Within this section we details a number of auxiliary notions and the procedures with will be required by the main functionality offered by this Component.

Within the `:Portfolio:AssetParameters:` EJB Component we provide procedures for the evaluation of various quantities which are required within the application of this Component. These parameters include:

- Estimate of the Volatility of an asset prices:
 1. Historical Approach: `:Portfolio:AssetParameters:volatility(double[])`:
 2. Scenario Approach: `:Portfolio:AssetParameters:volatility(double[], double[])`:
- Estimate Expected Return of an asset:
 1. Historical Approach: `:Portfolio:AssetParameters:expectedReturn(double[])`:
 2. Scenario Approach: `:Portfolio:AssetParameters:expectedReturn(double[], double[])`:
 3. ARCH based Approach: `:Portfolio:AssetParameters:archExpectedPriceEstimate:`
- Estimate Covariance Matrix of the collection assets:
 1. Historical Approach: `:Portfolio:AssetParameters:covarianceMatrix(double[][])`:
 2. Scenario Approach: `:Portfolio:AssetParameters:covarianceMatrix(double[], double[][])`:

Note Further methods for the estimation of the Volatility are provided within the `:Portfolio:Volatility:` EJB Component.

3.2.1 Choices in Approach and other issues

As listed about the main two approaches by which the expected return, volatility and covariance are estimates is the historical and scenarios based approaches. Here we motivate the essential differences between these methods. We also give a brief explanation of the ARCH procedure for the estimation of the expected returns along with advice concerning the number of historical values which should be used within the application of the historical estimate.

When should the scenario approach be used?

One such instance is when a takeover of a quoted company has been announced and the share price converges to almost the offer price in anticipation of the takeover being completed. In this scenario the more likely the takeover will be completed the closer the price will converge to the offer price. However, if the takeover breaks down then the price is likely to experience sharp moves to the price level found prior to the intended takeover being announced. By estimating the likely-hood of each scenario and the likely level of volatility resulting we are able to give a realistic forward looking estimate.

When should the historical approach be used?

If the market under consideration goes through seasonal or business cycles, or if a given company has transformed itself then the observations used in order to estimate the expected volatility should reflect these issues. For example, if company which was a diversified general industrial company has since refocused on certain key areas, then in terms of estimating its expected volatility from historical values it is reasonable to only consider the period after the company refocused.

The number of historical values which should be used

The number of historical values used here in order to estimate the volatility should reflect the length of the period over which a reliable estimate of the volatility is required. For example, if an estimate of the 1-month volatility is sort then it is reasonable to use at least the last 1-months historical values up to a few years of historical values.

ARCH type models for Estimate Returns and Volatility

The ARCH model in the general sense can be thought of as a (linear) weighted scheme. Here we allow the past historical asset prices to have a weighted effect, along with a weighted measurement of the expected value in accordance with the assets characteristic line. Note, that when we set all the weights of the historical values to zero this estimate reduces to an estimate based on extrapolation of the characteristic line.

When to use the ARCH approach

This method assumes that the asset under consideration exhibits a long term drift (in particular, linear correlation) with respect to some market index. Therefore, this estimate is particularly applicable to assets which have been observed to exhibit significant linearly correlation to market indexes such as the FT100, S&P500 etc.

3.2.2 Basic Formulae for the Return, Covariance, Correlation and Risk

Within this subsection we collect together a number of formulae which illustrate how the scenario and the historical approach can be used in order to either define or investigate relationships between the return, covariance, correlation or risk of a portfolio and its assets.

Definition of the Return of a Portfolio

Before we move onto the Markowitz Theory we should define exactly what we mean by the return of a portfolio. The return of a portfolio over a given period is simply the weighted arithmetic average of the returns of the individual assets. That is, the return r_p of a

portfolio over t periods is given by:

$$r_p = \sum_{i=1}^N x_i r_i(t) \quad (3.2.1)$$

where x_i is the weight for the asset i and $r_i(t)$ is the return on the asset i over the t periods. Note that by definition the weights x_i , for any portfolio must add up to 1, i.e. $\sum_{i=1}^N x_i = 1$.

Remark Note that this definition depends on knowledge in the returns of the individual assets. Therefore, if we use the forwarding looking scenario based approach to estimating the return of the asset then the above formulae will evaluate the forwarding looking (expected) value of the (expected) return of the portfolio. Similarly, if we had evaluated the historical returns using historical values then the above formulae would refer to the evaluation of the historical value of the return of the portfolio.

Evaluating the Covariance of Returns (Scenario Approach)

The covariance of returns is a statistical measure of how the returns of two assets are correlated. That is, to what degree do they move together. (In the following section we offer more motivation and show how the covariance and correlation of connected).

Using the forward looking scenario approach the covariance of returns σ_{ij} , between two assets i and j is given by:

$$\sigma_{ij} = \sum_{s=1}^N P_s [r_{is} - E(r_i)][r_{js} - E(r_j)] \quad (3.2.2)$$

where,

- r_{is} (respectively r_{js}) denotes the rate of return for the asset i (respectively j) in the state s
- $E(r_i)$ (respectively $E(r_j)$) is the expected return for the asset i (respectively j)
- P_s is the probability of the state s occurring

Our Implementation: An approach to estimating all the covariances between all the pairs of a collection of asset (i.e. the covariance matrix has been implemented within the method `:Portfolio:AssetParameters:covarianceMatrix(double[], double[][])`).

Correlation, Covariance and the Volatility (aka standard deviation)

Though knowledge of the Correlation is not directly used within the full application⁴ of the Markowitz and CAPM, the notion does play an important role. The principle reason being that the effects of ‘diversification’ (explained below) which Portfolio Theory uses can be

⁴In the case of a portfolio with two assets some of the results are phrased in terms of the correlation.

influences by the level of correlation between the assets from which the portfolios can be constructed. In particular, from prospective of Portfolio Theory if two assets are perfectly correlation then one would expect them to have the same level of expected return since otherwise the asset with the lower expected return could be excluded from the collection of assets. Below we further illustrate these points by giving the primary relation between the Correlation, covariance and standard deviation (or volatility).

The correlation coefficient ρ_{ij} is another measure of the relationship between two assets i and j . The correlation coefficient is a measure which lies within the closed interval $[-1, 1]$, where a value of -1 which depicts that the assets are perfectly negatively correlation meaning that there prices always move in opposite direction and 1 which depicts that the assets of perfectly positively correlation meaning that there prices always move in the same direction.

The Correlation, covariance and Volatility (also know as the standard deviation) are related by the following equation:

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \quad (3.2.3)$$

where σ_i and σ_j is the standard deviations of the returns of the assets i and j , and σ_{ij} is the covariance of the returns between the assets i and j .

Scenario Approach to the Correlation Coefficient and the Covariance

Since the covariance is given by (3.2.2), it only remains to find the two standard deviations (or volatility) σ_i and σ_j , in order to evaluate (3.2.3). The standard deviation of an asset is given by the following equation:

$$\sigma = \sqrt{\sum_{t=1}^N P_t [r_t - E(r)]^2} \quad (3.2.4)$$

where P_t is the probability of the t -th state occurring, r_t denotes the historical average return when the t -th state occurs and $E(r)$ is the expected (future) return of the asset.

Remark If we are considering only two assets A and B then a computationally efficient way to calculate the correlation coefficient between A and B over N periods is by using the following expression:

$$\sigma_{ij} = \frac{N \sum_{t=1}^N A_t B_t - \sum_{t=1}^N A_t \sum_{t=1}^N B_t}{\sqrt{\left[(N \sum_{t=1}^N A_t^2) - (\sum_{t=1}^N A_t)^2 \right] \left[(N \sum_{t=1}^N B_t^2) - (\sum_{t=1}^N B_t)^2 \right]}} \quad (3.2.5)$$

where A_t (respectively B_t) is the percentage increase of the asset A (respectively B) in the t^{th} period.

Evaluating the Expected Return(s) (Scenario Approach)

The following expression is often used to define (average) expected return $E(r)$ of an asset over some (future) time period:

$$E(r) = \sum_{s=1}^N P_s r_s \quad (3.2.6)$$

where P_s is the probability of the state s occurring and r_s is the corresponding expected return of that given state.

Our Implementation: Such an approach has been implemented within `:AssetParameters:expectedReturn(double[], double[])`:

Covariance of the Expected Return (Historical Approach)

Similarly, we may use an historical approach in order to estimate the covariance between the expected return of assets. That is, if we know the historical rates of return then the covariance is given by the expression:

$$\sigma_{ij} = \frac{1}{N} \sum_{t=1}^N (r_{it} - \bar{r}_i)(r_{jt} - \bar{r}_j) \quad (3.2.7)$$

where,

- N is the number of equally likely paired observations
- r_{it} (respectively r_{jt}) is the return of the asset i (respectively j) in the t^{th} period
- $\bar{r}_i$ (respectively $\bar{r}_j$) is the historical average return for the asset i (respectively j)

Our Implementation: The allow the covariance to be evaluated/estimated uses a scenario or historical approach within `:Portfolio:AssetParameters:covariance(double[],double[],double[])`; or `:Portfolio:AssetParameters:covariance(double[],double[])`: respectively. The evaluation of the covariance is only required when portfolio with only two assets are required. For portfolio with more than two assets you will be required to evaluate the covariance matrix of the collection of assets from which the portfolio can be constructed. Within the `:Portfolio:AssetParameters`: EJB Component we provide the methods `:Portfolio:AssetParameters:covarianceMatrix(double[][])`; and `:Portfolio:AssetParameters:covariance(double[],double[],double[])` which allow the covariance matrix to be evaluated via historical or a scenario based approach respectively.

Variance and Expected Return of a Portfolio with n holdings

We give the general formula for the **variance** $\sigma^2(r_p)$, of the returns for a portfolio of N assets:

$$\text{var}(r_p) = \sigma^2(r_p) = \sum_{i=1}^N x_i^2 \sigma_{ii} + \sum_{i=1}^N \sum_{j=1}^N x_i x_j \sigma_{ij} \quad \text{for } i = j \quad (3.2.8)$$

where x_i and x_j are the weights for the assets i and j , σ_{ii} is the variance for the i^{th} asset, σ_{ij} is the covariance of the assets i and j .

Remark The standard deviation of the portfolio σ_p , is always the square root of the variance (i.e. $\sigma_p = \sqrt{\text{var}(r_p)}$).

The **expected return** $E(r_p)$, is given by:

$$E(r_p) = \sum_{i=1}^n x_i E(r_i) \quad (3.2.9)$$

where $E(r_i)$ is the expected return of the i^{th} asset x_i and is the proportion of the portfolios value invested in the i^{th} asset.

Remarks

- If we hold a positive quantity of the asset i (i.e. go long) then clearly $x_i > 0$. We are able to model going short on an asset i , by letting $x_i < 0$. Therefore, if we set $x_i \geq 0$ then no short selling is allowed.
- For a portfolio with a large number of holdings the covariance terms will dominate the variance terms. Hence, the risk of the portfolio will depend more on the average covariance between the investments than on the risk of the investments themselves.

3.3 Expected Return and Risk from a Portfolio with two Assets

Now we will start Portfolio Theory proper! In that we will discuss formulae which detail the interaction between the risk and return of a portfolio and the assets from which it is constructed. Here we will deal with the special case of portfolios which consist of two assets.

3.3.1 Motivation

This case is particularly useful because the relevant formulae take a simple form and we are able to calculate the effect a single purchase (or sale) has to a portfolio by viewing the portfolio (or portfolio minus a holding) itself as a single asset.

In the case of portfolios with only two assets evaluation of the portfolio risk, expected return and optimal weights are all closed formulae. As mentioned above the usefulness of these closed formulae is that the effect of a single purchase (or sale) has to a portfolios risk/reward profile can be studied by viewing the portfolio (or portfolio minus a holding) itself as a single asset. Note that this portfolio which is viewed as a single asset could however have been constructed by use of the Markowitz (see :Portfolio:Markowitz: EJB Component) or CAPM (see :Portfolio:CapitalMarket: EJB Component) Theories.

3.3.2 Formulation for a Portfolio of two assets

If we can construct portfolios from two assets A and B , then the **general portfolio** P , will take the form:

$$P = \alpha E(R_A) + (1 - \alpha) E(R_B)$$

where $\{\alpha : 0 \leq \alpha \leq 1\}$, determines the weighting of the portfolio between the two assets A and B .

If the corresponding **expected returns** of the two asset from which portfolio can be construct are $E(R_A)$ and $E(R_B)$, then the expected return of the general portfolio $E(R_p)$ is:

$$E(R_p) = \alpha E(R_A) + (1 - \alpha) E(R_B) \quad (3.3.10)$$

with a **standard deviation** of:

$$\sigma_p = \sqrt{\alpha^2 \sigma_A^2 + (1 - \alpha)^2 \sigma_B^2 + 2\alpha(1 - \alpha) \sigma_{AB}} \quad (3.3.11)$$

By using the identity (3.2.3), $\sigma_{AB} = \sigma_A \sigma_B \rho_{AB}$, we can rewrite (3.3.11) as:

$$\sigma_p^2 = \alpha^2 \sigma_A^2 + (1 - \alpha)^2 \sigma_B^2 + 2\alpha(1 - \alpha) \sigma_A \sigma_B \rho_{AB} \quad (3.3.12)$$

3.3.3 Minimum risk of a portfolio with two assets

By direct calculation for a two-asset portfolio composed of the assets A and B , the portfolio with the minimum risk for some corresponding value of the expected return occurs when the weighting of the asset A (α) is equal to:

$$\alpha = \frac{\sigma_B^2 - \rho_{AB} \sigma_A \sigma_B}{\sigma_A^2 + \sigma_B^2 - 2\rho_{AB} \sigma_A \sigma_B}$$

where α is the percentage (in decimal format) invested in asset A within the two asset portfolio. Hence the percentage invested in the other asset B is $(1 - \alpha)$.

In particular, for the special cases when the assets are uncorrelated (i.e. knowledge of the movement of one asset either up or down does not imply anything concerning the movement of the other) and perfectly negatively correlated (i.e. the assets always move of opposite directions) we have:

- A and B are uncorrelated (i.e. $\sigma_{AB} = 0$) is given by:

$$\alpha = \frac{\sigma_B^2}{\sigma_A^2 + \sigma_B^2} \quad (3.3.13)$$

- A and B are perfectly negatively correlated (i.e. $\sigma_{AB} = -1$) is given when:

$$\alpha = \frac{\sigma_B}{\sigma_A + \sigma_B} \quad (3.3.14)$$

Concluding Remarks: Above we have constructed the portfolio under various conditions which exhibits the minimal risk which some expected return. However, often an investor will wish to balance the risk against the return which is the main question we will consider the following sections.

3.4 Markowitz Theory

The Markowitz Model is used to analyze the construction and qualitative nature of a portfolio's risk-return characteristics. In particular, we offer methods by which the continuum (in risk and expected return) of the portfolios (known as the Efficient Frontier) which consists of the portfolios with the minimum risk for a given value of the expected return constructed from a given collection of assets. Moreover, from this collection of Portfolios a unique optimal portfolio can be selected with respect to a given value of the expected return, risk or an investors risk - return profile given in terms of a utility function.

3.4.1 Overview of Markowitz Theory Implemented

The Efficient Frontier is the collection of portfolios constructed from the given set of assets which have the lowest possible risk for a given level of the expected return. Note that the weights of the assets making up the portfolio may themselves be subject to constraints (for example, no one asset can have a weighting more than 20 percent or less the 5 percent) which we will deal with below.

Once the Efficient Frontier is known, we are able to select from this continuum a unique portfolio which represent the optimal portfolio with respect to an investors risk - return profile. The return profile of the investor may be given in three distinct ways and the correspond optimal portfolio can them be constructed. The by one of the following means:

1. With respect to the investors risk - reward utility function, see `:Portfolio:Markowitz:setUtilityFunction` and `:Portfolio:Markowitz:setUtilityFunctionPoly`.
2. Maximum Risk - the investor gives the (maximum) risk which they are prepared to accept and then the corresponding portfolio with the highest expected return for that given level of risk is constructed.
3. Expected Return - the investor gives the expected return which they desire and the portfolio with the least risk for that given level of expected return is constructed.

3.4.2 Construction of the Efficient Frontier

For a given collection of securities with various risk profiles, an investor can construct a whole range of portfolios with various risk-return characteristics. In particular, there exists a continuous family of portfolios within the range of the risk and expectation of the individual securities. The investor will wish to choose a portfolio from this family which offers the lowest level of risk for a given expected return, according to the assumptions of

the Markowitz model.

