Version 3.1

OFC Charts, User Guide

Document Type	**User Guide**
Reference	OFCChartsUserGuide_eng.doc
Date	August 18, 2006

Table of Content

1. Introduction

OFC Charts is a 100% JAVA class library with a simple API to create graphs and charts.

Supporting several output formats, the generated images can be embedded into any GUI such as Web pages, .PDFs, etc.

1.1 Supported output formats

OFC-Charts can create images as :

- JPEG : call method `drawJPEG( … )`
- PNG : call method `drawPNG( … )`
- SVG : call method `drawSVG( … )`
- Animated SVG : call method `drawAnimateSVG( … )`

1.2 Display types

OFC-Charts can create several types of charts:

- Curves (**VerticalLinePlot**)
- Histograms which can be :
    - Vertical or horizontal (**VerticalBarChart**)
    - Surface curves (**VerticalLineChart**)
    - Stacked (**VerticalStackedBarChart**)
    - Combined charts with one or two Y axis (**CombinedVerticalBarChartXY , CombinedVerticalBarChartXYY**). When using combined charts, the types of graphs can be different, i.e. one can be a curve and the other one a BarChart.
- Pie Charts or Half Pie charts (**PieChart, ExplodedPieChart**)
- 2D contour charts (**Contour2D**). This type of chart creates a 'flat' representation of a 3D surface chart : X axis and Y axis determine a Z value which is encoded as a color and isolines drawn between adjacent colored zones.

Input parameters are graph sizes in pixels, axis types, and data.

Graphs can be animated within an SVG file by displaying each data set after the other.

1.3 Presentation attributes

OFC-Charts offers a large choice of attributes and properties which give full control on the look and feel of the generated graphics:

- Image size and graph position within the drawing area ;
- Background color ;
- Custom color palette;
- Custom Legend: position, font and color.
- Visible or invisible grid with its color ;

- Display or hide axis with control over horizontal and vertical axis graduation, color, arrow, label and font. Each axis scale can be set or automatically computed ;
- Axis position can be controlled and an offset assigned in order to improve readability. Labels can be displayed at an angle on the horizontal axis ;
- For line charts each value can be displayed as a circle, a triangle or a square ;
- 2D graphs can easily be turned into 3D using the coeff3D parameter.

In the following chapters you will find several examples which will allow you to better understand how OFC Charts can be used.

2. Vertical Line Plot

2.1 Description

The ***VerticalLinePlot*** class plots one or several lines.

Each line is defined by *Series* of X,Y coordinates.

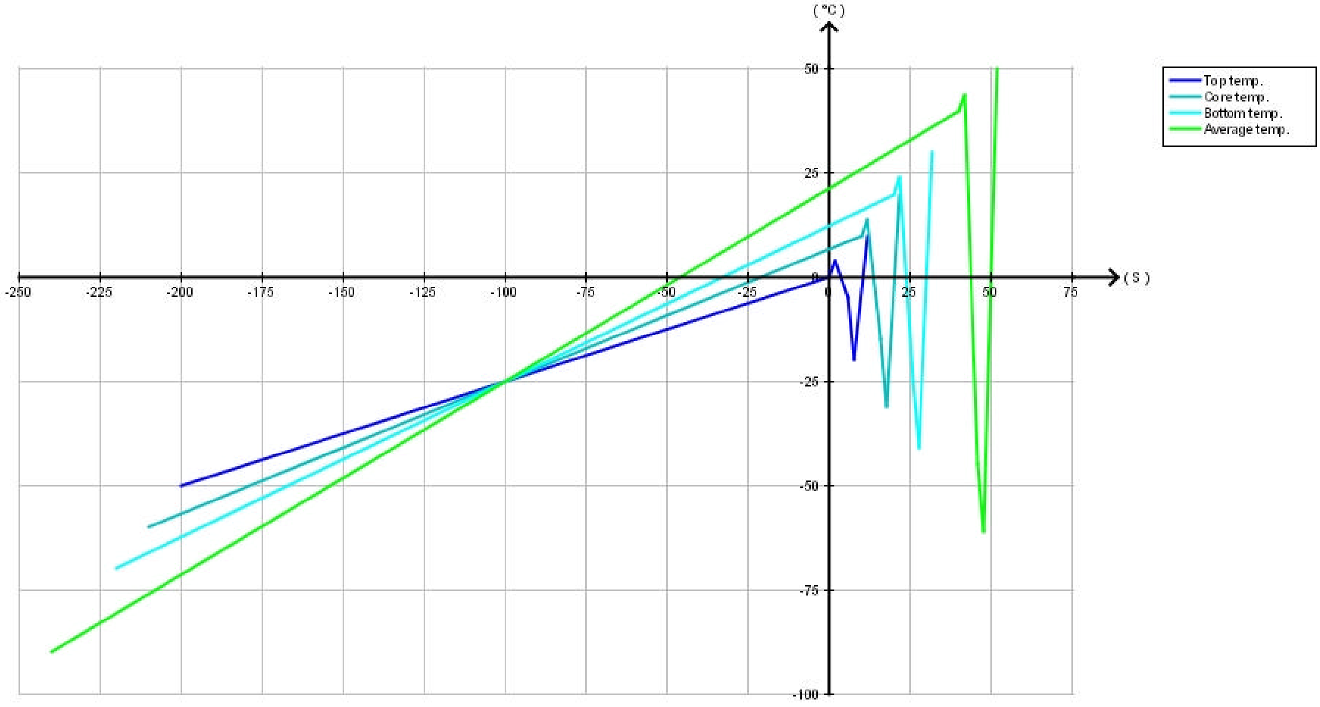
Each *Serie* is identified by its string id and contains:

- A legend label,
- A set of X, Y values.

Since a single graphic can contain multiple lines, each of them being a *serie*, a *series* container manages the complete set of series.

2.2 Example

In the following example 4 lines, i.e. 4 series, are displayed:



2.3 Sample Java Code

The *Point2DSeries* class contains the set of points and the legend label for each line.

The *Point2DSeriesContainer* contains all four *Point2DSeries*.