We consider the family of all possible portfolios formed from a combination of the available securities. Within this family we choose a locus of points by selecting the portfolios which offer the least risk for a given level of the expected return. Such optimal portfolios will exist within the range of the expected returns. This locus of points is referred to as the **Efficient Frontier**.

The Efficient Frontier is constructed by the following steps:

1. Evaluated the Efficient Frontier at a finite number of points. That is find the portfolio which exhibit the lowest risk for a given expected return.
2. Interpolate about these points using cubic spline (or some other method) in order to construct the Efficient Frontier.

The points on the Efficient Frontier are portfolios constructed from the set of assets considered which exhibit the lowest risk for a given expected return. These portfolio are characterized by the following three characteristics:

1. Expected Return - The expected return of the portfolio which is estimated from the historical returns of the assets within the portfolio.
2. Total Risk - The total risk of the portfolio which is estimated from the historical returns of the assets within the portfolio.
3. Asset Weights - the weights of the collection of assets from which the portfolio can be constructed.

It is important to point out that the Efficient Frontier is a monotonically increasing function in risk and expected return. This means that if we are given a value of the expected return then there will correspond a unique portfolio on the Efficient Frontier with a given total risk. Conversely, if we are given the total risk of the portfolio then there will exist a unique portfolio on the Efficient Frontier with a corresponding value its expected return.

Constraining the Weights of the assets of the Efficient Frontier's Portfolios

Within our implementation we offer the possibility to constrain the weights of the assets from which the portfolios on the Efficient Frontier are constructed. The constraints on the weights on the portfolios are set by using the method `setConstraints`. We illustrate the use constraints with the following example.

Say an investor requires a portfolio selected from n asset which has the lowest risk for a given expected return but also has the requirement that all of the assets must have a weight between 0.05 and 0.1 (i.e. between 5 and 10 percent). In this instance we would need to set the constraints on the assets to be:

```
lowerBounds = { 0.05, 0.05, 0.05, ..., 0.05 }  
upperBounds = { 0.1, 0.1, 0.1, ..., 0.1 }
```

where each of the arrays above has the same number of terms as the number of assets from which the portfolio can be constructed. Constraints can be set on the assets from which the Efficient Frontier is constructed by using `:Portfolio:Markowitz:setConstraints:`, constraints can also be added within the context of the evaluation of the Efficient Frontier within the CAPM using `:Portfolio:Markowitz:setConstraints:`.

Remark: If the constraints are not set then they will take there default values which are 0 and 1, for the lower bound respectively upper bound of each asset. That is, they will remain as weights in the usual sense.

3.4.3 Constraints effect on the range of the Efficient Frontier

The placing of constraints on the weights of the assets effects the range of expected returns for which the resulting portfolios can be constructed. Since the (constrained) Efficient Frontier is just a collection of portfolios subject also subject to the constraints which minimize the risk for a given level of the expected return. The range of values over which the Efficient Frontier exists must correspond to the range of expected returns of the possible constructed portfolios.

Within the Markowitz and the CapitalMarket EJB Component's we provide the methods `maxFrontierReturn`, and `minFrontierReturn` we allow the maximum and respectively minimum values of the expected return over which the (possibly constrained) Efficient Frontier exists. We also offer two associated methods `maxFrontierReturnWeights` and `minFrontierReturnWeights`, which evaluate the assets weights of the portfolio at these two ends points. These two methods which construct the Portfolios on the Efficient Frontier at its end points have the significant advantage of having almost no computational overhead, unlike the construction of the portfolios on the Efficient Frontier at other points.

Also, note that since the Efficient Frontier is monotonically increase in risk and expected return, the therefore the point at which the maximum and minimum expected return on the Efficient Frontier occur is also the points at which the maximum respectively minimum of the risk of the portfolios on the Efficient Frontier occur.

Performance Issues

The introduction of constraints on the weights of the portfolios which form the Efficient Frontier will have the following consequences with regards to overall performance:

- **Upper Bounds** - the presence of upper bounds (not all equal to 1) on the asset weights will result in the construction of the corresponding Efficient Frontier requiring significantly more computational resources. If however the upper bounds are all set to 1 (i.e. there default value), then they will not effect the computational demands

required to evaluate the points on the Efficient Frontier. This allows lower bounds to be set on the asset weight without reducing the performance of this class.

- **Lower Bounds** - the presence of lower bounds on the asset weights will have no significant effects on the efficiency of the construction and analysis of portfolio on the Efficient Frontier.

Notes on the Evaluation of the Efficient Frontier and the Optimal Portfolio

To calculate the Efficient Frontier, Rosen's gradient projection optimization algorithm is used. If you directly try to evaluate the optimal portfolio with respect to an investors utility function then you will need numerous applications of Rosen's algorithm which will become computationally intensive. Therefore, we designed this class so that this would not be necessary by allowing the computation at the beginning a number of points on the Efficient Frontier, from which the other points will be deduced (in fact, estimated) through the use of cubic spline interpolation. These interpolation points are determined by `calculateEfficientFrontier`, which must be called prior to any subsequent method which depends on the Efficient Frontier being known.

The use of this approach will also allow us to deal with the possible situation.

3.4.4 Consistency of the Assets and Effects on the Efficient Frontier

This section deals with a rather technical issue concerning how the consistency of the collection of assets from which the Portfolio on the Efficient Frontier can be constructed. By consistency we mean consistency with the assumptions of Markowitz Theory and CAPM, and how these inconsistencies effect are ability the construct optimal portfolio in a neighborhood of the lower bound of the expected returns over which the Efficient Frontier exists. That is, how appropriate the portfolios constructed from the available ("inconsistent") assets are to issues concerning the selection of the optimal portfolio. Hence the selection of either portfolios with the lowest risk for a given expected return or portfolio with the highest expected return with a given maximum risk.

When these effects occur

Note that the effects we will consider with this section will have relevance in the following situations:

- Only effect a neighborhood of the lower bound of the expected returns over which the Efficient Frontier exists.
- Will only occur when the portfolio considered are constructed from collections of assets which are not consistent (see note below) the assumptions of Markowitz Theory and CAPM.

Remark Similar effects do not occur in a neighborhood of the upper bound of the expected return of the Efficient Frontier because at the upper bound by definition there do not exist any portfolios with a higher expected returns so for that level of returns there are no preferred portfolios.

Advice to the Reader

We advise the reader to skip this section at the first reading, and refer back if you encounter such effects with the collection of assets you are using in order to construct the portfolios within your given application.

Nature of Inconsistency

The way in which the collection of assets can be inconsistent lies in the fact that the Markowitz Theory (and as we will see later the CAPM) assume that the investor will only take on additional risk for a higher level of expected return. Therefore, in order for a collection of assets to be consistent with these assumptions there must exist a portfolio with a greater return if it has a higher risk. Unfortunately, this may not be the case if the assets do not obey the rule that for a higher level of the expected return the risk increases.

If such a portfolio exists then the Efficient Frontier should only be considered on a subset over the entire range of expected values on which it exists when selecting an optimal portfolio. In particular, we will need to increase the lower bound of the expected returns over which the Efficient Frontier is considered. The precise construction of the lower bound will in general involve analysis of the differential of the Efficient Frontier. We implement these procedures within the `MaxRange` EJB Component.

When Inconsistency will effect the Efficient Frontier application

He we detail when the effects of inconsistency of the historical source data will effect the Efficient Frontiers ability the select the optimal portfolio. We only offer conditions under which such effects are certain to take place. We refer the reader to the section on precise lower bounds for a precise statement under which this effect take place take place.

When the asset with the lowest expected return does not have an upper bound on its weight then this effect is certain to occur when:

‘the asset with the lowest expected return does not have the lowest risk and does not have a constraint placed on its upper bound’

In the case when this asset does not have an upper bound on its asset weight then these effects are certain to take place if:

‘The portfolio constructed by taking the maximum weight of the asset with the lowest expected return, then the maximum weight of the asset with the next

to lowest expected return and so on... until the sum of the weights is one. If there exists an asset from the remaining assets with a higher expected return with a lower risk.'

Illustration of the Effect of Inconsistency

We illustrate in the diagram below the shape of the Efficient Frontier which can occur if the collection of assets are not consistent.

Notes of Illustration: For all portfolios on the Efficient Frontier (i.e. red line) which have a risk which lies between C and D, or equivalently an expected return which lies between A and B; there exists a portfolio which has the same level of risk but has a high level of return. Moreover, for all points (except end point) there exists a portfolio with a lower risk and a higher expected return. Therefore, this section of the Efficient Frontier should be ignored with respect to the construction an optimal portfolio from the assets available (represented by small black squares in diagram).

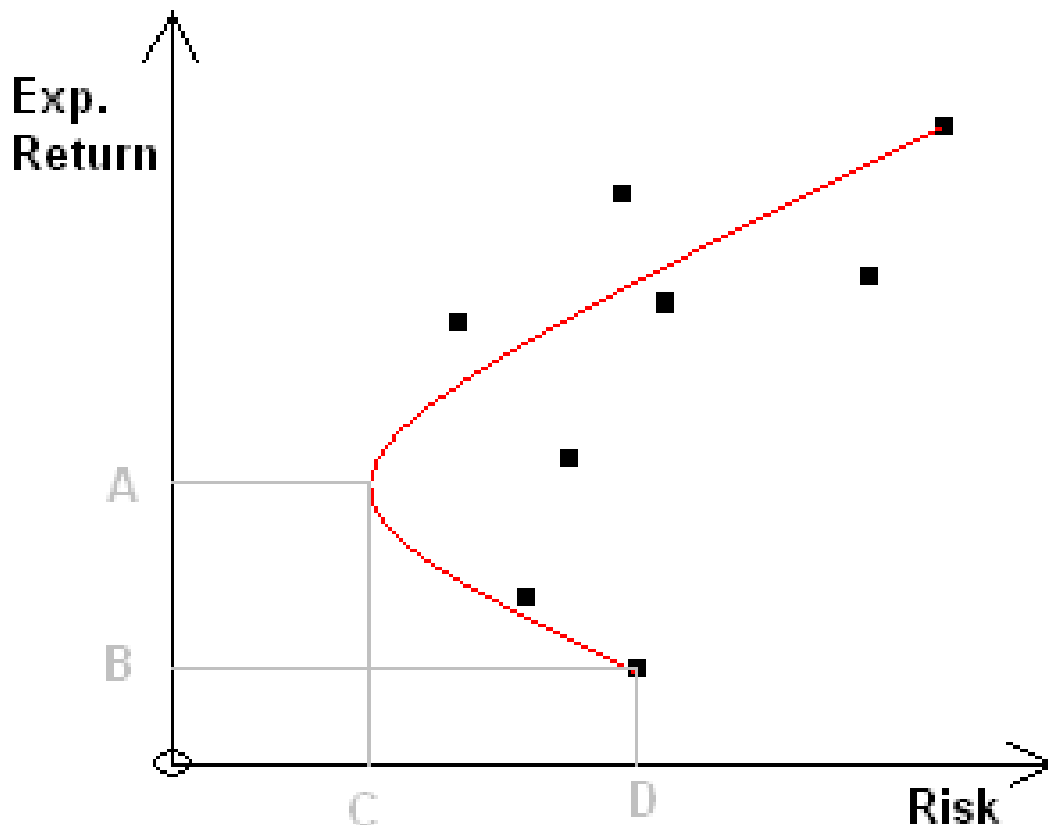


Fig: The loop-back effect at the lower end of the range of the expected return

Further Remarks: Note that in order to obtain the portfolios with the lowest respectively highest values of the expected return it is necessary to either construct a portfolio which consists solely of the asset with the lowest respectively higher expected return (we

assume that these assets weights are not constrained). For this reason there is a rationale for considering the Efficient Frontier at both of these points, since they are unique they must by definition be the portfolio with that given (exact) level of return with the lowest risk. However from the prospective of selecting the optimal portfolio corresponding to an expected return at A (in the above diagram), it is not appropriate.

Another motivation for including the portfolio corresponding the A, and similar non-optimal portfolios in the Efficient Frontier is that the portfolios at the extremes of the Efficient Frontier lie at the vertex of the arced region (see dark gray shading in diagram below) in which all the possible portfolios which can be constructed from the available assets lie.

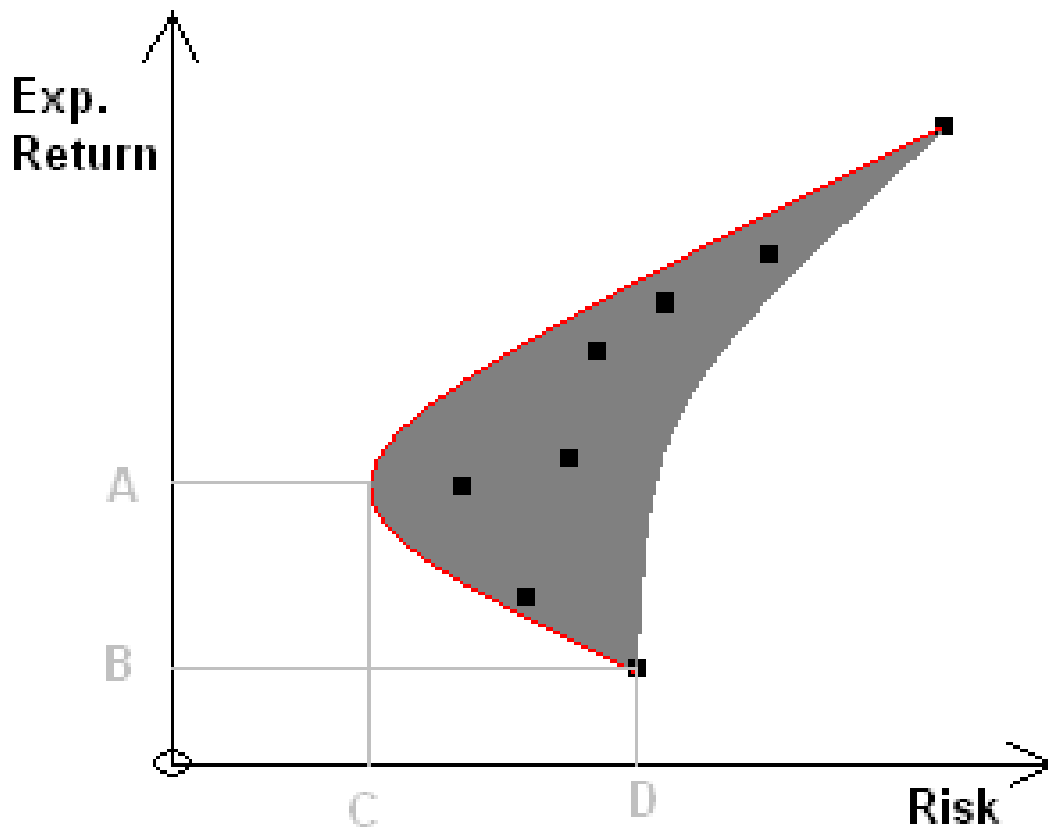


Fig: All possible portfolios lie within the dark gray region.

Approximate Lower Bound of the ‘Maximum Range’

Therefore, from the point of view of applying the Efficient Frontier to selecting the optimal portfolio we will need to evaluate a (new) lower bound greater than the bound over which the Efficient Frontier exists over which the Frontier does only contain (justifiable) optimal portfolios. Here we will provide a quick and easy approach which allows a lower bound to be evaluated. Note that this lower bound will often not be optimal.

No Upper Bound Constraint on asset with the Lowest Risk

In the case when there are no constraints placed on the weight of the asset with the lowest risk we are able to evaluate the (approx.) range of the expected return which should be used when constructing the optimal portfolio is given by:

- Upper Bound on the Expected Returns: The expected return of the asset with the highest expected return.
- Lower Bound on the Expected Returns: The expected return of the asset with the lowest RISK.

If you use this range of expected returns then the Efficient Frontier constructed will not display the problem of suggesting the below optimal portfolio. Note that in this instance if you require a portfolio below this lower bound of the expected return for which the exists an portfolio (i.e. an expected return is above the expected return of the asset with the lowest expected return and below the lower bound set). Then you should just select the portfolio on the Efficient Frontier with a risk equal to the risk of the asset with the lowest risk.

With Constraints

In the case when there are constraints on the weights of the assets (or at least an upper bound constraint on the asset with the lowest risk) then a construction based on similar principles applies. This construction is as follows:

- Upper Bound on the Expected Returns: The expected return of the asset with the highest expected return.
- Lower Bound on the Expected Returns: If you have an upper bound on the asset with the lowest risk then form a portfolio by taking the maximum weight of this asset then the asset with the next highest risk and so on... until the sum of the weights is one. Then the lower bound on the expected return should be the expected return of the portfolio constructed in this means.

These constructions have been implemented and offered as public methods within the `MaxRange` EJB Component.

Discussion and Illustration of this Approach

In the below diagram we illustrate this approach. Note that in this case the lower bound corresponding to a risk of F , is not optimal since the optimal points has an expected return of A , and risk of C . Also, note that only the Efficient Frontier to the right of the intersection will be considered (denoted by the arrow).

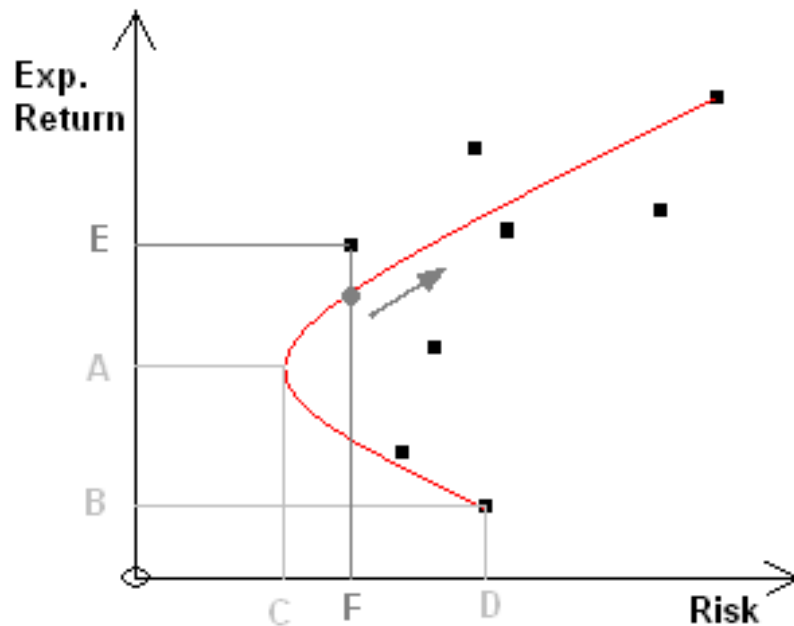


Fig: All possible portfolios lie within the dark gray region.

When we are interested in portfolios with an expected return above the lower bound provided by this approach then this approach is quite sufficient. However if we are interested in portfolios which are below this bound then we will require the more precise (and computationally demanding approach) given below. There are also instances in which the data is so inconsistent that this approach will return a lower bound which is equal to the higher bound. For an un-constrained asset set we illustrate this possibility within the following diagram.

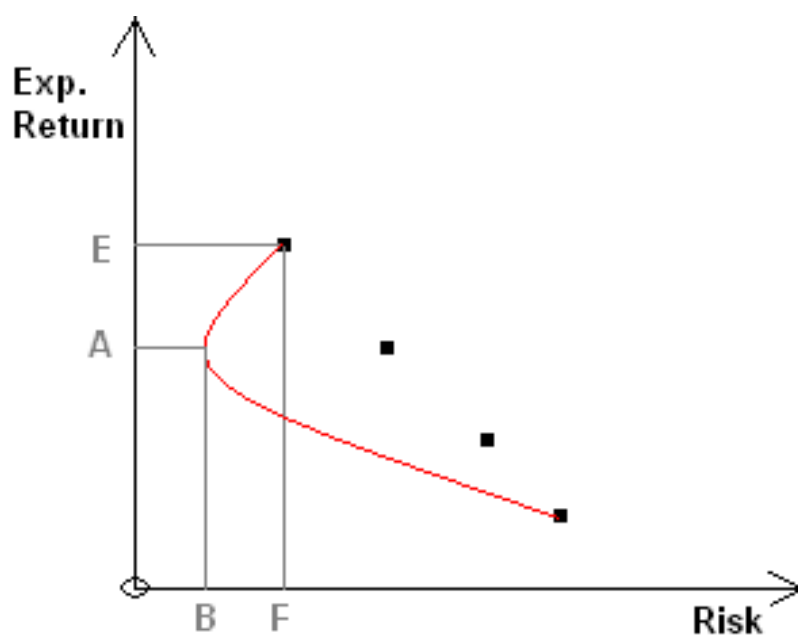


Fig: When the (approx) lower bound equals the upper bound.

Exact Evaluation of the Lower Bound of the ‘Maximum Range’

Though the above approach will yield an range over which all portfolios of the Efficient Frontier are optimal portfolios, the lower bound on the range of expected returns will not in general be optimal. In order to provide a precise evaluation of the lower bound of the expected return we will need to use an algorithm which considers the differential of the Efficient Frontier. In particular, we will need to construct the Efficient Frontier of the possibly constrained assets and evaluate unique point (if it exist) where the rate of change of the risk with respect to the expected return is zero. If this point is a minimum then the corresponding value of the expected return at that point is the precise lower bound over which the Efficient Frontier should be considered when used to construct an optimal portfolio.

We use this indirect (i.e. via Frontier) approach because even if the historical source data is not consistent with the assumptions of Portfolio Theory the above mentioned effects may not take effect. We offer procedures which deal with this case within the **MaxRange** EJB Component.

3.4.5 Selecting the Optimal Portfolio

The construction of a portfolio with regard to an investors risk - reward profile lies at the core of Markowitz Theory. By obtaining information concerning the value judgments to the various risk - reward combinations which will be available to the investor we are able to select the optimal (or preferred) portfolio for the continuum of portfolios on the Efficient Frontier. If we were not aware of a given investors risk preferences then it would not be possible to select a preferred portfolio on the Efficient Frontier. There are different level of granularity of the investors preferences which are sufficient in order to select an optimal portfolio. At one end the investor may only provide information concerning the maximal level of risk acceptable or the level of expected return desired from which an optimal portfolio can be selected. On the other hand the investors utility function may be given which offers much more fine grained information concerning this investors risk-reward preferences and allows a value judgment to be assign to a continuum of risk-reward combinations.

As mentioned above and within the over view of this section within Markowitz Theory there are essentially three ways in which to select an optimal portfolio from the selection of portfolios known as the Efficient Frontier which are each optimal for there corresponding value of the expected return or risk. The three mechanism for selection the optimal portfolio are as follows:

1. **Give value of the Expected Return** - Since the Efficient Frontier is monotonically increasing and continuous in the expected return a point and hence portfolio can be selected for a given value of the expected return over the range of the expected return. Once the (possibly constrained) Efficient Frontier has been constructed we are able to select an optimal portfolio by using `:Portfolio:Markowitz:efficientFrontier(double,int);`; alternatively you may wish to use the approach of `:Portfolio:Markowitz:findRisk(double,`

double[], double[]):

2. **Given value of the Risk** - Since the Efficient Frontier is monotonically increasing and continuous in the value of the risk a point and hence portfolio can be selected for a given value of the expected return over the range of the total risk. Once the (possibly constrained) Efficient Frontier is known we are able to select an optimal portfolio by using `:Portfolio:SolveFrontier:findReturn(double, double[], double[])`:
3. **Provide the Investors Utility Function** - The investors utility function is the locus of points at which the investor gets a particular level of satisfaction or utility from a combination of expected return and risk. This function may be a function with respect to the expected return or risk. By identifying the points at which the utility function and Efficient Frontier coincide we are able to identify a collection of portfolios which are optimal with respect to the investors expressed risk/return preferences. If the utility function is given as a function of the expected return then you can use `:Portfolio:Markowitz:optimalPortfolio(double, double, double[][])`, `:Portfolio:Markowitz:optimalPortfolioMaxExpected(double, double, double[][])`, or alternatively you may wish to use the approach; `:Portfolio:SolveFrontier:findRisk(double[], double[], double[], double[], double)`. If the utility function is given as a function of the total risk then the you will need to used `:Portfolio:SolveFrontier:findReturn(double[], double[], double[], double[], double)`. For further details concerning the details of our implementation we refer the reader to `:Portfolio:Markowitz:` or `:Portfolio:CapitalMarket:`

Since the Efficient Frontier is monotonically increasing in the expected return and risk the application and nature of selecting the optimal portfolio from knowledge of the expected return or risk is reasonably straightforward and we refer the reader to the `:Portfolio:Markowitz:` and `:Portfolio:SolveFrontier:` for further discussion. However, with regard to the use of the investors utility function there are a number of issues such as the means and equivalent of the ways in which the utility function are given and the means by which a solution is found which should be detailed further. In the following discussion we will treat each one of these issues in turn.

Providing the Investors Utility function

A utility function is the locus of points at which the investor gets a particular level of satisfaction or utility from a combination of expected return and risk. Clearly, each investor will have their own utility function depending on their individual trade-off between expected return and risk.

We provide methods by which the optimal portfolio can be selected from the Efficient Frontier when the investors utility function is given. Within our implementation you are able to provide the investors utility function in one of two ways:

- **Set of Interpolation Points** - The utility function is given on a finite set of points around which it can be interpolated in order to construct a continuous utility function. This can be done by using `:Portfolio:Markowitz:setUtilityFunctionInterp:`.

- **Given as a polynomial expansion by providing its coefficients** - We are able to construct a polynomial which represents the utility function if the coefficients of the polynomial are given. This can be done by using :Portfolio:Markowitz:setUtilityFunctionPoly:.

Structure of the ‘Polynomial’ and ‘Interpolated’ Utility Function

As mentioned above the investors utility function can be given as a set of interpolation points or in polynomial form. Here we provide further details as to the exact structure of the utility function (in either form) which will need to be provided within our implementation.

1. Structure of the polynomial which defines the Utility Function

The polynomial which defines the investors utility function takes the following form:

$$p(x) = coef[0] + (coef[1] * x_i) + \dots + (coef[n - 1] * x_{n-1})$$

where the $coef[i]$, $i = 0, \dots, n - 1$ are coefficients of the polynomial utility function which are provided as a parameter. Now within this representation the values at the variable x , corresponds to the risk level of a polynomial and the corresponding value of $p(x)$, is the value of the expected return which to investors demands in order to be exposed to the chosen level of risk.

2. Structure of the Interpolation Utility Function

The interpolation utility function is generated by interpolating a tabulated function which is provided as two arrays. The first array corresponds to an ordered sequence of the various total risk levels of the portfolio and is denote by $x[0.., n - 1]$ (with $x[0] < x[1] < \dots < x[n - 1]$). The first term of the second array corresponds to the expected return for the total risk $x[0]$. The second term of the second array corresponds to the expected return for the total risk $x[1]$. The third term is defined in a similar fashion and so on. This provides n coordinate points or equivalently a tabulated function which we can interpolate in order to provide a unique utility function which expresses the investors risk-reward profile.

The relationship between the Interpolation and Polynomial methods of setting the Utility Function

Note that the two means of setting the utility function namely the interpolation and polynomial procedures are closely related. Moreover we are able to roughly map between these two means of representing the investors utility function.

If the interpolation points are known at (x_i, y_i) , for $i = 0, 1, \dots, n - 1$ then we can construct the utility function given as a polynomial of order n , by solving the following n polynomial expressions which will allow us to deduce to values of the coefficients of the polynomial which takes the same values at the interpolation points:

$$p(x_i) = coef[0] + (coef[1] * x_i) + \dots + (coef[n - 1] * x_i^{n-1})$$

where $i = 0, \dots, n$ and $coef[i]$ are the coefficients of the utility function given in polynomial form. Alternatively, if we are given a polynomial $p(x) = y$, which represents the utility function of the investor then we are able to read off the interpolation points at $x_i : i = 0, \dots, n - 1$, by which the utility function can be defined in accordance with the interpolation approach. That is, the interpolation points (x_i, y_i) , for $i = 0, 1, \dots, n - 1$, for some $x_i, i = 0, \dots, n - 1$, are given by:

$$p(x_i) = y_i$$

where $i = 0, \dots, n - 1$.

The above procedure illustrates that there is a close relationship between the polynomial and interpolation ways of defining the utility function. However care should be taken to point out that though they are closely related they are not equivalent. The reason for this is that the interpolation method uses cubic spline interpolation in order to construct the utility function from the interpolation points. The polynomial method in general uses an n degree polynomial in order to define the utility function. Therefore, except in the case of the polynomial method using a cubic polynomial these two approaches can not represent a utility curve which is identical for all points. However, in practice the above procedure will result in a polynomial and interpolation representation which agrees on the interpolation points and is generally qualitatively and quantitatively very close for non-interpolation points.

3.4.6 Discovering the Investors risk - return profile

One final and in fact rather essential practical issue is how best to go about discovering the investors utility function which can naturally only be done by obtaining information from the investor themselves. However, information concerning the investors risk profile in a usable quantitative form is often difficult to obtain because the investor has no natural way in order to quantize his attitude to risk. Moreover, the nature (i.e. granularity) of the information concerning the investors risk - reward profile you are able to obtain will determine the form of the Markowitz Theory you are able to apply. For example, if you are only able to obtain either the investors maximum risk or a lower bound of the level of the expected return desired then you will only be able to obtain the portfolio on the Efficient Frontier which corresponds to the upper bound on the expected return (in the case of knowledge of the maximum risk) or a lower bound on the risk (in the case of knowledge of the lower bound of the expected return).

Here we suggest a few approaches which should assist in your attempt to discover an investors risk profile and translate this understanding into a usable quantitative form. Often the discovery of the investors risk profile will be closely related to the aims, objectives, financial situation and motivation of the investor. By understanding the investment driving forces better you will be able to better tailor an investment portfolio which suits there needs. Below we offer three scenarios and approaches which allow the investors risk profile

to be discovered sufficiently well that a unique optimal portfolio can be chosen from the continuum of portfolios on the efficient frontier. The approaches include:

1. **Investors attitude to differing losses:** What probability of a 5%, 10%, 15%, 20%, 30% loss can the investor live with? From such information you will be able to estimate the investors utility function if you assume that the distribution of asset returns is in accordance with a given distribution (such as the LogNormal distribution). The procedure required in order to estimate the investor utility function is to fit the distribution as closely as possible to the answers given. Since to distribution has been we are able to read off the values of the standard deviation (i.e. risk) for different values of the expected return (i.e. the utility function).
2. **Learning of the investors objectives:** Say an investor wishes to obtain a given sum at a future date, that is, his utility is very much focused at achieving a certain finance goal. Such instances come about with retirement planning where a given sum is desired at retirement and we wish to achieve this sum by taking to lowest risk in order to achieve this return.

In this instance you just need the investor to inform you of the return required. Once this is known you can select the portfolio from the efficient frontier given for this level of expected return. That is, the portfolio which can be constructed from the available assets which has the desired expected return with the lowest risk.

3. **Understanding the Investors Obligations:** An investors has a sum for which he wish to obtain the maximum return but also at the same time can not afford to lose more than a given amount. Such instances could occur within a large corporation which wishes to invest surplus cash within the market but cannot obtain loses of a certain level on this cash if it is to retain its present credit rating. In such instances the level of which the lose which the investor wishes to avoid at all possible costs should be stated. Then assuming the distribution of returns follows a given distribution (such as the LogNormal distribution) you will be able to state the change of such a loss occurring for a given portfolio on the efficient frontier.

The type of procedure performed above in order to ascertain the investors risk profile is in general referred to as calibration. Often the success of the application of a given model from the theory of quantitative finance will come down the success with which we are able to calibrate the parameters on which it depends.

3.4.7 Examples of selecting the Optimal Portfolio from the Investors Utility Function

Within this section we include a number of examples which illustrate how the optimal portfolio can be selected from the Efficient Frontier with the use of the investors utility function.

Cross Below Efficient Frontier

This is the most common example which you will encounter. The range of the risk of the utility function lies within a fairly narrow range implying that the investor is focused on controlling the risk of the portfolio and hence is not prepared to significantly increase the level of risk for significantly higher expected returns.

Note that if on the utility function the value of the risk is strictly positive when the expected return is zero. Then the utility function will always determine at least one optimal portfolio if there exists a value of the expected return such that the corresponding value of the risk of the Efficient Frontier is greater than the corresponding value of the risk of the investors utility function.

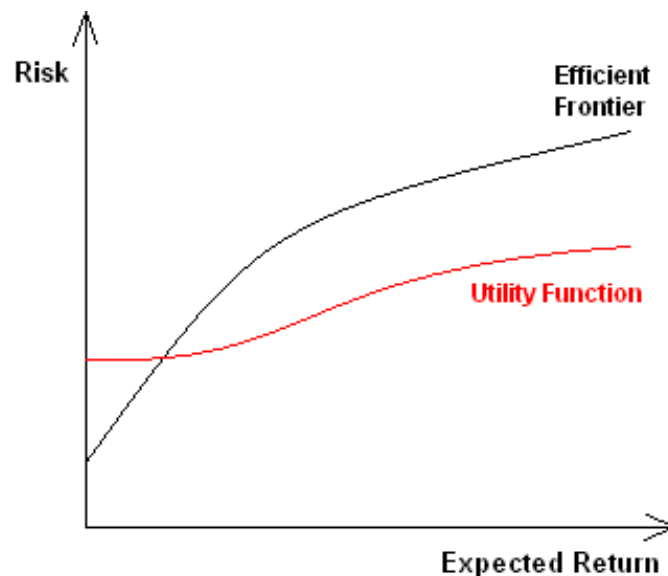


Fig: Typical Utility function shape.