The *VerticalLinePlot* class draws the graphic using the desired size (height and width).

```
// Create the container for the4 lines
Point2DSeriesContainer point2DSeriesContainer = new Point2DSeriesContainer();

// Line 1 (create the first line with id and legend - then add the points)
Point2DSeries point2DSeries = new Point2DSeries("S1", "Top temp.");
point2DSeries.addPoint2D(new Point2D(-200, -50));
point2DSeries.addPoint2D(new Point2D(0, 0));
point2DSeries.addPoint2D(new Point2D(2, 4));
point2DSeries.addPoint2D(new Point2D(6, -5));
point2DSeries.addPoint2D(new Point2D(8, -20));
point2DSeries.addPoint2D(new Point2D(12, 10));

// Add the first line to the container
point2DSeriesContainer.addPoint2DSeries(point2DSeries);

// Line 2 (create the second line with id and legend - then add the points)
point2DSeries = new Point2DSeries("S2", "Core temp.");
point2DSeries.addPoint2D(new Point2D(-210, -60));
point2DSeries.addPoint2D(new Point2D(10, 10));
point2DSeries.addPoint2D(new Point2D(12, 14));
point2DSeries.addPoint2D(new Point2D(16, -15));
point2DSeries.addPoint2D(new Point2D(18, -31));
point2DSeries.addPoint2D(new Point2D(22, 20));

// Add the second line to the container
point2DSeriesContainer.addPoint2DSeries(point2DSeries);

// Create line 3
point2DSeries = new Point2DSeries("S3", "Bottom temp.");
point2DSeries.addPoint2D(new Point2D(-220, -70));
point2DSeries.addPoint2D(new Point2D(20, 20));
point2DSeries.addPoint2D(new Point2D(22, 24));
point2DSeries.addPoint2D(new Point2D(26, -25));
point2DSeries.addPoint2D(new Point2D(28, -41));
point2DSeries.addPoint2D(new Point2D(32, 30));

// Add line 3
point2DSeriesContainer.addPoint2DSeries(point2DSeries);

// Create line 4
point2DSeries = new Point2DSeries("S4", "Average temp.");
point2DSeries.addPoint2D(new Point2D(-240, -90));
point2DSeries.addPoint2D(new Point2D(40, 40));
point2DSeries.addPoint2D(new Point2D(42, 44));
point2DSeries.addPoint2D(new Point2D(46, -45));
point2DSeries.addPoint2D(new Point2D(48, -61));
point2DSeries.addPoint2D(new Point2D(52, 50));

// Add line 4
point2DSeriesContainer.addPoint2DSeries(point2DSeries);

// Generate the JPEG image
// Set chart size
int imageHeight = 500; int imageWidth = 900;
// Create the chart with legend for each axis
VerticalLinePlot verticalLinePlot = new VerticalLinePlot(imageWidth, imageHeight, new HorizontalNumberAxis("(   S )"),
new VerticalNumberAxis( "( °C )"), point2DSeriesContainer);
// Create image as a JPEG file
 verticalLinePlot.drawJPEG("test.jpg");
```

3. Vertical Bar Chart

3.1 Description

The ***VerticalBarChart*** class creates single or multiple bar charts.

The X axis is split into several *categories*. Within the *Series* of numerical data to be displayed as bars, each individual data is assigned to a category.

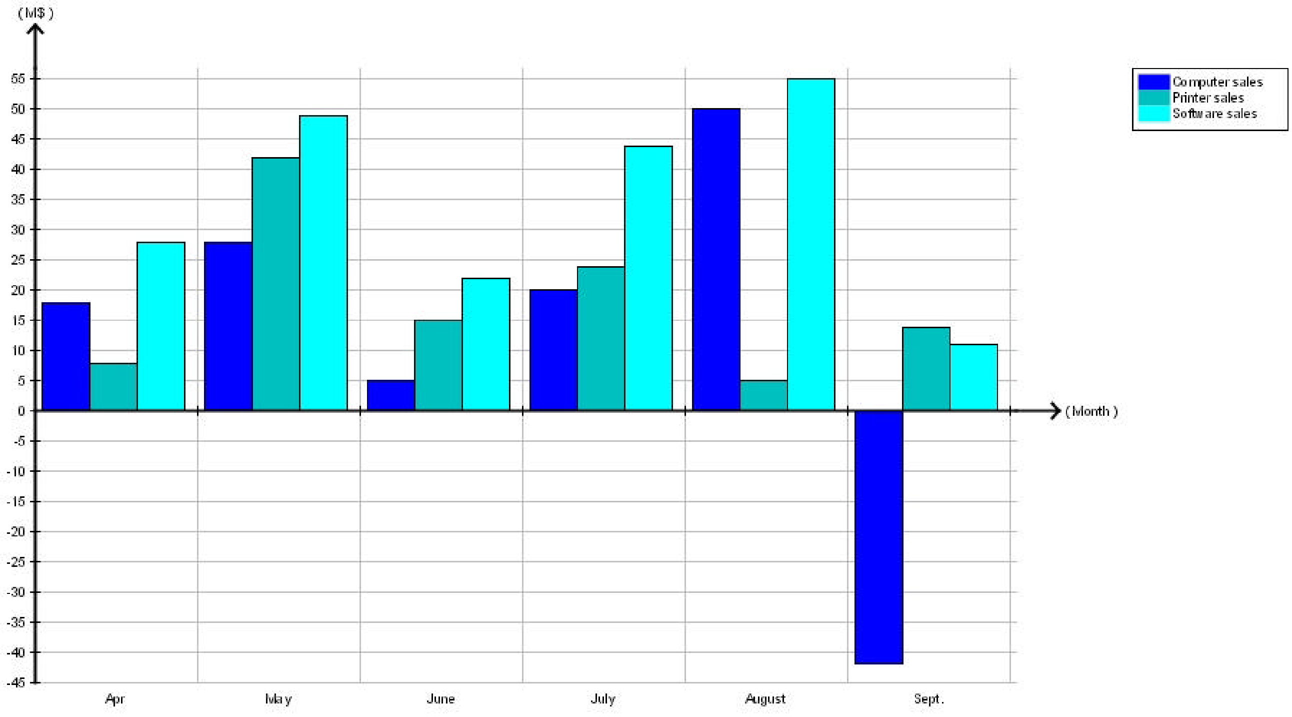
Each *Value Series* is identified by its id and contains

- A legend label,
- A set of Category/Data couples.

Categories and series are stored in a container.