Concave Utility Function

This example is fairly typical with the concave (i.e increasing positive gradient) Utility function usually resulting in an optimal portfolio with a relatively high value of the expected return. The reason being that the for relatively high values of the expected return the investor is prepared to take on proportionally larger amounts of risk in order to achieve a given increase in the expected return. Therefore, this utility functions shape would lead us to believe that the investor places paramount importance in obtaining a high level of expected return.

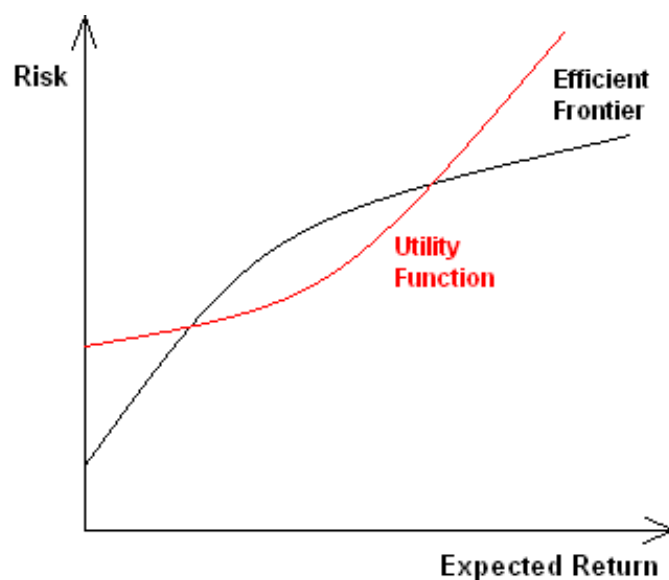


Fig: Concave Utility function resulting in a high expected return.

Cross Above Efficient Frontier

In this case the fact that the utility function for zero risk take of positive value of the expected return implies that the investor is in theory content to hold a cash portfolio and will only invest in risky assets if they match his risk-reward expectations. In order to meet these expectation the utility function must become equal to or greater than the Efficient Frontier for some value of the expected return.

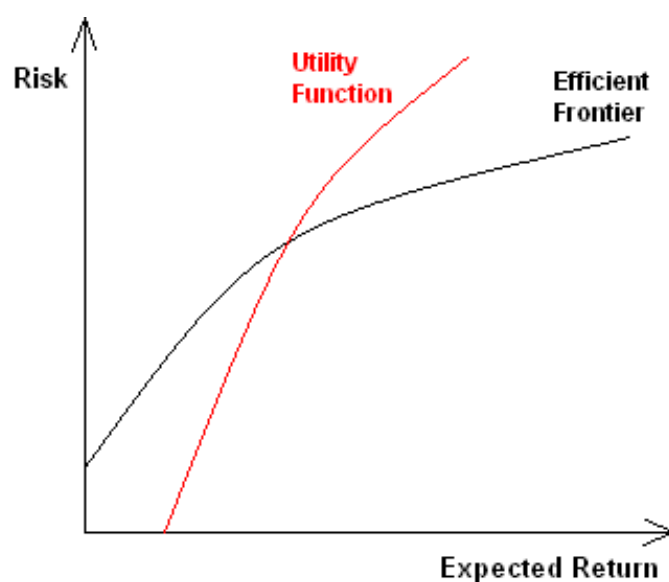


Fig: Possibility of holding cash.

Utility Function as a function of Risk

The following example was included to illustrate how the methods provided within the `:Portfolio:SolveFrontier:` class allow the use of utility functions which are functions of risk (rather than the expected return). What this means in practice is that you are able to investigate the investors risk-reward profile by asking the required expected return for a level of risk. If the investors provides this information for a range of values of the risk then you will be able to construct the (risk) utility function. We use the term ‘(risk) utility function’ because by collection the investors profile in this form we are only certain that the result utility function will be a function in risk. That is, given a value of the risk over the range for which the Utility function is given there exists a unique corresponding value of the expected return. However, if we select a value of the expected return there may not exist a corresponding unique value of the risk of the utility function constructed by this means.

In the following diagram we provide a example of a (risk) utility function).

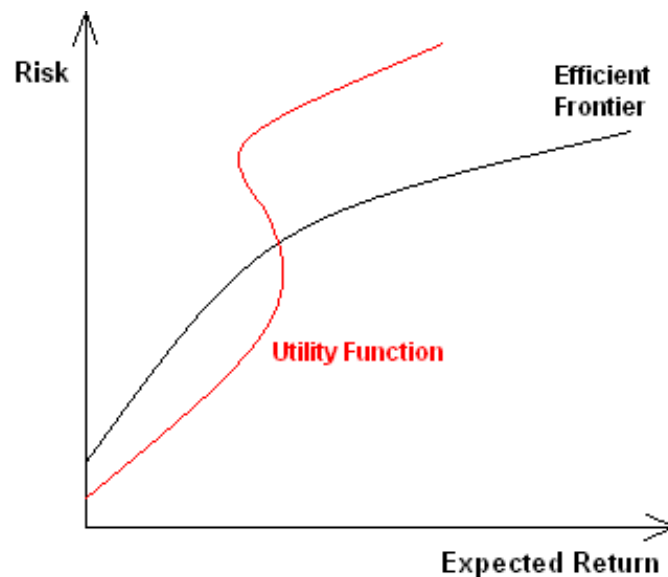


Fig: (risk) Utility Function

3.5 Capital Asset Pricing Model (CAPM)

3.5.1 Overview

The CAPM allows the investors portfolio to be constructed from risky assets and ‘cash’ holdings or borrowings which are either lent or borrow from the market at some prevailing rate. These additional options of allowing the portfolio to either borrow or lend cash at a prevailing market rate accurately reflects the possibilities available and practices used with the management of portfolios. That is, when a managed portfolio has excess capital it will typical be lent to the market at the prevailing rate (which is typically around LIBOR).

Conversely, if the portfolio wishes to increase its positions (and hence expected return) above the capital available within the fund then it will often borrow money at approximately the prevailing market rate (which is typically around LIBOR).

Remark The CAPM simplifies the situation in that it does not take into effect the credit worthiness of the fund when it wishes to borrow cash from the market. However, the assumptions are not unreasonable if we assume that the fund does not become very leveraged, since the fund could in principle at least arrange repo agreements which would allow it to borrow additional capital at close to the prevailing market rate.

Comparison of CAPM with Markowitz Model

The Capital Asset Pricing Model (CAPM) is an extension of the Markowitz Model. That is, within the construction of the CAPM we assume all the assumption of the Markowitz Model (see 3.1.1) but in addition the investor has the following two options when constructing a portfolio:

1. **Lend excess capital at the Market Rate:** The investor may lend money at the prevailing market rate. This constitutes the ability to hold within the portfolio a risk free asset which will provide the prevailing return available on cash. In practice, such assets are often referred to as a money market accounts and will yield a return in the region of LIBOR.
2. **Borrow capital at the Market Rate:** The investor may borrow money from the market at the prevailing market rate in order to invest within (risky) assets. This will increase the expected return of the original capital base and also increase its risk. In practice, the rate at which money can be lent from the market by a fund will be in the region of LIBOR (at least if the fund structure the loan as a REPO type agreement).

Remark Note that the rate of which money can be borrowed or lend from or to the market is the same.

The similarity of the assumptions of the two models are reflected in there development. In particular, the Efficient Frontier in the sense of Markowitz Theory also plays in central role within the CAPM. In the following section we will explain why the optimal portfolio which in the Markowitz Theory consisted of a locus (i.e. curve) have been transformed to lie of a line in the case of CAPM.

3.5.2 Nature of the Capital Asset Pricing Model (CAPM)

The introduction of the risk free asset, namely cash along with the risky assets transforms the ‘curve’ on which the optimal portfolios found with respect to the Markowitz Theory into a straight line, known as the Capital Market Line (CML) on which the optimal portfolios with respect to the CAPM lie. The reason for this is that in all cases in order to obtain the optimal portfolio with respect to the CAPM, that is the portfolio which offers

a given return for the minimal risk. You will need to construct a portfolio which consists of a ‘weighting’ of the ‘**Market Portfolio**’ (explained below) and either lend excess cash to the market or borrow cash from the market in order to purchase more (baskets) of the Market Portfolio.

The **Market Portfolio** is the portfolio on the Efficient Frontier in the sense of Markowitz Theory (see 3.4.2) which offers the greatest return per unit of risk. Expressing this analytically the **Market Portfolio** is the portfolio on the Efficient Frontier which maximizes:

$$\frac{\text{Expected Return of the Portfolio}}{\text{Total Risk of the Portfolio}}$$

The reason for this is that the investor whatever his risk/reward profile will seek the portfolio which offers the greatest return of a given level of risk, or the minimum risk for a given level of return. Clearly, at the level of the expected return of the Market Portfolio, the Market Portfolio itself is the optimal portfolio. However, if you wish to obtain a portfolio with a higher expected return then you should borrow cash (with zero risk) from the market in order to purchase more (baskets) of the ‘Market Portfolio’. Since the Market Portfolio is the cheapest way in terms of risk to increase the return of the portfolio. Similarly, if you wish to obtain a portfolio with a lower expected return than the Market Portfolio. You will need to hold a sufficient weighting of the Market Portfolio in order that a portfolio consisting of a weighting of the Market Portfolio with the remaining capital held as (zero risk) cash holdings will ensure the desired level of the expected return. Again the rationale for this construction is that the cheapest way to increase the expected return of a portfolio invested in cash is to move some of the cash into the Market Portfolio which by definition offers the cheapest means with regard to risk in obtained a higher expected return.

Therefore, when selecting the optimal portfolio with respect to the CAPM one of three possible scenarios will occur:

1. **Borrow money** at the prevailing market rate in order to purchase further blocks of the Market Portfolio. That is, the portfolio which offers the greatest expected return per unit of risk. Since the cash borrowings are risk free and unlimited you can in principle hold an arbitrarily large amount of the Market Portfolio using the same (original) capital base. Allowing the portfolio to be geared to a level where are arbitrarily high value of the expected return can be obtained.
2. **Lend Money** at the prevailing rate if you do not require the level of expected return which is offered by investing all the available capital in the Market Portfolio. That is, the portfolio with the highest expected return per unit of risk. The rationale being that even if you require a lower level of expected return a weighted portfolio between the cash with zero risk (and a positive return) and the most efficient means to purchasing additional expected return (i.e. the Market Portfolio) is going to be most efficient means of obtaining the desired level of the expected return.

3. **No cash holdings or borrowing:** if the expected return (or risk) you require is the same as the expected return (or risk) of the Market Portfolio.

The line within the risk/reward plain which replaces the Efficient Frontier of Markowitz Theory is known as the **Capital Market Line (CML)**. Where each point on the CML corresponds to an optimal portfolio (i.e. minimum risk for a given value of the expected return) with respect to the CAPM.

3.5.3 Applying the CAPM

Since the optimal portfolios with regard to the CAPM all lie on the CML within applications we will nearly always proceed along the following lines:

1. Construction of the Efficient Frontier
2. Evaluation of the Market Portfolio
3. Constructing the CML
4. Selecting a Portfolio from the CML

Construction of the Efficient Frontier

The implementation of the construction of the Efficient Frontier for the CAPM follows along exactly the same lines as the implementation provided for the Markowitz Theory and we refer the reader to the earlier section 3.4.2 for general advice concerning the setting of constraints and other issues effecting the construction of the Efficient Frontier. Though the techniques used within the construction does following along similar lines within the implementation of the CAPM we have only exposed the functionality necessary for the application of CAPM. In particular, you will find the followings two methods within the :Portfolio:CapitalMarket: EJB Component:

1. :Portfolio:CapitalMarket:setConstraints: - Sets of the constraints (lower and upper bounds) on the assets within the collection of (risky) assets from which the market portfolio and the other portfolios on the Efficient Frontier will be evaluated.
2. :Portfolio:CapitalMarket:calculateEfficientFrontier: - Evaluates a finite number of points on the Efficient Frontier of the collection of assets from which the market portfolio can be constructed and store them within a private field. It is necessary to evaluate the Efficient Frontier because by definition the Market Portfolio is the portfolio on the Efficient Frontier which offers the maximum expected return per unit of risk.

Remark In order to make a more detailed study of the Efficient Frontier we refer you to the methods contained within the Markowitz EJB Component.

By applying these two methods the Efficient Frontier is constructed as follows:

1. Evaluated the Efficient Frontier at a finite number of points. That is, find the portfolio which exhibits the lowest risk for a given expected return from the collection of asset available. Note that the weights of the assets may be subject to constraints.
2. Interpolate about these points using cubic spline (or some other method) in order to construct the Efficient Frontier. Since the Efficient Frontier by nature is ‘smooth’ using cubic spline interpolation to construct the Efficient Frontier will not introduce significant errors.

The points on the Efficient Frontier are portfolios constructed from the set of assets considered which exhibit the lowest risk for a given expected return. These portfolio are characterized by the following three characteristics:

1. Expected Return - The expected return of the portfolio which is estimated from the historical returns of the assets within the portfolio.
2. Total Risk - The total risk of the portfolio which is estimated from the historical returns of the assets within the portfolio.
3. Asset Weights - the weights of the collection of assets from which the portfolio can be constructed.

It is important to point out that the Efficient Frontier is monotonically increasing function in risk and expected return. This means that if we are given a value of the expected return then there will correspond a unique portfolio on the Efficient Frontier with a given total risk. Conversely, if we are given the total risk of the portfolio then there will exist a unique portfolio on the Efficient Frontier with a corresponding value its expected return.

Evaluation of the Market Portfolio

The Efficient Frontier is the collection of portfolios constructed from the given set of available assets. These portfolios have the lowest risk for a given value of the expected return with the possibility of constraints on the weights of the assets.

Once the Efficient Frontier has been constructed are next aim within the application of the CAPM is to find which portfolio on the Efficient Frontier which is the Market Portfolio. That is, the portfolio on the Efficient Frontier which maximizes:

$$\frac{\text{Expected Return of the Portfolio}}{\text{Total Risk of the Portfolio}}$$

The Market Portfolio is selected from the Efficient Frontier using the method :Portfolio:CapitalMarket:marketPortfolio;, which will return the (possibly constrained) weights of the Market Portfolio which has the highest expected return per unit of risk of the portfolios on the Efficient Frontier constructed from the available set of (risky) assets.

Uniqueness of the Market Portfolio

Here we discuss the ‘uniqueness’ of the market portfolio subject to some remarks concerning the given collection of assets from which the portfolios of the Efficient Frontier are constructed. These remarks though assisting in a deeper understanding of CAPM (and Markowitz Theory) can be safely skipped for those solely interested in the application of the CAPM (or Markowitz Theory).

The uniqueness of the Market Portfolio depends in the nature of the collection of assets from which the Efficient Frontier is constructed. Clearly the Efficient Frontier is convex and hence if we assume that there are two (or more) distinct Market Portfolios then it follows that one Market Portfolio is a leveraged version of the other in the following sense. The covariance between the two (distinct) market portfolios must be one since otherwise by holding a weighting of both market portfolios and gaining from the effects of diversification we are able to construct a portfolio with a higher expected return per unit of risk. Therefore, the covariance between any distinct Market Portfolios must be one.

In fact, if there are two distinct market portfolios with differing values of the risk and expected return then there exists an continuous range of Market Portfolios (i.e. an infinite number) in risk and expected return within the intervals of the risk and expected return of the two given Market Portfolios. The reason for this is that we can form a weighting of the two Market portfolios which forms another Market Portfolio (i.e. it has the same expected return per unit of risk) which has any value of the expected return or risk within there respective intervals. This possibility becomes clear if we consider the portfolio $R = aP + (a - 1)Q$, where a lies in the interval $[0, 1]$, and where P, Q are the two given Market Portfolios. Now as a varies from 0 to 1 the portfolio R ’s expected return and risk will vary over all values within the respective domains but the expected return per unit of risk will remain fixed. That is, they will all be Market Portfolios.

Constructing the CML

The Capital Market Line (CML) is simple the line within the risk - expected return plain which passes through the Market Portfolio and cash. Below we derive formulae for the construction, expected return and risk of an arbitrary portfolio on the CML. We also refer the reader to the diagram of the CML at 3.5.4 which is probably to quickest and easiest means to obtain a better understanding on the CML and its construction.