3.2 Example

The following example gives a monthly display of the difference between sales forecast and real sales for 3 different product lines (computers / printers / software) over a 6 month period. Each *Category* corresponds to a particular month.



3.3 Sample Java Code

The *CategoryValueSeriesContainer* class contains all three series and the 6 categories.

The *CategoryValueSeries* class contains the legend label and the data set for each *Serie*.

The *VerticalBarChart* class draws the graphic using the desired size (height and width).

```
// Create the Container for Series and Categories
CategoryValueSeriesContainer categoryValueSeriesContainer = new CategoryValueSeriesContainer();

// Create the different categories (month)
categoryValueSeriesContainer.addCategory(new Category("id1","Apr"));
categoryValueSeriesContainer.addCategory(new Category("id2","May"));
categoryValueSeriesContainer.addCategory(new Category("id3","June"));
categoryValueSeriesContainer.addCategory(new Category("id4","July"));
categoryValueSeriesContainer.addCategory(new Category("id5","August"));
categoryValueSeriesContainer.addCategory(new Category("id6","Sept."));

// First Serie (create the serie and add values for each category)
CategoryValueSeries categoryValueSeries = new CategoryValueSeries("S1", "Computer sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 18));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 28));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 5));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 20));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 50));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", -42));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Second Serie (create the serie and add values for each category)
categoryValueSeries = new CategoryValueSeries("S2", "Printer sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 8));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 42));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 15));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 24));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 5));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 14));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Third Serie
categoryValueSeries = new CategoryValueSeries("S3", "Software sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 28));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 49));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 22));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 44));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 55));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 11));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Generate JPEG image
int imageHeight = 500; int imageWidth = 900;
VerticalBarChart verticalBarChart = new VerticalBarChart(imageWidth, imageHeight, new HorizontalCategoryAxis("( Month )"), new
VerticalNumberAxis("( M$ )"), categoryValueSeriesContainer);
verticalBarChart.drawJPEG("test.jpg");
```

4. Vertical Line Chart

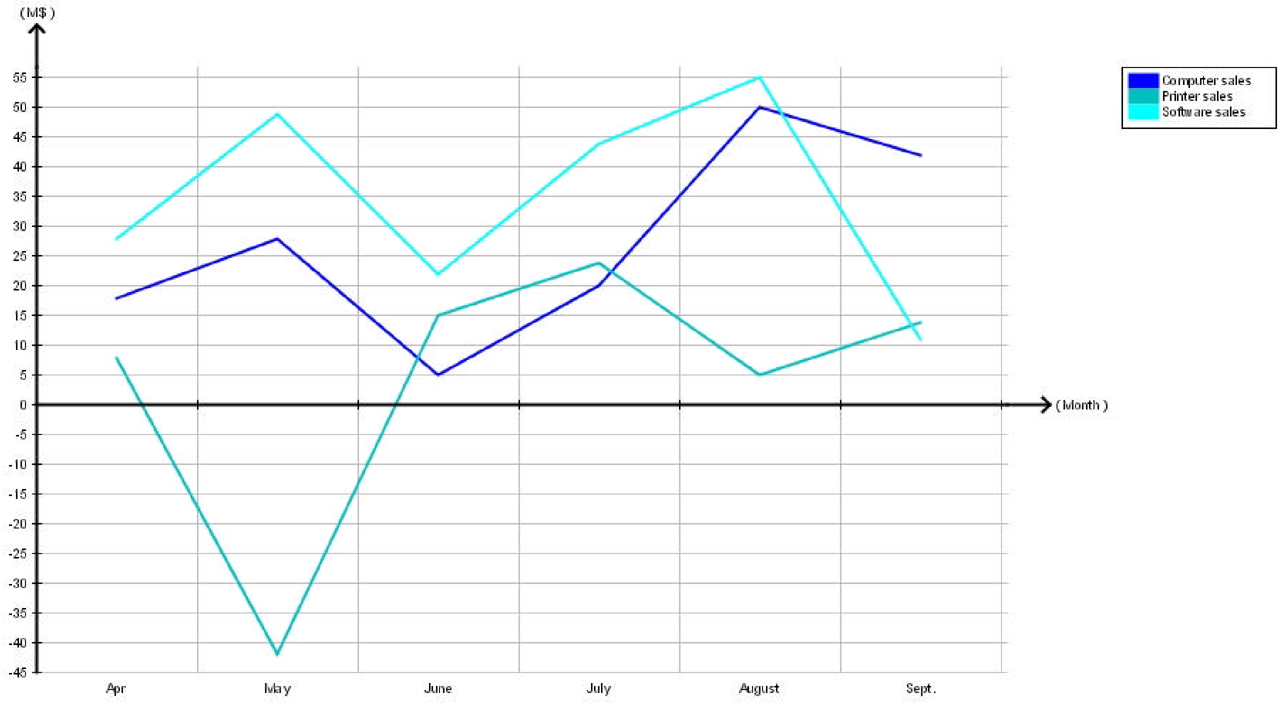
4.1 Description

The ***VerticalLineChart*** class creates single or multiple line charts.

This type of graphic is similar to the previous one. Instead of creating bars, we draw lines that go through the top of each bar. The same data set is used to create the chart.

4.2 Example

We are using the same example as above.



4.3 Sample Java Code

The *CategoryValueSeriesContainer* class contains all three series and the 6 categories.

The *CategoryValueSeries* class contains the legend label and the data set for each *Serie*.

The *VerticalLineChart* class draws the graphic using the desired size (height and width).

```
// Create the Container for Series  and Categories
CategoryValueSeriesContainer categoryValueSeriesContainer = new CategoryValueSeriesContainer();

// Create the different categories (month)
categoryValueSeriesContainer.addCategory(new Category("id1","Apr"));
categoryValueSeriesContainer.addCategory(new Category("id2","May"));
categoryValueSeriesContainer.addCategory(new Category("id3","June"));
categoryValueSeriesContainer.addCategory(new Category("id4","July"));
categoryValueSeriesContainer.addCategory(new Category("id5","August"));
categoryValueSeriesContainer.addCategory(new Category("id6","Sept."));

// First  Serie (create the serie and add values for each category)
CategoryValueSeries categoryValueSeries = new CategoryValueSeries("S1", "Computer sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 18));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 28));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 5));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 20));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 50));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 42));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Second Serie (create the serie and add values for each category)
categoryValueSeries = new CategoryValueSeries("S2", "Printer sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 8));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", -42));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 15));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 24));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 5));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 14));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Third Serie
categoryValueSeries = new CategoryValueSeries("S3", "Software sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 28));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 49));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 22));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 44));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 55));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 11));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Generate JPEG image
int imageHeight = 500; int imageWidth = 900;
VerticalLineChart verticalLineChart = new VerticalLineChart(imageWidth, imageHeight, new HorizontalCategoryAxis("( Month )"), new
VerticalNumberAxis("( M$ )"), categoryValueSeriesContainer);
verticalLineChart.drawJPEG("test.jpg");
```

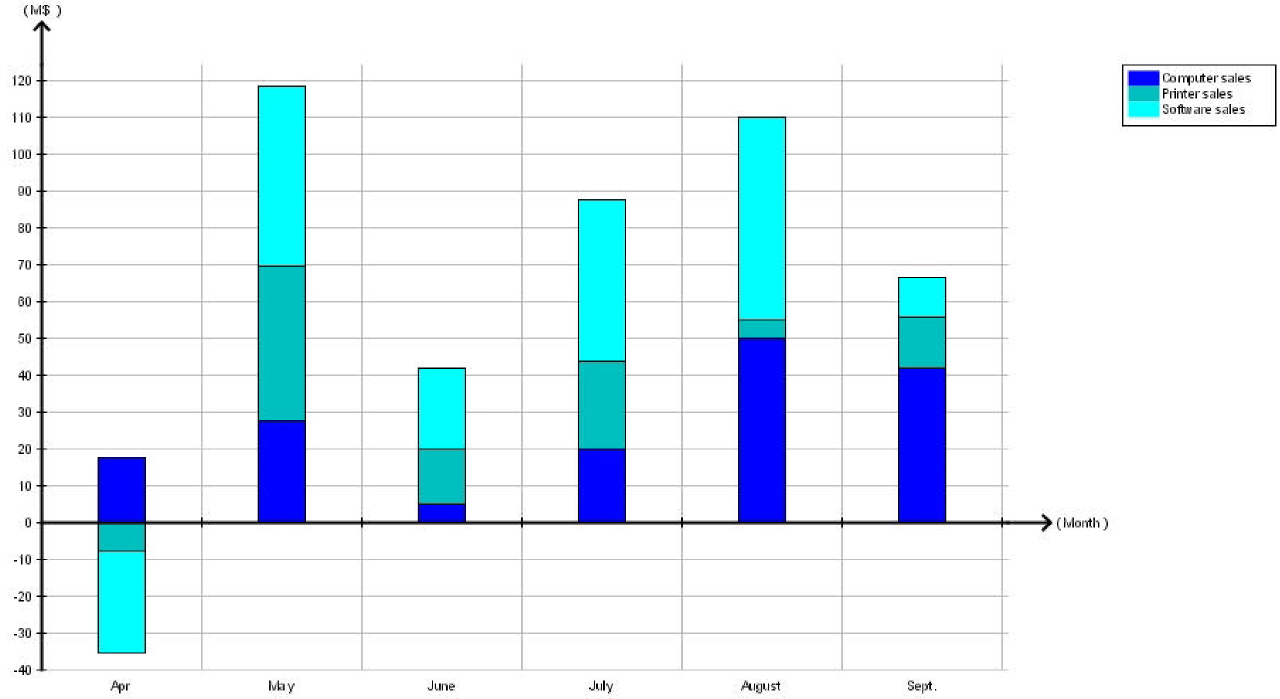
5. Vertical Stacked Bar Chart

5.1 Description

The ***VerticalStackedBarChart*** class creates stacked bar charts. For each category, the different values are displayed as colored bars stacked one above (or under) the other. Each positive value is stacked above the X axis, when negative values are stacked below the axis.

5.2 Example

We are using the same example as above.



5.3 Sample Java Code

The *CategoryValueSeriesContainer* class contains all three series and the 6 categories.

The *CategoryValueSeries* class contains the legend label and the data set for each *Serie*.

The *VerticalStackedBarChart* class draws the graphic using the desired size (height and width).

```
// Create the Container for Series  and Categories
CategoryValueSeriesContainer categoryValueSeriesContainer = new CategoryValueSeriesContainer();

// Create the different categories (month)
categoryValueSeriesContainer.addCategory(new Category("id1","Apr"));
categoryValueSeriesContainer.addCategory(new Category("id2","May"));
categoryValueSeriesContainer.addCategory(new Category("id3","June"));
categoryValueSeriesContainer.addCategory(new Category("id4","July"));
categoryValueSeriesContainer.addCategory(new Category("id5","August"));
categoryValueSeriesContainer.addCategory(new Category("id6","Sept."));

// First  Serie (create the serie and add values for each category)
CategoryValueSeries categoryValueSeries = new CategoryValueSeries("S1", "Computer sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 18));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 28));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 5));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 20));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 50));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 42));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Second Serie (create the serie and add values for each category)
categoryValueSeries = new CategoryValueSeries("S2", "Printer sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", -8));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 42));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 15));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 24));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 5));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 14));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Third Serie
categoryValueSeries = new CategoryValueSeries("S3", "Software sales");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", -28));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 49));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 22));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 44));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 55));
categoryValueSeries.addCategoryValue(new CategoryValue("id6", 11));

// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);

// Generate JPEG image
int imageHeight = 500; int imageWidth = 900;
VerticalStackedBarChart  verticalStackedBarChart = new VerticalStackedBarChart(imageWidth, imageHeight,
    new  HorizontalCategoryAxis("( Month )"), new VerticalNumberAxis("( M$  )"), categoryValueSeriesContainer);
VerticalStackedBarChart.drawJPEG("test.jpg");
```

6. Combined Vertical Bar Chart XYY

6.1 Description

The ***CombinedVerticalBarChartXYY*** class allows combining vertical bar charts having two different Y axis. The categories of each bar chart must be identical as they share the same X axis. However, two different Y axis are displayed, one on the left side, the other on the right.