Expressing this analytically: If we construct a portfolio from a (positive or negative) weighting of cash (C) where the prevailing market rate on lending and borrowing cash is r , and the Market Portfolio (M) for a given collection of (possibly constrained) assets denoted by M with an expected return of $E(M)$ and risk of $R(M)$, then all portfolios (P) on the CML will take the following form:

$$P = x_{cash}C + x_{market}M$$

where $x_{cash} + x_{market} = 1$; with the real numbers x_{cash} , x_{market} denoting the weighting of cash and the Market Portfolio within the constructed portfolio on the CML P .

The **expected return** $E(P)$, of the constructed portfolio P is given by:

$$E(P) = x_{cash}R + x_{market}E(M)$$

where as mentioned above R is the return on cash and $E(M)$ is the expected return of the Market Portfolio, and where again $x_{cash} + x_{market} = 1$; with the real numbers x_{cash} , x_{market} denoting the weighting of cash and the Market Portfolio within the constructed portfolio on the CML P .

Similarly, the **risk** σ_P , of the portfolio P is given by:

$$\sigma_P = (1 - x_{cash})\sigma_m = x_{market}\sigma_m$$

where σ_m is the risk of the Market Portfolio, and where again $x_{cash} + x_{market} = 1$; with the real numbers x_{cash} , x_{market} denoting the weighting of cash and the Market Portfolio within the constructed portfolio on the CML P .

Remark With the :Portfolio:CapitalMarket: EJB Component we provide methods by which the expected return and risk of the Market Portfolio can be constructed. Moreover, we also offer within this class means by which the weighting of the Market Portfolio of an arbitrary portfolio on the CML can be found. In fact, it turns out (see 3.5.3) that if we know any one of the expected return, risk or Market Portfolio weighting of a portfolio on the CML then we are able to deduce the other two properties.

Selecting a Portfolio from the CML

The Capital Market Line (CML) is a collection of portfolios which can be constructed from the available (risky) assets with the option of borrowing or lending (risk free) cash at the prevailing market rate. Since by default the amount of cash which can be borrowed or lend is not restricted the expected return of the resulting portfolios can be anything equal or greater than the return available from cash. Similarly, the risk from possible portfolios can (by default) be any positive number.

We allow within this Component the portfolios on the CML to be selected from knowledge any one of the following:

1. Expected Return
2. Total Risk
3. Weighting of the Market Portfolio

This in fact allows for the greatest generality and moreover which even property is used in the identification the other two properties can be deduced as described below.

Completeness of the Methods: riskCML, weightCML, returnCML, weight2Risk of the CapitalMarket EJB Component

The portfolio on the CML can be selected when one of the following three properties is known:

1. Total Risk of the optimal portfolio on the CML.
2. Expected Return of the optimal portfolio on the CML.
3. Weighting of the market portfolio on the CML.

Then using the above mentioned methods we are able to evaluate the other (two) quantities from the three properties which are listed above. For example, if the expected return of the portfolio is known then the weight of the market portfolio can be evaluated using weightCML and the risk can be evaluated using riskCML. If on the other hand the total risk of the portfolio is known then the corresponding expected return of the portfolio can be evaluated by returnCML, and then using this deduced value we are able to evaluate the weight of the market portfolio using weightCML. For completeness we include the method weight2Risk which evaluates the risk of a portfolio on the CML when the weight of the market portfolio within the CML portfolio is known. From knowledge of the risk of the CML portfolio we are able to evaluate the corresponding value of the expected return using the method returnCML.

Therefore, using the three ‘...CML’ methods along with ‘weight2Risk’, whichever one of: total risk, expected return or weight of market portfolio, is used in order to select the optimal portfolio from the CML we are able to deduce the other two quantitative properties.

3.5.4 Summary of the CAPM

We summarize the CAPM with the following diagram which contains the Efficient Frontier (the black curve) which is first evaluated, then the Market Portfolio (denoted by a red square) is found on the Efficient Frontier. After which when ‘cash’ (denoted by red square on Expected return axis) we are able to construct the CML (red line) by drawing a line through cash and the Market Portfolio.

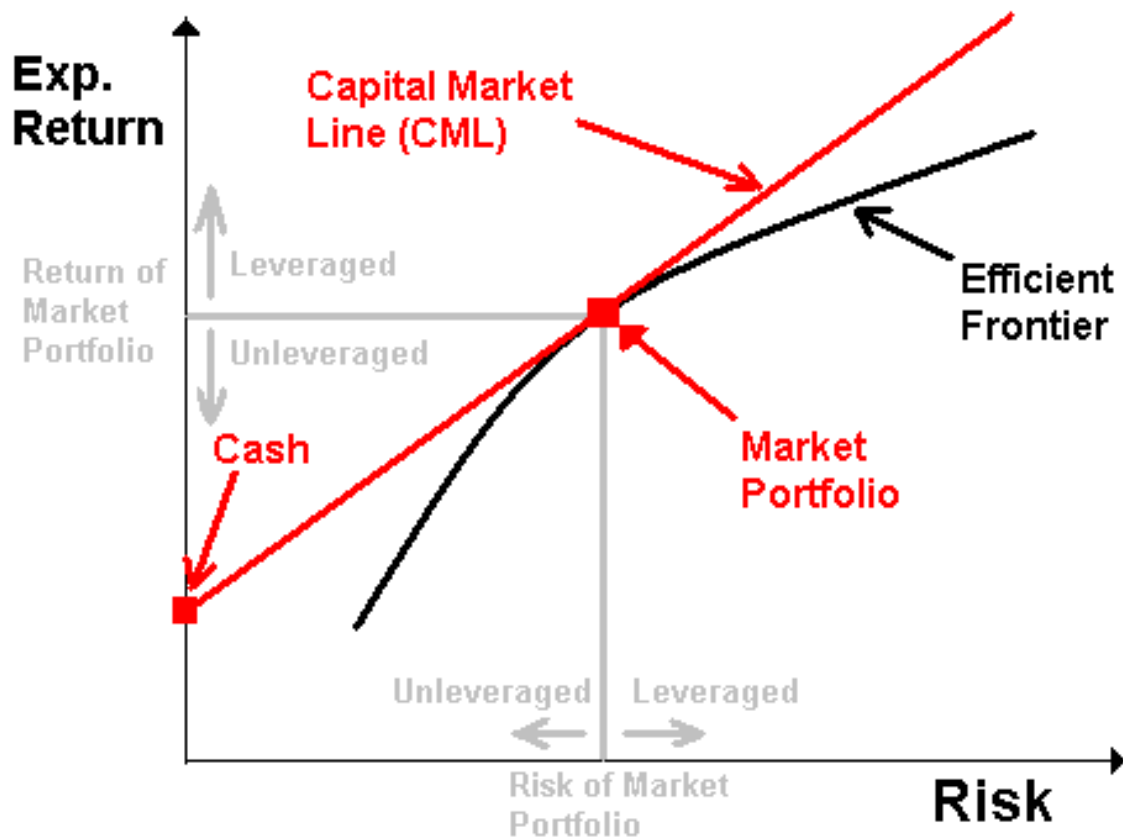


Fig: The CML, Market Portfolio, and the Efficient Frontier

The gray perpendicular lines on the diagram running off from the Market Portfolio indicates the values of the risk and expected return of the Market Portfolio. Any portfolios to the left on the risk axis to the Market Portfolio on the CML are portfolio which borrow cash from the market. Similarly, any portfolio to the right on the risk axis on the CML are portfolio which lend cash to the market. In a similar fashion, any portfolios on the CML with an expected return greater than the Market Portfolio (i.e. above the gray horizontal line) borrow cash from the market, and any with a lower expected return (i.e. below the gray) lend cash to the market.

3.5.5 Putting constraints on the level of borrowing and lending

At present we are able to place constraints on the weights of the (risky) assets from which the portfolios on the Efficient Frontier used within the Markowitz Theory and CAPM can be evaluated. This allows the risky asset held within the portfolios on the Efficient Frontier from which in the case of Markowitz Theory the optimal portfolio will be selected. Hence, within the framework on Markowitz Theory we have a means by which to place constraints on all the assets of the optimal portfolio.

In the case of the CAPM, a portfolio can also contain either cash holdings (i.e. free risk assets yielding the prevailing market rate on cash) or borrowings (i.e. money borrowed from the market at the prevailing market rate). Are aim here is to rationalize and detail

the issues related to why and how constraints can also be placed on the cash holdings or borrowings.

Rational of Borrowing Limit

Within the mandate of a fund, the manager will often be set a maximum amount of gearing which the fund can take. If the manager ever wishes to go above this level then they will need to obtain authorization from the funds board.

From the point of view of risk management and hence credit rating and regulation. If there is maximum level of gearing then there is a worst case scenario otherwise the worst case scenario since the gearing is unlimited is unlimited losses.

Rational for a Cash Limit

The existence of a cash limit forces the fund to invest within the market which it addresses. The primary motivation for introduction such a limit are as follows:

1. **Role of a Fund Manager:** A portfolio manager to paid to invest money by selecting assets within a given sector or market. After all if the money is just deposited within a bank account the fund manager will still collect the management fee without reflecting his mandate. The mandate explicitly said invest in... If the investor wants to be out of the market then they can take money out of the fund.
2. **Reflect fund description/mandate:** An equity fund should not be a money market fund. I.e. Having limit like no more the 40% in cash at any time is reasonable. If this is not possible then capital should be returned to investors for similar reasons as within any other business.⁵

Mechanics of controlling the Cash/Borrowing Level

Within the CAPM the optimal portfolios which are constructed from a combination of the Market Portfolios with the possibility of cash holdings or borrowings lie on the CML. If there is a pre-defined limit on the level of cash holdings or borrowings then these limits will translate into a limit on the weighting the Market Portfolio within the portfolio of the CML. From limits on the weighting we will be able to imply the limits on the risk, or expected returns of the portfolio on the CML which satisfies the cash and borrowing limits since as shown above if one of the following three: weighting, risk, expected return; are known then the other two can be deduced.

⁵For further discussion on the effect of retaining earners and compounding we refer the invested reader to; The Essays of Warren Buffet, Lessons for Investors and Managers, edited by L.Cunningham.

Implemented as an Example

We provide within this package a client example which illustrates how constraints on the level of cash lending or borrowing can be used. The client is entitled:

`CapitalMarketClient_CashConstraints`

This example illustrate how constraints may be placed onto the levels of the cash held or borrowed to the market by an optimal portfolio constructed in accordance within the Capital Asset Pricing Model. We show how the resulting continuous range of the values of the expected return and total risk can then be deduced which identify which optimal portfolios can be selected when the cash level constraints are observed.

In particular we consider the following problem:

We know that the market portfolio of a collection of assets has an expected return of 10 percent per year and a risk of 20 percent per year, we also know that cash can be borrow or lend from the market at a prevailing rate of 5 percent a year. According to the CAPM if the portfolio manager can only leverage his portfolio by 20 percent, and must also never hold more than 30 percent of the funds capital in cash then what is the range of total risk and expected return of the (optimal) portfolios which the fund manager can hold?

3.6 Performance Evaluation

As we discussed earlier since there is a relation between the risk and the expected return from an asset (CAPM) an assessment of a portfolios performance based only on the absolute return is misleading and unfair. Therefore, we describe standard methods which are used to given a risk adjusted performance measure.

Within the US mutual fund market there are more than 12,000 funds which apply a variety of investment styles and have a different mix of investment assets. Broadly speaking US mutual funds can be split into the following categories growth funds, balanced funds, income funds, government bonds funds, junk bond funds and international funds. According the CAPM the investors would expect a greater return from a growth fund which invests in stocks than a government bond fund because the growth fund will exhibit more volatility. The amount of “out performance” of the growth fund over the bond fund which the investor would expect is the topic of this section.

We offer various methodologies which are used for the evaluation of portfolio performance in the broad sense. These measures are widely used and offer a fast and efficient way in order to make informed comparisons between portfolio performance taking into account risk, return, asset class mix, yield and market conditions.

Within our components we implement the following methods:

- **totalReturn** - calculates the total return of the portfolio over a given period taking into account any disbursements and dividends payments
- **geometricMeanReturn** - the geometric mean of the return over a number of periods. Say the return is quoted for a 2 year period then by taking the number of periods as 2 the method will return the equivalent return over a 1 year period
- **sharpesRatio** - we implement Sharpe's ratio which takes into account both the return and risk when accessing a portfolios performance
- **treynorsMeasure** - Treynor's performance measure which takes into account the systematic risk (or beta) and the average return when assessing the overall performance

3.6.1 Comparing the Sharpe and Treynor Performance Measures

The Sharpe portfolio performance measure is based on the capital market line (CML) and total risk, which makes it more suitable for evaluating portfolios rather than individual assets. On the other hand, Treynor performance measure is based on the capital asset pricing model (CAPM) and are more flexible because by using systematic risk (beta) it can be used to evaluate the performance of both portfolios and individual assets. Both performance measures tend to rank a group of diversified portfolios similarly.

3.7 Further and Supplementary Reading

3.7.1 Supplementary Reading

- **H. Markowitz** Portfolio Selection, Journal of Finance, 1952, p77-91

This fundamental paper establishes the basis of the Markowitz theory and shows that portfolios dominate individual assets from a risk and return standpoint. Further it shows how to construct optimal portfolios with ideal risk - return characteristics.

3.7.2 Further Reading

- **H. Markowitz** Portfolio Selection, Efficient Diversification of Investments, Basil Blackwell

Expands on the classic 1952 article listed above detailing the modern treatment and development of the theory.

- **W. Sharp** A Simplified Model for Portfolio Analysis, Management Science, Vol. 9, p27-93

Chapter 4

Programmer's guide

4.1 The Portfolio components methods

This component offers portfolio analysis according to Markowitz and Capital Market Theory. There are two main categories of methods which are provided within this component:

- statistical parameter calculation
- portfolio analysis

The values returned from the first category are used when calling portfolio analysis methods. You can calculate either individual asset parameters (e.g. the expected return from an asset), or portfolio parameters (e.g. the expected return of the portfolio). Usually you will be interested more in the analysis functions. Almost all of these functions require the covariance matrix and / or the expected returns vector. You can obtain them by calling *covMatrix*, respectively *expArray*. Also, when calculating statistical parameters, you can give the probabilities as parameters or you can use an estimation based on historic rates of return (see 3.1.1 for details).

The most time consuming part of the analysis is the determination of the efficient frontier (see 3.4). This requires a series of minimizations, which are done using Rosen's gradient projection algorithm. Because it is inefficient to do this every time you want to know the portfolio on the efficient frontier corresponding to a given expected return, the result is determined by interpolating through a series of points. The interpolation points must be calculated before a call to *efficientFrontier(double, int)* is made. The method *calculateEfficientFrontier* does this. If you are interested instead in only one point from the efficient frontier, you can use *efficientFrontier(double, double[[[[]]], double[], int, double)*. In order to determine the optimum portfolio (according to Markowitz model) a utility function (indifference curve) must be provided (see 3.5). It can be given either as a set of interpolation points, or as a polynomial (by specifying coefficients). Having a utility function and an efficient frontier, you can determine the set of optimal portfolios (i.e. the set of intersection points between the two curves) using *optimalPortfolio*. If you are interested only in one value you can use *optimalPortfolioMaxExpected* which determines the point in the set with

the maximum expected return. The maximum number of optimal portfolios found is 100. If this number is exceeded, an exception will be thrown.

We provide a method for calculating the equity portfolio (see 3.6). This also requires the efficient frontier to be calculated.

Remarks You should be aware that there are two sources of errors: the first is the optimization procedure, the second is the interpolation. Usually the errors generated by the interpolation are much bigger than that resulting from the optimization. You can reduce the interpolation errors by increasing the number of interpolation points. It is much harder to control the optimization errors. If you are especially interested in an accurate result, you can adjust the 'portal' parameter to lower values, otherwise this should always be set between $1E - 3$ and $1E - 6$. The 'projtol' parameter is not the tolerance of the result, so there are no guarantees that the error will be less than a specific number. Also the increase in duration of the algorithm does not depend linearly on the value of 'projtol', so a small decrease of it can cause a great increase in time consumption. Generally, the accuracy of the result depends on the input data. Usually, the more assets are in the portfolio, the bigger optimization errors are.

4.2 Exceptions

If one of the methods which require the efficient frontier to be calculated is called before *calculateEfficientFrontier*, the exception ***EfficientFrontierNotCalculatedException*** will be thrown.

If one of the methods which require the utility function is called without initializing it, the exception ***UtilityFunctionNotInitializedException*** will be thrown.