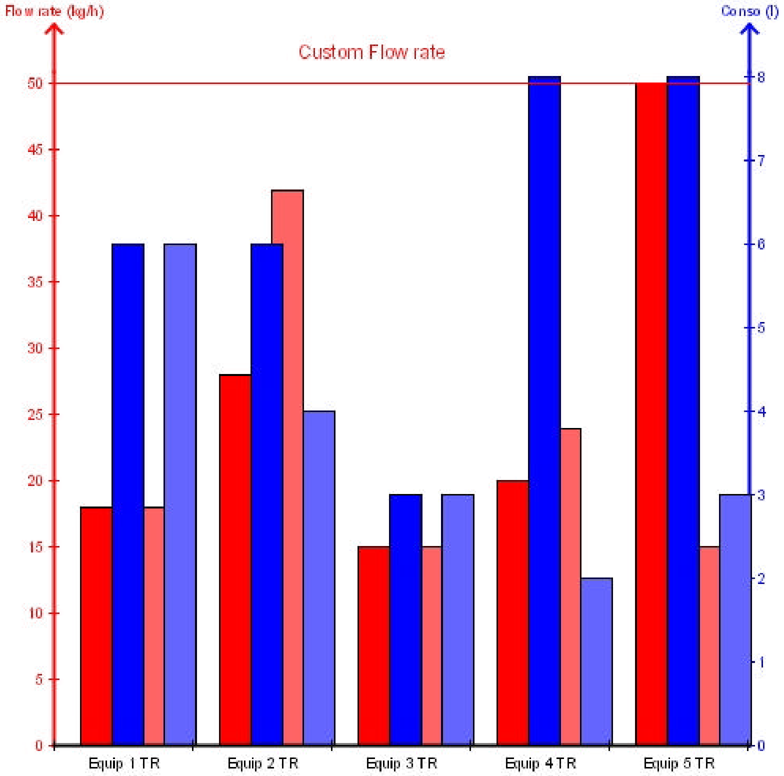
The renderer allows to define left or right positioning of the Y axis as well as control the horizontal offset of each bar.

It is possible to add a horizontal line with a label.

6.2 Example

The following example contains 2 Bar Charts, each of them with 2 series. The red bars are assigned to the left Y axis, while the blue ones are assigned to the right Y axis.

A horizontal information line has been added with value 50 on the left Y axis.



6.3 Sample Java Code

The *CategoryValueSeriesContainer* class contains all series and categories for each Bar Chart. A slightly different programming approach is used with an intermediate *Vector* containing the 5 common categories for the X axis. The Vector is then used for both the red and blue Bar Charts.

The *CategorieValueSerie* class contains the legend label and the data set for each *Serie.*

The *VerticalBarChartRenderer* class contains the series for each Bar Chart. It controls the visual display of the chart, axis position and bar width.

The *CombinedVerticalBarChartXYY* class draws graphic using the desired size (height and width) plus the information in the *VerticalBarChartRenderer*.

```
// Create Container for Series  and Categories
CategoryValueSeriesContainer categoryValueSeriesContainer = new CategoryValueSeriesContainer();

// 1 – Create categories common to both charts
Vector allCategories = new Vector();
allCategories.addElement(new Category("id1","Equip 1 TR"));
allCategories.addElement(new Category("id2","Equip 2 TR"));
allCategories.addElement(new Category("id3","Equip 3 TR"));
allCategories.addElement(new Category("id4","Equip 4 TR"));
allCategories.addElement(new Category("id5","Equip 5 TR"));

// 2 – Create the red bar chart (left Y axis)
// Define categories
categoryValueSeriesContainer.addCategory(allCategories.elements());
// First  Serie (create the serie and add values for each category)
CategoryValueSeries categoryValueSeries = new CategoryValueSeries("S1", "");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 18));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 28));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 15));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 20));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 50));
categoryValueSeries.setColor(ColorPalette.getColor(255, 0, 0));
// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);
// Second  Serie (create the serie and add values for each category)
categoryValueSeries = new CategoryValueSeries("S2", "");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 18));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 42));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 15));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 24));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 15));
categoryValueSeries.setColor(ColorPalette.getColor(255, 100, 100));
// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);
// Create red bar chart
VerticalBarChartRenderer renderer1 = new VerticalBarChartRenderer(categoryValueSeriesContainer);


// 3 - Create the blue bar chart (right Y axis)
// Create series container
categoryValueSeriesContainer = new CategoryValueSeriesContainer();
// Define categories
categoryValueSeriesContainer.addCategory(allCategories.elements());
// First  Serie (create the serie and add values for each category)
categoryValueSeries = new CategoryValueSeries("S1", "");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 6));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 6));
categoryValueSeries.addCategoryValue(new CategoryValue("id3", 3));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 8));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 8));
categoryValueSeries.setColor(ColorPalette.getColor(0, 0, 255));
// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);
// Second  Serie (create the serie and add values for each category)
categoryValueSeries = new CategoryValueSeries("S2", "");
categoryValueSeries.addCategoryValue(new CategoryValue("id1", 6));
categoryValueSeries.addCategoryValue(new CategoryValue("id2", 4));
categoryValueSeriesaddCategoryValue(new CategoryValue("id3", 3));
categoryValueSeries.addCategoryValue(new CategoryValue("id4", 2));
categoryValueSeries.addCategoryValue(new CategoryValue("id5", 3));
categoryValueSeries.setColor(ColorPalette.getColor(100, 100, 255));
// Add to the container
categoryValueSeriesContainer.addCategoryValueSeries(categoryValueSeries);
// Create blue bar chart
VerticalBarChartRenderer renderer2 = new VerticalBarChartRenderer(categoryValueSeriesContainer);
 renderer2.setPercentageBetweenEachBarHisto(15);
 renderer2.setPercentageFreeBeforeSeries(20);
 renderer2.setPercentageFreeAfterSeries(20);
// Assign it to the right Y axis
```

```
renderer2.setIsBasedOnAxisRight(true);
// Set the offset from between the red and the blue charts
renderer2.setNbPixelToMoveRendererStartPosition(20);

// 4 - Generate JPEG image
// Create object
int imageHeight = 500; int imageWidth = 500;
CombinedVerticalBarChartXYY combinedVerticalBarChart = new CombinedVerticalBarChartXYY(
         imageWidth, imageHeight, new HorizontalCategoryAxis(""),
        new VerticalNumberAxis("Flow rate (kg/h)"), new VerticalNumberAxisRight("Conso (l)"));
// Add  red bar chart
combinedVerticalBarChart.addVerticalBarChartRenderer(renderer1);
// Add blue bar chart
combinedVerticalBarChart.addVerticalBarChartRenderer(renderer2);
// Add horizontal line
HorizontalLineInfo  lineInfo  = new HorizontalLineInfo("Custom Flow rate", 50);
lineInfo.setLineColor(ColorPalette.getColor(255, 0, 0));
combinedVerticalBarChart.addHorizontalLineInfo(lineInfo);
// Generate image
combinedVerticalBarChart.setLegendDisplayMode(Legend.LEGEND_DISPLAY_MODE_NONE);
combinedVerticalBarChart.drawJPEG("test.jpg");
```

7. Pie Chart

7.1 Description

The ***ExplodedPieChart*** class creates full and half pie charts.