If the number of optimum portfolios found is greater than 100, the exception ***TooManyPortfoliosException*** is thrown.

The expected return of a portfolio cannot be greater than the maximum expected return of the assets. If you want an expected return higher than this, the problem has no solutions. So, when calling *efficientFrontier* with such an expected return, you will get a ***NoSolutionException***. Also if the range given to *calculateEfficientFrontier* contains any points for which there is no solution, the exception will be raised.

4.3 Interfacing with databases

The component has methods which provide database connectivity. In order to use these methods you must have access to a DBMS and also you must have a JDBC driver installed. All database-aware methods have at least four parameters: a string containing the fully qualified name of the JDBC driver, a string containing the URL of your database, a string

with the name of the table, and a Properties parameter containing a list of arbitrary string tag/value pairs as connection arguments. Usually here you should include at least a “user” and “password” property. There are three methods of this kind:

- `covarianceMatrix(String, String, String, java.util.Properties, int N)` - calculates the covariance matrix using the historical rates of return given in a database table.
- `expectedArray(String, String, String, java.util.Properties, int N)` - calculates the expected returns vector using the historical rates of return given in a database table.
- `setUtilityFunctionInterp(String, String, String, java.util.Properties)` - initializes the utility function, the interpolation points being get from a database table.

Example: Suppose you have a portfolio composed of 3 assets. You will need a table named **hist_returns** on your database. The table will look like this:

Time interval	Asset1	Asset2	Asset3
1 st interval	100	50	27.5
2 nd interval	22	55	30
3 rd interval	40	57.2	32

Also, suppose you have installed MySQL on `server.yourdomain.com`, with the classpath of the JDBC being `org.gjt.mm.mysql.Driver`. The database name is **PortfolioDB** and you don't need a username or a password to connect.

The code needed to determine the covariance matrix is:

```
double[] [] cov = covarianceMatrix("org.gjt.mm.mysql.Driver",
    "jdbc:mysql://server.yourdomain.com/PortfolioDB", "hist_returns",
    null)
```

4.4 Writing a typical client

Here is a typical application of the Portfolio bean most other applications will take a similar form. The following steps must be followed in any EJB client program:

- Locate the home interface using JNDI.
- Create an instance of the bean.
- Invoke the bean's methods.

Below is the source code which would implement should a client:

```
//TypicalClient.java
import java.rmi.RemoteException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.sql.*;
public class TypicalClient {
    static Context initialContext = null;
    static com.webcab.ejb.Portfolio.PortfolioHome home = null;
    public static void main(String[] arg) {
```

In order to retrieve the home interface you must create an initial context.

```
        // Get InitialContext
        try {
            initialContext = new InitialContext();
        } catch (NamingException e) {
            e.printStackTrace();
            System.exit(2);
        }

        // Lookup bean home
        String beanName = "PortfolioHome";
        try {
            home = (com.webcab.ejb.Portfolio.PortfolioHome)
initialContext.lookup(beanName);
```

Now you can create the bean object, using the create method from the home interface.

```
com.webcab.ejb.Portfolio.Portfolio bean = home.create();
```

From now on you can use the newly created bean. Suppose you have the historic rates of return given in a database table and you wish to find the optimal equity portfolio. The steps are:

- Get the covariance matrix using `bean.covarianceMatrix(...)`
- Get the expected array using `bean.expectedArray(...)`
- Calculate a number of points (in our case 10) from the efficient frontier using `bean.calculateEfficientFrontier(...)`

- Calculate the array of weights corresponding to the equity portfolio using `bean.equityPortfolio(...)`

```
int N = 3, observations = 3;
double Emin = 30, Emax = 60;
double[][] hist_returns, cov = null;
double[] r = null;
java.util.Properties info = null;
try {
    cov = bean.covarianceMatrix("org.gjt.mm.mysql.Driver",
    "jdbc:mysql://server.yourdomain.com/PortfolioDB",
    "hist_returns", info, N);
    r = bean.expectedArray("org.gjt.mm.mysql.Driver",
    "jdbc:mysql://server.yourdomain.com/PortfolioDB",
    "hist_returns", info, N);
} catch (SQLException e) {
    System.out.println("SQLException: " +
e.getMessage());
    System.out.println("SQLState: " + e.getSQLState());
    System.out.println("Vendor error: " +
e.getErrorCode());
    System.exit(-1);
}
bean.calculateEfficientFrontier(Emin, Emax, cov, r, N,
10, 1E-3);
System.out.println("\n\nEquity portfolio: ");
double[] x = bean.equityPortfolio(Emin, Emax, cov, N);
```

You can use now the statistical methods to calculate the parameters of the obtained portfolio.

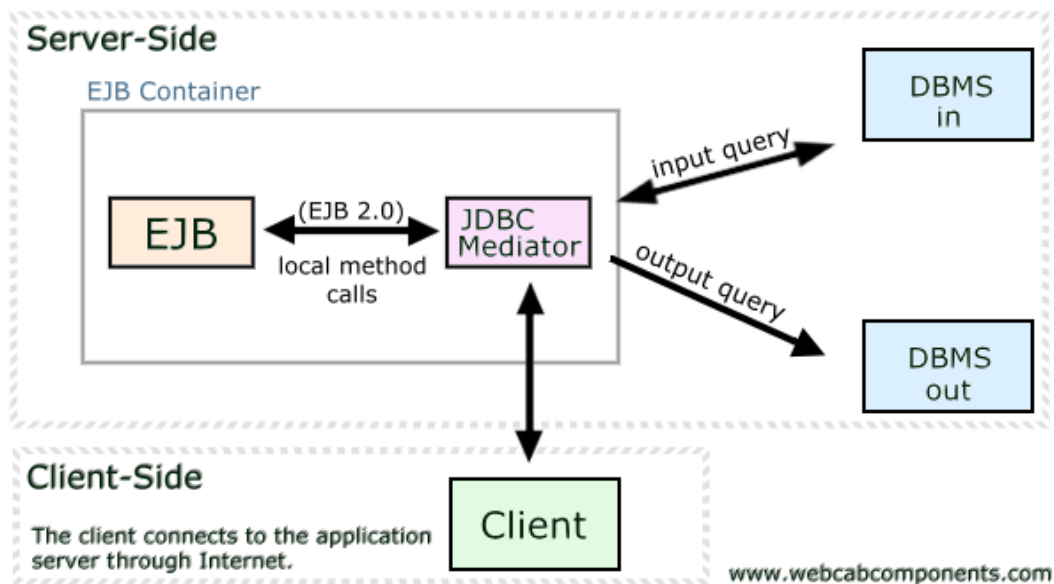
```
        double equity_expected = bean.portfolioReturn(x, r, N);
        double equity_risk = bean.portfolioRisk(x, cov, N);
        for (int j = 0; j <= N - 1; j++)
            System.out.println("\tx[" + j + "] = " + x[j]);
        System.out.println("\trisk = " + equity_risk +
"\texpected return = " +
            equity_expected + "\n");
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(-1);
    }
}
```

You can find the code presented above in the following folder: */Jars/Examples/Typical*

4.5 Using JDBC Mediator with our EJBs

4.5.1 Overview

The JDBC mediator is an EJB component which mediates between an EJB component, its clients and DBMS. For each EJB component there exists a corresponding mediator with its own exception class. The mediator receives an input and an output SQL query for the DBMS and a method name to analysis. The mediator sends the SQL input query to the (input) DBMS and then applies the method from the EJB component to each row of the input query results. Each result row is written into the (output) DBMS according to the output SQL query.



A diagram that illustrates how JDBC Mediator intermediates the communication between the client, the Application Server and the Database Servers.

4.5.2 Connecting to your Database Server

If you wish to connect by JDBC to your Database server in order to run a select query an update statement or a stored procedure you will need to describe the connection with certain properties. You will be given the opportunity to use JDBC Mediator with Application Server DataSources and Connection Pools or directly specifying these properties and passing them to the `create` methods.

- **Driver**

The driver is a Java class provided by your Database vendor that implements the `java.sql.Driver` interface.

- **URL**

This property defines the address of your DBMS, the port number and additional information such as service and database name.

- **Username and Password**

For authentication reasons, most connections will not work without providing a user name and a password.

- **Additional Properties**

This field allows you to customize even more your JDBC connection as described in the JavaDocs

We describe these properties individually for the most well known database systems. Please feel to jump to the section which refers to the particular DBMS you plan to use with our JDBC components.

Oracle 9i

We recommend the JDBC thin driver `oracle.jdbc.driver.OracleDriver`. You may download this driver from http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html. The Oracle URL has the following format:

```
jdbc:oracle:thin:@servername:port:service
```

where *servername* is the Internet name of your Oracle DBMS, *port* is usually 1521 and *service* is the name of the service you are going to connect to.

IBM DB2

The name of the DB2 driver is `COM.ibm.db2.jdbc.net.DB2Driver` and can be downloaded off the IBM site at <http://www-106.ibm.com/developerworks/db2/zones/java/>. The format of the DB2 URL is:

```
jdbc:db2://[servername]:[port]/[database]
```

where *servername* is the Internet name of your DB2 DBMS, *port* may be ignored and *database* is the name of the database you plan to connect to.

Microsoft SQL Server

The name of the SQL Server driver is `com.microsoft.jdbc.sqlserver.SQLServerDriver` and can be downloaded off the Microsoft site at <http://msdn.microsoft.com/...>. The format of the Microsoft SQL Server URL is:

```
jdbc:microsoft:sqlserver://servername:1433
```

where *servername* is the Internet name of your SQL Server.

Sybase

The name of the Sybase driver is `com.sybase.jdbc.SybDriver` and can be downloaded off the Sybase site at <http://www.sybase.com/home>. The format of the Sybase JDBC URL is:

```
jdbc:sybase:Tds:servername:port/
```

where *servername* is the Internet name of your Sybase DBMS and *port* is the port number where the Sybase server is running.

4.5.3 JDBC Components

Most methods implemented by our enterprise Beans accept numeric parameters and return numbers or arrays of numbers. By using these JDBC methods directly inside a database

server without any change in functionality will definitely prove necessary. In this case the database server becomes the `DataSource` for the input and/or output parameters for every one of these methods.

What we are going to discuss applies only to the following modules included in this application. Modules not listed here either do not implement JDBC or use a particular JDBC implementation.

- The “**Portfolio**” J2EE Module

For every bean implementing number-based methods (methods with numeric parameters and numeric return values) there is a bean in a sub-package called “`jdbc`” bearing a similar name. For example, if there's a bean called “`FinancialBean`” within a package called “`com.webcab.ejb.finance`”, the corresponding JDBC-implementing component would be “`com.webcab.ejb.finance.jdbc.FinancialBeanJDBC`”.

The JDBC components are designed to easily transfer to an Application Server JDBC specific tasks that make use of the capabilities of every bean within our enterprise application. The enterprise database performs demanding transactions which take place on the same machine that hosts your EJB components. This allows the client to take advantage of the processing power of your Application Server and minimize bandwidth transfer.

The JDBC beans are created by specifying all necessary JDBC connection parameters, such as drivers, url, username and password plus the enterprise Bean's creation parameters. Every JDBC component contains several methods that invoke methods from their corresponding enterprise Bean and apply them directly to your database tables and/or write the answer back into the same or another DBMS. The JDBC methods accept the name of the enterprise method as the first parameter and an SQL Query or an array of Java objects for the method's input and output.

The following source code listing exemplifies how to use an input/output SQL query JDBC method for a generic bean called “`com.webcab.ejb.finance.FinancialBean`” and a generic method “`double calculateAmount (double, double)`”.

```
import com.webcab.ejb.finance.FinancialBean;
import com.webcab.ejb.finance.jdbc.FinancialBeanJDBC;
...
try {
    ...
    /*
     * The first creation parameter (riskFreeInterestRate) is FinancialBean
     * specific. The next creation parameters define the input and the output
     * DBMS connections by driver, url, username, password and additional
     * properties.
     */
    FinancialBeanJDBC jdbcBean = home.create (riskFreeInterestRate,
        "oracle.jdbc.driver.OracleDriver", // Input Connection
        "jdbc:oracle:thin:@inputserver.com:1521:ORACLE",
        "SCOTT", "tiger", null,
        "oracle.jdbc.driver.OracleDriver", // Output Connection
        "jdbc:oracle:thin:@updateserver.com:1521:ORACLE",
        "MIKE", "wolf", null);

    jdbcBean.call ("calculateAmount",
        "SELECT C_ID, SHARES, VALUE FROM TRADES",
        "UPDATE CUSTOMER SET MONEY=? WHERE C_ID=?",
        new int[] [] {{2, 1}});
    catch (Exception e) {
        e.printStackTrace ();
    }
}
```

The first parameter of the `call` method represents the `FinancialBean` method name. The second parameter is the input query, a select query which returns three columns `C_ID`, `SHARES` and `VALUE`. The third parameter is a prepared statement that is meant to write the value returned by `calculateAmount` in the `MONEY` field and the `C_ID` primary key in the where clause. This assignment is described by the last parameter, an array of integer pairs that specifies the index of the IN parameter in the output query and the column number of the input query. Both indices start counting from 1. In our case, the second IN parameter is the second question mark in the update statement and the first input column is `C_ID`.

Every output update is made according to a primary key `C_ID`. Even though this is returned by the select statement it is not used for computation by the two-parameter “`calculateAmount`” method.

As an alternative you may wish to define two `DataSources` that point to the input and output connections described above. You may then map the corresponding resource references

defined inside the `ejb-jar.xml` deployment descriptor to the JNDI names corresponding to these `DataSources`. In this case scenario, the previous code would look like this:

```
import com.webcab.ejb.finance.FinancialBean;
import com.webcab.ejb.finance.jdbc.FinancialBeanJDBC;
...
try {
    ...
    /*
     * The only creation parameter is riskFreeInterestRate. Ignore the
     * JDBC specific parameters. The bean will be using the
     * resource references defined inside the ejb-jar.xml XML
     * to locate the Input and Output JDBC connections.
     */
    FinancialBeanJDBC jdbcBean = home.create (riskFreeInterestRate);

    jdbcBean.call ("calculateAmount",
        "SELECT C_ID, SHARES, VALUE FROM TRADES",
        "UPDATE CUSTOMER SET MONEY=? WHERE C_ID=?",
        new int[] [] {{2, 1}});
    catch (Exception e) {
        e.printStackTrace ();
    }
}
```

Note The primary key reference can be made across two different database servers a very useful feature provided by this type of JDBC methods.

4.6 Support for Developers

4.6.1 Compilation and Custom Modification of Source Code

This J2EE Application and related Java™ source files have been compiled using Sun's J2SDK1.4.2 and J2EE 1.3, should you prefer compilation of the source code using another compiler then please contact us. Some functions (for example standard mathematical operations) are embedded within the source code, if you license classes which you believe offer a similar level of functionality from a third party and wish to include these classes within our components then please contact us. Both of the above services are offered free of charge at WebCab's discretion.

4.6.2 Online Support

If you have any questions or queries concerning the use of this component then please feel free to contact us via our support forum at:

<http://www.webcabcomponents.com/support/index.php>

For updates and new releases, visit:

<http://www.WebCabComponents.com>

Chapter 5

Examples

Within this chapter we illustrate how the Portfolio Component can be applied to solve real life problems. We provide with this Component examples of two types:

1. **QA Examples**
2. **Custom Examples**

5.1 Question and Answer (QA) Client Examples

5.1.1 Overview

Within this folder we offer Java client examples for the J2EE Business Components provided within this package. In particular, for each business method contained within each business Component we provide a client side example; which illustrates how functionality contained within this component can be applied to solve real world problems. In addition, to these examples we also include custom applications which solve some of the generic problems which this Component is designed to address.

5.1.2 Structure of QA Examples Directory

By drilling down the QA Client directory structurr you will find the following six directory levels:

1. Client Level
2. Technology Level
3. Business Module Level
4. Business Class Level
5. Example Level
6. Custom Example Level

which have the structure as shown in the following diagrams.

5.1.3 Top Four levels of the QA Directory Structure

```

      (Technology Level) (Business Module Level)          (Business Class Level)

Client +
      |
      + Java -----+
      |              |
      + Readme.txt   + 'Business Module 1' -----+
                      |                             |
                      + 'Business Module 2' --+... + 'Business Class 1' -----
                      |                             |
                      + QALibrary.jar           + 'Business Class 2' --+...
                                                  |
                                                  + ...

```

5.1.4 Bottom Two levels of the QA Directory Structure

```

      (Example Level)          (Custom Example Level)

-----+
      |
      + 'BusinessClass'.java
      |
      + compile.cmd (.sh for Linux)
      |
      + run.cmd (.sh for Linux)
      |
      + run_gui.cmd (.sh for Linux)
      |
      + 'CustomClient' -----+ 'CustomClient'.java
      |                       |
      + ...                   + compile.cmd (.sh for Linux)
                              |
                              + run.cmd (.sh for Linux)

```

5.1.5 Quick Start Guide

Browse down to the particular client within the business module for the given client technology you are interested in. Run the compile.cmd (.sh for Linux), script in order to compile the example and then run the run.cmd (.sh for Linux), script in order to run the example.

5.1.6 Explanation of the QA Directory Structure and its files

1. Client Level

The directory containing the Questions and Answer Examples is named 'QAClients'. This directory can be located by using the Index.html file, which is presented at the end of the installation process, by selecting:

Run Examples > Client Examples Directory

Within this folder you will also find the following file:

- Readme.txt - this readme file

2. Client Technology Level

Within the 'Client' folder is the Technology Level of the QA directory structure. Within this folder you will find sub-directories; which contain examples written in each of the Java Client Technologies in which the clients have been implemented.

As mentioned above within the overview we provide an implementation of each 'Question and Answer (QA)' example using each of the client technologies used. Selecting one of these folders in order to view the client implemented within the corresponding technology.

3. Business Module Level

After selecting one of the technology sub-folders at the 'Client Technology Level' (see (2) above) you will enter a folder which contains a sub-folder for each of the modules contained within this component package. The modules are convenient collections of business classes containing the business functionality offered by this component. Each module has a sub-folder which contains the 'Question and Answer' examples of each method of the classes which it contains.

This directory will also contain the following file:

- QALibrary.jar - contains utility functionality necessary for running the clients.

4. Business Class Level

At the business class level under the module level we are presented with one sub-folder for each of the classes within the business module. Each of these folders contains the files of the examples for each of the methods contained with the respective class.

5. Example Level

At the Example level you will be presented with the files which allow you to compile

and then run the examples of the application of the Business methods of the respective class. The files within this directory which constitute the Client example are as follows:

- Source Code File(s) - Depending on the Java compatible technology selected at the 'Technology Level' this folder will contain a source code file(s) of the client.
- Compile Batch File - Assuming that you have access to the relevant compiler the compile batch file (compile.cmd or compile.sh) once run will compile the client example from the folders source code file.
- Run Batch File - The run batch file (run.cmd or run.sh) will run the examples executable file (exe) which is generated from the compilation of the source code file. If the example over runs your console screenful then press space in order to page down or enter is order to scroll down.

6. Custom Example Level

At the Custom Example level you will find a source code file containing the implementation of the Application which addresses a typical question for which the Component is designed. The source code has a linear structure with full inline commentary. By just running the compile and then run scripts contained within this directory you will be able to solve to given problems view the results obtained. The files contained within each Custom Client Example directory are as follows:

- Source Code File(s) - The custom clients source code in the appropriate client technology.
- Compile Batch File - Assuming that you have access to the relevant compiler the compile batch file (compile.cmd or compile.sh) once run will compile the client example from the folders source code file.
- Run Batch File - The run batch file (run.cmd or run.sh) will run the examples executable file which is generated from the compilation of the source code file. If the example over runs your console screenful then press space in order to page down or enter in order to scroll down..

5.2 Custom QA Examples

Portfolio theory addresses the very practical aim of trying to get the highest expected return from a portfolio for a given level of risk. Within this section we detail the custom examples which we provide with the portfolio components standard QA Example, which illustrate the application of the Portfolio Component.

5.2.1 Markowitz Custom Clients

Within this section we describe the custom examples which we provide for the Morkowitz EJB Component. The source code for these examples can be found within sub-directories

of the directory:

Clients > QAClients > "Language/Technology" > Portfolio > Markowitz

where "*Language/Technology*" refers the program language and/or technology (for example Java, C, Delphi, EJB, COM, Web services, CORBA etc) by which the example are written in.

1) AbsoluteRelative

This example demonstrates that whether the absolute or relative values are used for the source data of the historical returns the Portfolio Component will yield the "same" results in terms of selecting the optimal portfolio.

In particular, we will show the independence of the Weights of the Portfolio found on the Efficient Frontier for a given expected return. Moreover, we will show that whether we use absolute or relative values for the historical returns the weights of the optimal portfolio found are identical.

One reason for this independence is that the units in which the historical returns are given does not effect the weights of the portfolio of the Efficient Frontier, include the weights of the optimal portfolio is that asset weights themselves are nit-less' (in the sense of Measurement Theory). Hence intuitively speaking during the computation of the asset weights the units are being canceled out with the end result (i.e. the weights) being recording without an attached unit.

2) Efficient Subdirectory

This example demonstrates how we are able to evaluate a single Portfolio on the Efficient Frontier. That is, we demonstrate how we are able to find any portfolio of the Efficient Frontier with only one call to the underlying Rosen's optimization algorithm.

In instances when we only require to evaluate a few portfolios on the Efficient Frontier then this direct approach is not only more efficient but also more accurate.

Level of the Precision

Another aspect of this example is that we show how the portfolio found depends on the precision used. In particular, we find that a precision of 1E-12 (or higher) should be used in order to assure that the weights of the portfolio are corrects to 2 decimal places.

Historical Values Used

Within this example we use the following relative historical values of five assets historical returns over the last five periods:

- 1st Asset: 0.03, 0.04, 0.02, 0.05, 0.03
- 2nd Asset: 0.03, 0.040, 0.1, 0.03, 0.035
- 3rd Asset: 0.03, 0.032, 0.033, 0.035, 0.036
- 4th Asset: 0.02, 0.021, 0.021, 0.021, 0.0205
- 5th Asset: 0.02, 0.01, 0.02, 0.015, 0.028

Where the portfolios on the Efficient Frontier can be constructed with respect to the following asset weight constraints:

- 1st Asset Weight Constraints: Lower bound of = 0.05, Upper bound = 0.2
- 2nd Asset Weight Constraints: Lower bound = 0.05, Upper bound = 1
- 3rd Asset Weight Constraints: Lower bound = 0.05, Upper bound = 1
- 4th Asset Weight Constraints: Lower bound = 0.05, Upper bound = 0.5
- 5th Asset Weight Constraints: Lower bound = 0.05, Upper bound = 1

3) Optimal Subdirectory

With this example we select the Optimal Portfolio in accordance with Markowitz Theory using an Investors (risk/reward) Utility Function. Here the returns are given in absolute terms.

Suppose you have the historic rates of return for 5 assets given in the table:

Month	Asset1	Asset2	Asset3	Asset4	Asset5
January	100	75	120	50	150
February	105	76	120	50	150
March	107	76.5	130	50	145
April	110	77	130	51	135
May	90	70	125	45	120
June	75	71	105	46	110
July	80	73	105	46	110
August	80	75	110	47	170
September	90	77	120	48	200
October	97	79	120	49	200
November	105	78	125	50	205
December	115	77	135	51	210

Problem: What are the asset weights of the optimal portfolio which has an expected return of 120?

That is, what are the weights of the 5 assets such that the risk of the resulting portfolio is minimized and the return is equal to 120. We have chosen to use a tolerance of $1E-3$.

Solution: The required weights are:

Asset	weight
Asset1	0
Asset2	0.01067
Asset3	0.98764
Asset4	0
Asset5	0.00169

For further details we refer the reader to the source code and in-line documentation of this example.

4) Optimal2 Subdirectory

Within this example we will consider the construction the efficient frontier for the portfolio which can be constructed from three managed fund with returns of 12 periods of:

- Fund 1: 1.14, 0.47, 0.84, 0.93, 1.1, 0.82, 0.63, 0.72, 0.8, 1.06, 0.79, 0.78
- Fund 2: -6.06, -4.15, 1.37, 4.29, 1.37, -1.21, 4.11, 2.19, -3.71, 1.18, -1.25, 1.29
- Fund 3: -1.7, -4.9, 3.64, 4.14, 1.14, 1.66, 4.18, 4.01, -0.11, 4.24, -2.26, 4.33

(The above returns are given in percentage form (i.e. 1 = 1 percent = 0.01))

Within this example we will construct the efficient frontier and then ead off' the optimal portfolio for a given level of return. We will then construct the optimal portfolio for three different investors with relatively a low, medium and high risk tolerance.

Each utility function for the three investors is given on five points represented a pairs risk and reward points on the utility curve. The three investors correspond to:

- Low Risk Tolerance Investor
- Medium Risk Investor
- Higher Risk Investor

The above returns and risk correspond to monthly expected return and corresponding risk with the three investors are prepared to accept.

5) Optimal4Risk4Return Subdirectory

Within this client we apply the Markowitz Theory to construct the portfolio which has:

- The lowest risk for a given expected return (i.e. expected performance level)
- The highest expected return (i.e. performance) for a given level of risk

for six investment funds where each of the funds historical monthly returns is known over the past 3 years (approx.).

Remark In order to address the first problem it will be sufficient to apply methods from the Markowitz class, in order to solve for risk we will need to use in addition functionality from the SolveFrontier class.

6) Optimal4Risk4Return50Funds Subdirectory

Within this client we apply the Markowitz Theory to construct the portfolio which has:

- The lowest risk for a given expected return (i.e. expected performance level)
- The highest expected return (i.e. performance) for a given level of risk

for 50 investment funds where each of the funds historical monthly returns is known over the past 3 years (approx.).

The historical values of the funds is given within a array of dimension 2 where the first elements consists of 50 members which state the returns in Jan 2001, the second element the return in Feb 2001, and so on.

Remark In order to address the first problem it will be sufficient to apply methods from the Markowitz class, in order to solve for risk we will need to use in addition functionality from the SolveFrontier class.

WARNING: This example will take several minutes to run.

7) Optimal4Risk4Return200Funds Subdirectory

Within this client we apply the Markowitz Theory to construct the portfolio which has:

- The lowest risk for a given expected return (i.e. expected performance level)
- The highest expected return (i.e. performance) for a given level of risk

for 200 investment funds where each of the funds historical monthly returns is known over the past 3 years (approx.).

The historical values of the funds is given within a array of dimension 2 where the first

elements consists of 200 members which state the returns in Jan 2001, the second element the return in Feb 2001, and so on.

Remark In order to address the first problem it will be sufficient to apply methods from the Markowitz class, in order to solve for risk we will need to use in addition functionality from the SolveFrontier class.

WARNING: This examples will take more than an hour to complete on a desktop machine.

8) EfficientAlternative Subdirectory

Within this example we evaluate using the same historical values as within the example contained within the folder 'Efficient' and evaluate the portfolio on the Efficient Frontier. The difference between this example and the example contained within the 'Efficient' folder in that here we do not use the direction approach of evaluating the portfolio, but use the approach via the evaluate of the Efficient Frontier at a number of interpolation points.

The rationale for including this examples is to allow the comparison between these two approaches to the evaluation of the optimal portfolio for a given values of the expected return.

Historical Values Used

Within this example we use the following relative historical values of five assets historical returns over the last five periods:

- 1st Asset: 0.03, 0.04, 0.02, 0.05, 0.03
- 2nd Asset: 0.03, 0.040, 0.1, 0.03, 0.035
- 3rd Asset: 0.03, 0.032, 0.033, 0.035, 0.036
- 4th Asset: 0.02, 0.021, 0.021, 0.021, 0.0205
- 5th Asset: 0.02, 0.01, 0.02, 0.015, 0.028

Where the portfolios on the Efficient Frontier can be constructed with respect to the following asset weight constraints:

- 1st Asset Weight Constraints: Lower bound of = 0.05, Upper bound = 0.2
- 2nd Asset Weight Constraints: Lower bound = 0.05, Upper bound = 1
- 3rd Asset Weight Constraints: Lower bound = 0.05, Upper bound = 1
- 4th Asset Weight Constraints: Lower bound = 0.05, Upper bound = 0.5
- 5th Asset Weight Constraints: Lower bound = 0.05, Upper bound = 1

5.2.2 Capital Market Custom Clients

Within this section we describe the custom examples which we provide for the Capital Market EJB Component. The source code for these examples can be found within sub-directories of the directory:

Clients > QAClients > "Language/Technology" > Portfolio > CapitalMarket

where *"Language/Technology"* refers the program language and/or technology (for example Java, C, Delphi, EJB, COM, Web services, CORBA etc) by which the example are written in.

1) CashConstraints Subdirectory

Within this example we illustrate how constraints may be placed onto the levels of the cash held or borrowed to the market by an optimal portfolio constructed in accordance within the Capital Asset Pricing Model. We show how the resulting continuous range of the values of the expected return and total risk can then be deduced which identify which optimal portfolios can be selected when the cash level constraints are observed.

2) Equity Subdirectory

This example illustrates the construction in accordance with the Capital Asset Pricing Model (CAPM) of the Optimal Portfolio on the CML where constraints are placed on the asset weights from which the portfolios can be constructed.

This example illustrates the application of the Capital Asset Pricing Model (CAPM) functionality implemented within the class CapitalMarket.

In particular, we seek the solution to the following question:

What is the weighting of the Market Portfolio and risk of the optimal portfolios with respect to the CAPM which can be constructed from 5 assets with historical returns of:

Time interval	Asset1	Asset2	Asset3	Asset4	Asset5
1 st interval	500	500	300	200	250
2 nd interval	450	450	320	210	270
3 rd interval	500	400	330	215	300
4 th interval	550	300	350	210	280
5 th interval	600	350	360	205	290

where the weights of each of the assets of the portfolio constructed must lie between 0.1 and 0.5, that is, each asset can have a minimum weights in 10 percent and a maximum weighting of 50 percent with the constructed portfolio.

Question: What if the Market Portfolio's risk and expected return? In accordance, with

respect to the CAPM what is the optimal portfolios (i.e. lowest risk with a given expected return) which has an expected absolute return of 300 and 210?

Overview of Approach

Once the source data is given we provide to find the constrained optimal portfolio by following the below steps:

- Instantiate the Capital Market (main class) and Asset Parameter classes which provide the functionality required.
- Evaluate the covariance and expected returns of the collection of assets from which the portfolios can be constructed. This is done using two auxiliary methods from the Asset Parameters class.
- Set the constraints on the asset weights from which the portfolios are constructed.
- We then evaluate the Efficient Frontier on the entire range on which it exists and set its interpolation points to a private field.
- Next we construct the Market Portfolio.
- We then evaluate the risk and expected return of the Market Portfolio which will be used within the follow calculations.
- In order to find the optimal portfolios with respect to the CAPM we evaluate the portfolios of the CML with an expected return of 210 and 300. That is, we evaluate the weighting of the market portfolio and the risk of each of the portfolios.

Historical Source Data

The source data is we use here are the historical values of the 5 assets returns from which the portfolios on the Efficient Frontier can be constructed measured over the last 5 periods.

- 1st Asset Returns: 500, 450, 500, 550, 600
- 2nd Asset Returns: 500, 450, 400, 300, 350
- 3rd Asset Returns: 300, 320, 330, 350, 360
- 4th Asset Returns: 200, 210, 215, 210, 205
- 5th Asset Returns: 250, 270, 300, 280, 290

Business Classes used

Within this example we use the following business classes:

- Capital Market - The CapitalMarket class contain the core functionality within our implementation of the CAPM.

- Asset Parameters - The Asset Parameters class offers a number of utility methods which assist in the evaluation and estimation of the parameters of the methods within the Capital Market class.

Solution Found:

After having calculated the efficient frontier, we find the optimum market portfolio which provide the highest expected return per unit of risk. The market portfolios asset weights are as follows:

	weight
Asset1	0.08605
Asset2	0.08125
Asset3	0.0454
Asset4	0.7873
Asset5	0

Where the market portfolio has a risk of 2.42057 and has an expected return of 256.07759.

Using the market portfolio we are able to construct through either borrow cash from the market to buy more units of the market portfolio or though lending to the market; a portfolio that has any return greater than the return from cash.

Action Require to obtain a return of 300: The method *weightCML* in order to obtain a return of 300 returns 1.71912. This number is greater than 1, which means that a sum greater than total investor's wealth must be invested in the equity portfolio in order to have a return of 300. That is, 0.71912 of the initial sum will be borrowed at the market's rate and invested within the optimal portfolio. In this instance the risk of the portfolio increases to 4.16127.

Action Require to obtain a return of 210: If the investor only requires an expected return of 210, then only 0.24559 of the investors equity needs to be invested within the optimal portfolio. With the remained being lend to the market at the prevailing market rate of 195. In this instance the investors risk of the portfolio has been reduced to 0.59447.

3) SelectOptimal Subdirectory

This example illustrates the application of the Capital Asset Pricing Model (CAPM) functionality implemented within the class CapitalMarket.