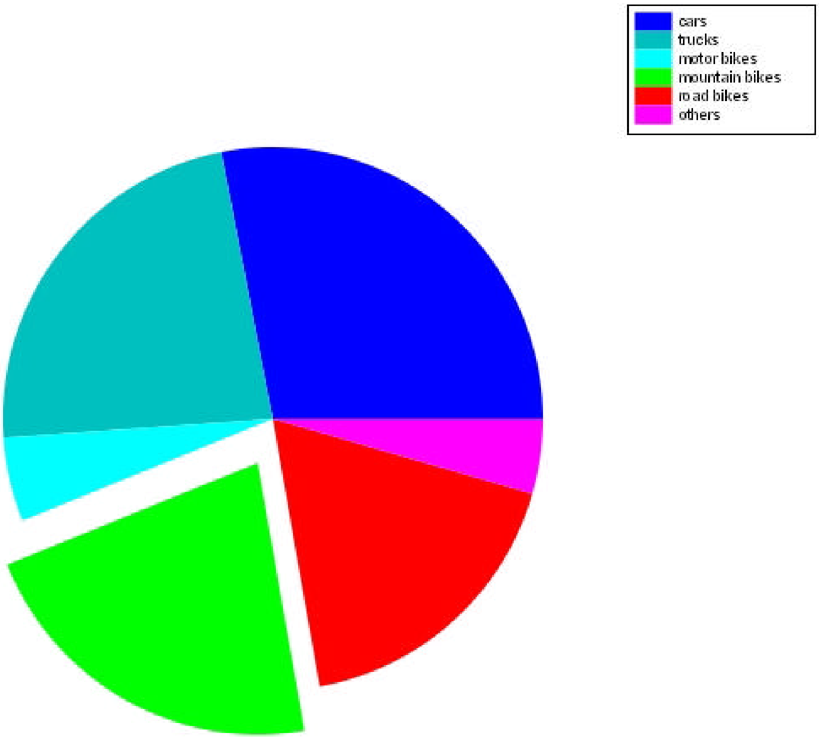
Each pie sector is defined by triplets:
- sector id
- legend label
- value

The different series are grouped in a container.

It is possible to create 2D or 3D effects, to explode one or several sectors, display the labels or the values.

7.2 Example

The following example shows the usage of different transportation means (cars, trucks, motor bikes, mountain bikes, road bikes, others), the green mountain bike sector being outlined.



7.3 Sample Java Code

The *ValueSeriesContainer* class contains the six series.

The *ValueSerie* class contains the triplets id / label / value.

The *PieChart* class draws the graphic using the desired size (height and width).

```
// Create Container for Series
ValueSeriesContainer valueSeriesContainer = new ValueSeriesContainer();

// Add series
valueSeriesContainer.addValueSeries( new PieSeries( "S6", " cars",              110,   0 ) );
valueSeriesContainer.addValueSeries( new PieSeries( "S5", " trucks ",           90,    0 ) );
valueSeriesContainer.addValueSeries( new PieSeries( "S4", " motor bikes ",      20,    0 ) );
valueSeriesContainer.addValueSeries( new  PieSeries( "S3", " mountain bikes ",  85,    20  ) ); // explode this sector
valueSeriesContainer.addValueSeries( new PieSeries( "S2", " road bikes ",       70,    0 ) );
valueSeriesContainer.addValueSeries( new PieSeries( "S1", " others",            15,    0 ) );

// Define image size
int imageHeight = 500; int imageWidth = 530;

ExplodedPieChart pieChart = new ExplodedPieChart(imageWidth, imageHeight, valueSeriesContainer);

// Generate  JPEG  image
pieChart.drawJPEG("test.jpg");

// Generate  SVG image
pieChart.drawSVG ("test.svg");
```

7.4 Different formatting options for pie charts

```
// Switch to 3D

pieChart.set3Dprofondeur( 20 );

// Stretch the pie chart on the horizontal axis

pieChart.setCircular( false );
pieChart.setVerticalEllipse( false );

// Add value tags to each sector

pieChart.setPieWithTags( true );

// Set label tags instead of value tags (default TAG_VALUE)

pieChart.setTypeOfTag( ExplodedPieChart.TAG_LABEL );

// Set the minimum percent value to display tags (default 2)

pieChart.setMinPercentToDisplayTag( 5 );  // in the example the serie “other” will not be tagged

// Set the margin between sector and tag (default 1)

pieChart.setTagOnPieWidth( 10 );

// Set the location of the legend
```