In particular, we seek the solution to the following question:

What is the weighting of the Market Portfolio and risk of the optimal portfolios with respect to the CAPM which can be constructed from the assets:

- Fund 1: Historical Returns = 0.03, 0.04, 0.02, 0.05, 0.03 per year
- Fund 2: Historical Returns = 0.04, 0.045, 0.03, 0.03, 0.035 per year

- Fund 3: Historical Returns = 0.02, 0.021, 0.021, 0.021, 0.0205 per year
- Fund 4: Historical Returns = 0.02, 0.021, 0.021, 0.021, 0.0205 per year
- Fund 5: Historical Returns = 0.025, 0.027, 0.03, 0.028, 0.029 per year

where the prevailing market rate at which cash can be lent or borrowed is 1.95 percent per year?

5.2.3 Asset Parameters Custom Clients

Within this section we describe the custom examples which we provide for the Asset Parameter EJB Component. The source code for these examples can be found within sub-directories of the directory:

Clients > QAClients > "Language/Technology" > Portfolio > AssetParameters

where "Language/Technology" refers the program language and/or technology (for example Java, C, Delphi, EJB, COM, Web services, CORBA etc) by which the example are written in.

1) ForwardLooking Sub-directory

Suppose you have the following data regarding 3 securities.

State	1 st Asset	2 nd Asset	3 rd Asset	Probability
1 st state	100	120	200	0.1
2 nd state	120	130	150	0.7
3 rd state	130	140	140	0.2

Each cell in this table contains two values: the return of the asset in one of three states, and the probability of that state occurring.

In our example we evaluate the following:

- The expected variance of each asset:
 - for the 1st asset $var = 60$
 - for the 2nd asset $var = 29$
 - for the 3rd asset $var = 261$
- The expected return for each asset:
 - for the 1st asset $exp = 120$
 - for the 2nd asset $exp = 131$
 - for the 3rd asset $exp = 153$

- The covariance between the 1st and 2nd Asset is: 40

Remark Since the returns of the assets are given as absolute value of the expected returns evaluate will also be given in absolute values rather than in relative (i.e. percentage) terms.

Portfolio constructed with Equal Weighting between the three assets available

Suppose that you have a portfolio containing equal shares from each of the three Assets available. That is, the weighting of each of 1st, 2nd and 3rd asset is 1/3. For this constructed portfolio we evaluate the following properties:

- The expected return which is evaluated to be: 134.66667
- The portfolio's risk which is evaluated to be: 2.21108
- The variance of returns which is evaluated to be: 4.88889

5.2.4 Two Asset Portfolio Custom Clients

Within this section we describe the custom examples which we provide for the Two Asset Portfolio EJB Component. The source code for these examples can be found within sub-directories of the directory:

Clients > QAClients > "Language/Technology" > Portfolio > TwoAssetPortfolio

where "Language/Technology" refers the program language and/or technology (for example Java, C, Delphi, EJB, COM, Web services, CORBA etc) by which the example are written in.

1) OptimalDiversification Sub-directory

Suppose you have two assets, for which you know three possible returns may take place and you can assign corresponding probabilities for each of these returns taking place in accordance with the following table:

State	Asset1	Asset2	Probability
1 st state	300	500	0.5
2 nd state	310	400	0.2
3 rd state	330	550	0.3

Problem:

Say the first asset represents a portfolio which is held and the second asset represents an asset which the portfolio manager wishes to add to the portfolio such the optimal effects (i.e. minimize the risk) of diversification from the purchase is felt. The question to address here now many of shares of Asset 2 should the portfolio manager buy in order to

acheive this?

Method Used:

Our implementation which can be found within the OptimalDiversification folder when run finds the following answers. Within this implementation we first evaluate the standard deviation of each of the two assets before evaluating the covariance from which we are able to evaluate the optimal weight using:

```
:TwoAssetPortfolio:weight2MinimizeRisk:
```

For further detail we refer the reader to the code of this example or order to see exactly how these solutions we arrived at.

5.2.5 Optimal portfolio

If you have the statistical parameters for a set of securities (the covariance matrix and the vector of expected returns), you can calculate the efficient frontier. Then, given a utility function, you can find what is the set of optimal portfolios in the sense of Markowitz model.

We use the same historic rates of return as in the previous example. The utility function is given as a set of interpolation points:

	1	2	3	4	5	6	7
x	0	40	60	80	100	120	140
y	0	6.0	6.4	7.0	14.5	22.0	29.2

There are 2 optimal portfolios:

	Portfolio 1
Asset1	0
Asset2	0
Asset3	0.269
Asset4	0
Asset5	0.7309

Modifying the Utility Function: You can change the utility function at any moment by calling one of the initialization functions. Then, you can calculate the new set of optimal portfolios. Suppose that you have a polynomial utility function:

$$ut(x) = 0.00117 * x^2 - 0.0443 * x$$

Again, we find 2 optimal portfolios:

	Portfolio 1	Portfolio 2
Asset1	0	0
Asset2	0.6327	0.0022
Asset3	0.0004	0.7769
Asset4	0.3667	0
Asset5	0	0.2207

5.3 Database Example with JDBC Mediator

The Database Example is located inside the *Client/Portfolio/DatabaseExample* directory. This examples is provided in order to detail how this component may be used with DBMS's. Before being able to run this examples you will need to complete the following steps:

1. Installing our Tables (MySQL Only)

In order to create the database structure from which you are able to load the output results, you will need to run the 'create.sql' scripts in the 'Data' subdirectory. Open the MySQL prompt from the 'Data' subdirectory and:

- Select (or create and then select) a database you can spare for a table named HISTORICAL.RETURNS. For example, if you have decided using the 'test' database, you would optinally type the SQL command to create it (unless it already exists):

```
CREATE DATABASE test;
```

and then, you would type the following to select it:

```
USE DATABASE
```

- Run the 'create.sql' SQL script located in the 'Data' subdirectory of the current directory by typing at the same MySQL prompt the following:

```
source create.sql
```

If this fails, make sure you have started the MySQL prompt from the 'Data' subdirectory (this is where the database files for this example are stored).

2. Configuring the Database Connection

Edit the Java source code file by filling out JDBC information about your database, such as:

- (a) Driver Name (e.g. `com.jdbc.mysql.Driver`)
- (b) JDBC Url (e.g. `jdbc:mysql://localhost/test`)
- (c) Username and Password

If you are unsure whether you have a JDBC Driver for your Database, skip this step and try running the example with the predefined values. If that fails, you will need to probably download the latest driver.

3. Running the example

Run the ‘compile’ script (compile.sh for Linux, compile.bat for Windows) to compile the Java source code, and then the ‘run’ script to see the results.

Uninstalling the Source Data

To delete all the tables used in this example, open the MySQL prompt from the ‘Data’ subdirectory, select the database where you created the tables, and run the following:

```
source delete.sql
```

5.4 How to use Markowitz Theory?

Within this section we provide several step-by-step guides on how to perform many of the most required tasks when apply Markowitz Theory. The methods references within this section will correspond to methods of the same name found within the Markowitz, AssetParameters or PerformanceEvaluation EJB Component’s.

5.4.1 How to find the portfolio from a collection of assets that exhibit the lowest risk for a given expected future return?

The problem of finding the optimal portfolio with the lowest risk with a given expected return from a collection of assets of which the historical performance is known, involves applying two methods, namely:

- `calculateEfficientFrontier` - we construct the efficient frontier that consists of a continuously varying family of portfolios
- `efficientFrontier` - this method selects a portfolio determined by its expected return from the family of portfolios on the efficient frontier

Constructing such a portfolio is the main application of classical Markowitz Theory. That is, if you take a collection of assets historical performance from which you imply expected performance. By evaluating the covariance matrix of the assets (see section entitled, ‘How to evaluate the covariance matrix?’) and the efficient frontier (see section entitled, ‘How to construct the Efficient Frontier?’), you may evaluate the optimal risk by applying the method:

```
efficientFrontier(double expectedReturn, int numberOfAssets)
```

from the Markowitz component.

Remark When we refer to the optimal risk we mean the minimal risk for a given expected return which is the same thing as saying the portfolio which is expected to exhibit the smallest standard deviation of its value for a given level of expected return.

5.4.2 How to construct the Efficient Frontier?

The Efficient Frontier is constructed by applying the following Markowitz EJB Component method:

```
public void calculateEfficientFrontier(double minimumExpectedReturn,
    double maximumExpectedReturn,
    double[] [] covarianceMatrix,
    double[] expectedReturns,
    int numberInterpolationPoints,
    double precision)
```

The developer will need to provide the correct input parameters starting with the following two:

```
double minimumExpectedReturn
double maximumExpectedReturn
```

The minimum and maximum expected returns can be taken to be the minimum and maximum historical returns from the best performing and worst performing assets within the given collection which we will use to construct the optimal portfolio. Since these two extremes certainly suffice (i.e. allow all potential optimal portfolio's to be considered), and in the general case these extreme values may even themselves represent optimal portfolios, which in fact only be the case if the covariance matrix is the Identity matrix.

```
double[] [] covarianceMatrix
```

We apply the method 'covarianceMatrix' within the AssetParameters EJB Component to the database table that contains the assets historical values from which the portfolio will be constructed. The historical data we are provided with should be stored within a database table in order to apply this method directly.

```
double[] expectedReturns
```

Please see the section entitled "How to estimate the expected return for the historical

returns?”

```
int numberInterpolationPoints, double precision
```

The input value's for these parameters is just a matter of balancing speed and accuracy.

Remark The Markowitz EJB Component uses highly optimized algorithms to construct the Efficient Frontier that leads to the construction of the optimal risk-return portfolio. This allows the optimal Portfolio to be constructed in one or two minutes for 100+ assets using a commodity desktop PC. Saving the end user time and offering the convenience of being able to use a desktop PC equipment rather than specialist server hardware.

5.4.3 How to estimate the expected return from the historical returns?

Say your source data is the 'historical' monthly performance of a given fund. If we assume that the historical returns of this fund will be closely related to the future expected returns. Then we will be able to estimate the future expected returns using the method:

```
expectedReturn(double[])
```

from the AssetParameters EJB Component.

Further Discussion

Here as in all estimates of the fluctuation of the price (or risk) there will always be two types of risk related to the application of the model. Namely the market risk itself and the risk that the market risk (or structure) will change. The success of any model will depend on its ability to adapt to changes in the market dynamics.

Towards this end we offer a number of approaches to estimating the expected returned which one could use for a variety of differing market dynamics. These procedures are contained within the AssetParameter EJB Component and include:

- Arithmetic Historical average - Take the historical return over a given period and take the expected future return to be the arithmetic average of the historical values. Say we take the last 6 monthly historical returns, then the probability of the market states which deliver each of these 6 historically recorded returns over the last 6 months is $\frac{1}{6}$.
- ARCH Type Model - This is an averaging procedure which attaches more weight to more recently measured values. For example, we could take:

$$\text{expectedReturn} = \sum (\exp(-i) \text{historicalReturn}[i])$$

Say we take the last 6 monthly value again in order to estimate the expected future return. The probability of the market states that delivered each of the return found in the last six months is:

$$\frac{\exp(-i)}{\sum_i \exp(i)}$$

for $i = 0, \dots, 6$.

Our implementation of the ARCH model assumes that the underlying asset has a drift in accordance to the characteristic line. Under this assumption the ARCH model provides an effective model for assets, which depend linearly on stocks or stock indexes.

Remark Predictive models of ARCH/GARCH type or models involving technical or fundamental analysis are not prescriptive and hence will depend heavily on the choices of the parameters the user makes. Saying this, these models enable the user to ensure that whatever the variables used the resulting model will exhibit certain qualitative properties which are embodied within the model itself.

5.4.4 How can I measure the performance characteristics of a portfolio?

One of the core principles of ‘Markowitz Theory’ (and CAPM), is that investors will only accept higher risk for a higher expected return. Therefore, since we are working within this framework it is only natural to consider the risk adjusted performance of a portfolio and related methods thereof. Within this product we have including methods by which the risk adjusted and non-risk adjusted return can be evaluated.

Risk Adjusted Performance

In order to evaluate the risk adjusted return we use methods from the PerformanceEvaluation EJB Component. In particular, we use one of the following two methods:

- Sharpes Ratio - This performance measurement parameter is the most widely used. The higher the value the better the portfolios risk return profile is said to be. To evaluate Sharpes Ratio use the method `SharpesRatio`.
- Treynors Ratio - This measure takes into account the systematic risk (or beta) and the average return when assessing the overall risk adjusted return of the portfolio. To evaluate Treynor’s performance measure use the method `treynorsMeasure`.

Non-risk adjusted Performance

The non-risk adjusted performance of a portfolio can be evaluated by using the methods Total Return and Portfolio Return within the performance evaluation EJB Component. The Portfolio Return method evaluates the arithmetic return of the portfolio over a period

of time where the return of each asset over the period and its initial portfolio weight is known. The total return method allows you to evaluate the return from the portfolio when cash disbursements or dividends are taken into account.

Even though these measures are not risk adjusted the user can still use them in conjunction with a measure of the risk of the portfolio in order to get a better feel for a portfolio's overall characteristics. The risk of a portfolio, can be evaluated by the method:

```
portfolioRisk(double[] weight, double[][] covarianceMatrix)
```

which is contained within the portfolio EJB Component.

Chapter 6

Guide to WebCab Components

6.1 The Company

WebCab is a privately owned British company that has built business solutions since its inception in 1999. We continue to refine and develop our Mathematical and Financial Framework which we have implemented on the J2SE, J2EE and .NET platforms.

6.2 Presentation of Products

We aim to provide good quality, useful information to help you decide which J2EE application best suits your development needs. WebCab is committed to honesty and realism when presenting our products. In order to achieve this a detailed, clear and factual style for presentation is adopted within our documentation and marketing material.

6.3 Supported Clients, IDEs, Containers and DBMSs

We support all major development, server and client side technologies. Each product contains detailed examples and advice concerning the integration of our enterprise applications within existing infrastructure. Our documentation provides the information that the developer needs to get their applications up and running as quickly and as easily as possible.

We detail exactly how the developer can use our J2EE Applications with the following technologies:

- Client containers - Internet Explorer, Netscape Navigator, Opera
- IDEs - JBuilder, Eclipse, BEA Studio, Sun ONE (formerly Forte For Java), IBM VisualAge, Oracle JDeveloper
- Client Side Technologies - Application Clients, Servlet, JSP, Applets
- EJB containers - IBM WebSphere, BEA WebLogic, Oracle 9iAS, Sun ONE AppServer, Borland AppServer, Sybase EAServer, Ironflare Orion, JBoss

- DBMS - Oracle, IBM DB2, SQL Server, Sybase, MySQL
- Platforms - Windows XP/2000/NT, Linux, Solaris, IBM AIX

For these technologies we include all the deployment descriptors, installation scripts and template examples which will help the developer quickly and easily assemble their multi-tier enterprise application.

6.4 Transparent Functionality

All technical and business intelligence incorporated within our products is described within the associated documentation. This allows the developer to see the nature of the methodology implemented within our proprietary algorithms.

6.5 Code Conventions

WebCab adheres to the 'Code Conventions for the Java Programming Language' as specified on April 20, 1999 by Sun Microsystems Inc. These conventions cover filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions and programming practices. Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

6.6 Company Culture and Activity

WebCab Components is focused solely on the production of high quality software modules within our Financial and Mathematical Framework. The WebCab team contains a wide range of expertise and experience from the academic, investment banking and software development worlds.

Within the company there exists significant internal competitive pressures with constant peer review and evaluation, resulting in higher quality products, which adhere to high professional standards and offer extended functionality.

6.7 Product Life cycle

We continuously add value to our products by evolving them according to new J2EE specifications, customer feedback and market demands. We give particular emphasis to incorporating our clients suggested modifications and additions into our product development cycle.

6.8 Support, Warranty and Upgrades

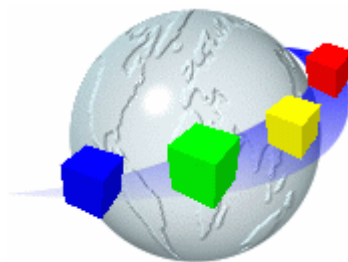
WebCab warrants that each component will perform substantially in accordance with the accompanying written material. We provide with all our products without charge sixty (60) days product support services including fixing bugs, compatibility issues and other technical support issues.

All maintenance updates (including service packs) will be distributed free of additional license cost. New version releases of this product will be offered to present licensee holders at a greatly reduced rate.

Dr Ben Fairfax

Founder and CEO

WebCab Components - From Developer, To Developer



Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. .NET is a trademark of Microsoft Corporation in the U.S. and other countries