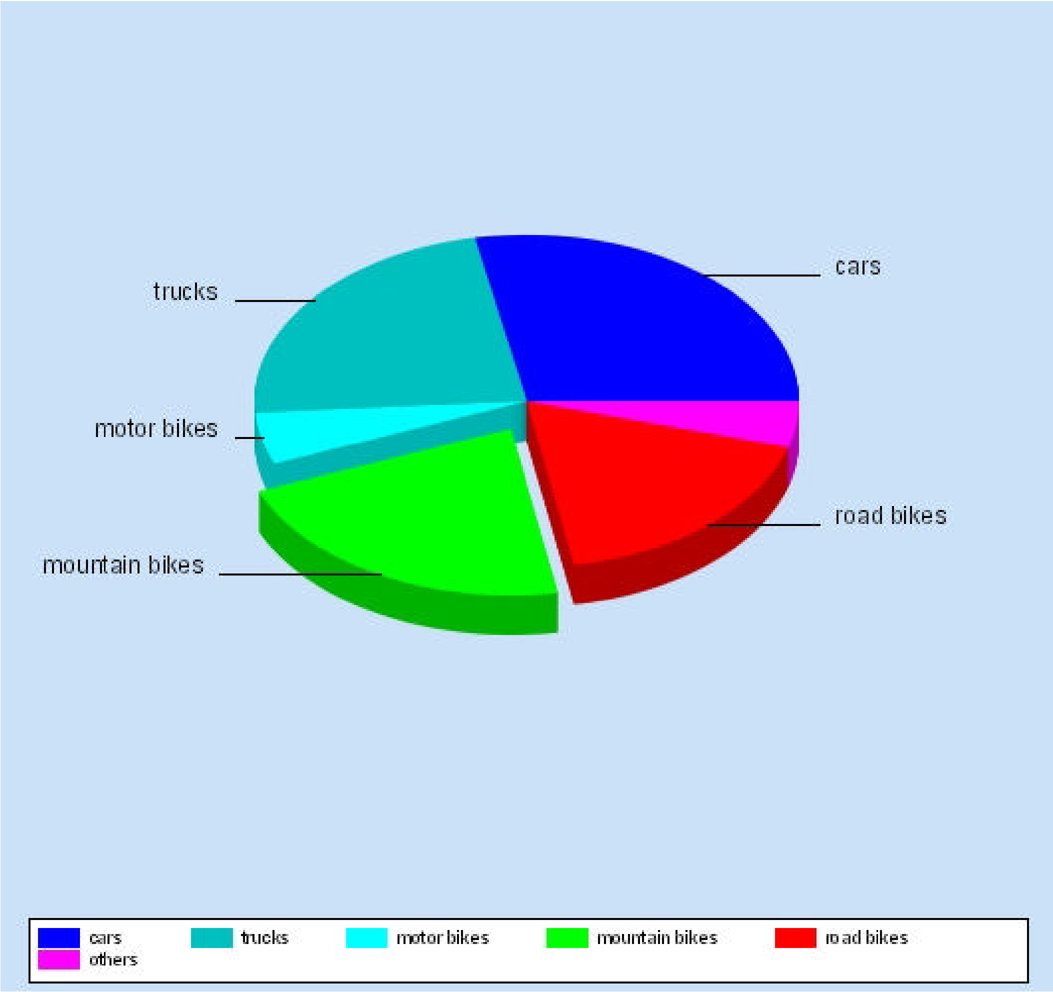
- **LEGEND_DISPLAY_MODE_BOTTOM**
- **LEGEND_DISPLAY_MODE_LEFT (default)**
- **LEGEND_DISPLAY_MODE_TOP**

```
pieChart.setLegendDisplayMode( Legend.LEGEND_DISPLAY_MODE_BOTTOM );


// Background colors

pieChart.setGraphicBackgroundColor( new Color( 202, 225, 247 ) );
pieChart.setImageBackgroundColor( new Color( 202, 225, 247 ) );
pieChart.setBackgroundColorVisible( true );
```

The result is the following image:



8. Chart Animation

8.1 Description

OFC-Charts supports animated graphics. The animation displays gradually the various items of the chart. More precisely, graphics which consist of different series can be animated and draw one serie after another.

Use the ***ChartAnimator*** class to create animated graphics as an SVG file.

Parameters such as animation duration and number of times to repeat can be set.

8.2 Sample Java Code

The following example creates the same Vertical Stacked Bar chart as in § 5.2. The duration of tha animation is set to 3 seconds and repeated indefinitely.

```
// Create the animated object
ChartAnimator animator = new ChartAnimator(verticalStackedBarChart);

// Set  duration in minutes and seconds
 int nbMinute = 0;
int nbSecond  = 3;

// Set the number of times to repeat
int nbRepeat =    SVGAnimationContext.INDEFINITE_REPEAT;

// Create the animated SVG file
animator.drawSVG(new FileOutputStream("testAnimateVerticalStackedBarChart.svg"),  nbMinute,  nbSecond, nbRepeat);
```

It is possible to animate « VerticalBarChart », «VerticalLineChart », «VerticalLine Plot », « PieChart ».

9. Contour 2D

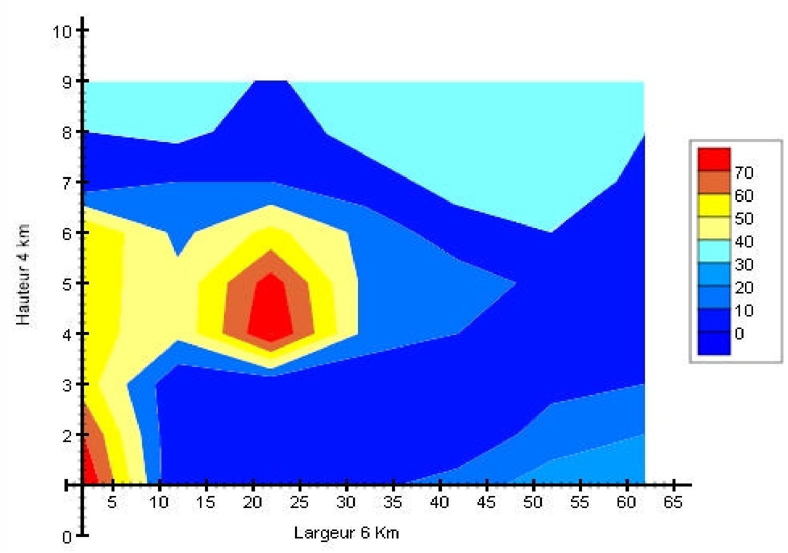
9.1 Description

The ***Contour2D*** class generates isolines from a grid. For example the grid could be a topographic map where isolines connect all points that have the same elevation.

9.2 Example

In the following example the grid is defined in a file contour2dTest.txt.

The result is:



9.3 Sample Java Code

```
// Image dimension
int imageHeight = 400;  int imageWidth = 500;

// Define horizontal axis
HorizontalNumberAxis axisX = new HorizontalNumberAxis("Largeur 6 Km");
axisX.setLabelPosition(HorizontalNumberAxis.LABEL_POSITION_HORIZONTAL_RIGHT_CENTER); axisX.disableArrow ();

// Define vertical axis
VerticalNumberAxis axisY = new VerticalNumberAxis("Hauteur 4 km");
axisY.setLabelPosition(VerticalNumberAxis.LABEL_POSITION_VERTICAL_LEFT_CENTER);
axisY.disableArrow();

// Set  step, start and end values
double _step = 10;   double _start = 70;  double _end = 0;
```

```
try {
        // Create 2D Contour object from data in contour2dTest.txt
        Contour2D c2d = new Contour2D(imageWidth, imageHeight, axisX, axisY,
                                        HaGrid.getGrid("contour2dTest.txt"),  _start, _end,  _step);
        // Create the JPEG file
        c2d.drawJPEG("contour2D.jpg");
} catch (Exception ex) {      ex.printStackTrace();    }
```

9.4 Data

The data used to define the graphic can be obtained from a file or computed directly in the application.

```
        IrregularGrid grid = new IrregularGrid(); // ou RegularGrid grid = new RegularGrid();
        Vector allLine = new Vector();
        Vector allGridNodeOfLine = new Vector();
// new GridNode(x, y, z)  ; for y = 0.5
// Add a node
        allGridNodeOfLine.addElement(new GridNode(10, 0.5, -73.3));
        allGridNodeOfLine.addElement(new GridNode(30, 0.5, -74.8));
        allGridNodeOfLine.addElement(new GridNode(50, 0.5, -75.1));
        allGridNodeOfLine.addElement(new GridNode(70, 0.5, -74.8));
        allGridNodeOfLine.addElement(new GridNode(90, 0.5, -73.3));
// Construct  an isoline
        allLine.addElement(allGridNodeOfLine);
// new GridNode(x, y, z)  ; for y = 1.5
        allGridNodeOfLine = new Vector();
        allGridNodeOfLine.addElement(new GridNode(10, 1.5, -73.4));
        allGridNodeOfLine.addElement(new GridNode(30, 1.5, -74.9));
        allGridNodeOfLine.addElement(new GridNode(50, 1.5, -75.2));
        allGridNodeOfLine.addElement(new GridNode(70, 1.5, -74.9));
        allGridNodeOfLine.addElement(new GridNode(90, 1.5, -73.4));
        allLine.addElement(allGridNodeOfLine);
// new GridNode(x, y, z)  ; for y = 2.5
        allGridNodeOfLine = new Vector();
        allGridNodeOfLine.addElement(new GridNode(10, 2.5, -73.5));
        …
// Construct the complete grid
        for(int i=allLine.size()  - 1; i >= 0; i--){
                grid.addLine((Vector)allLine.elementAt(i));
        }
```

In this example the file structure is:

First line = X values

Second line = Y values

The `HaGrid` class reads from this file and assigns x, y and z values:

```
        IrregularGrid grid = new IrregularGrid();
        Vector allLine = new Vector();
        for(int i= 0; i<nbLine; i++){
          Vector allGridNodeOfLine = new Vector();
          Vector lineData = (Vector)_temp.elementAt(i);
          double y = ((Double)_y.elementAt(i)).doubleValue();;
          for(int j=0; j< nbColumn; j++){
            double x = ((Double)_x.elementAt(j)).doubleValue();;
            double z = ((Double)lineData.elementAt(j)).doubleValue();
// Add one node
            allGridNodeOfLine.addElement(new GridNode(x, y, z));
          }
// Construct  an isoline
          allLine.addElement(allGridNodeOfLine);
        }
// Construct the grid
      for(int i=allLine.size() - 1; i >= 0; i--){
              grid.addLine((Vector)allLine.elementAt(i));
      }
// Construction de la grille complète des données
      allGridNodeOfLine.addElement(new GridNode(x, y, z));
```

9.5 Animation

Animating *Contour2D* graphics can prove extremely valuable in order to display a given phenomenon over time.

Using *Contour2DAnimator* class makes it easy. Just provide a set of SVG files in a sequence that follows the time flow and an animated SVG will be created.

10. Animation

10.1 Different possibilities

OFC-Charts allows the creation of animated SVG files.

Several possibilities are offered:

- Animations with predefined embedded images.
- Real time animations.

10.2 Predefined animations

Predefined animations are sequences of images embedded into an SVG file.

Several mechanisms are offered:

- Animation of series which are progressively made visible using the ***ChartAnimator*** class
- Animation of several *Contour2D* files which are played one after the other using the ***Contour2DAnimator*** class
- Animation of several charts played one after the other using the ***MultiChartAnimator*** class

In order to create predefined animations:

- Add images using the *add* method with one of the 3 classes hereabove
- Call *drawSVG* to create the SVG file with a given duration and a number of repetitions

Example :

```
int minute  =  0;
int second =  3;
int nbOfrepeat =  SVGAnimationContext.INDEFINITE_REPEAT;

// ChartAnimator
ChartAnimator animator = newChartAnimator(combinedVerticalBarChartXY);
animator.drawSVG ( new FileOutputStream("testCombinedVerticalBarChartXY.svg"),
                                minute,
                                second,
                                nbOfrepeat);
```

```
// Contour2DAnimator
 Contour2DAnimator animator = new Contour2DAnimator();
 double zStep = 5;
 double zSart = -90;
 double zEnd = 0;

 for (int i=0  ; i < 9 ; i++){
            animator.add( new Contour2D (imageWidth, imageHeight, axisX, axisY,
                                 HaGrid.getGrid ("courbe" + i + ".txt"), zStart, zEnd, zStep));
  }
  animator.drawSVG ( new FileOutputStream ("Contour2DAnimate.svg"),
                                minute,
                                second,
                                SVGAnimationContext.INDEFINITE_REPEAT);
```

10.3 Real time animation

The whole graphic or a part of it can be animated from an external source. With *OFC-Charts* you can specify what part of the graphic is dynamic and updated via an external URL. This part will get updated as time goes by thus allowing real time fluid animations from any data source.

Real time animation requires a *servlet* with two actions:

- The first action of the *servlet* creates the static part of the graphic and the animation context (refresh time and dynamic data source URL). The code for the static part can be any of the previous examples, e.g. the *VerticalBarChart* example with which only the axis and background information could be displayed while other parts are hidden.

- The second action of the *servlet* is in charge of the dynamic part. The code again can be any of the previous examples, e.g. the *VerticalBarChart* example where only the bars could be displayed.