Carpet Plots in Parametric Trade Studies: Development of a Matlab Tool to Create Carpet Plots

Matthias Oberhauser
Airbus Group Innovations

Lehrstuhl für Luftfahrtsysteme
Technische Universität München

Carpet Plots in Parametric Trade Studies: Development of a Matlab Tool to Create Carpet Plots

Matthias Oberhauser
LS-SA 12/24

LS-SA/DA lfd Nr. LS-SA 12/24	Vertraulichkeit Öffentlich
Titel (und Untertitel) Carpet Plots in Parametric Trade Studies: Development of a Matlab Tool to Create Carpet Plots	Autor Matthias Oberhauser Betreuer 1 Sky Sartorius Technische Universität München Betreuer 2

Daten zur Abgabe und Korrektur der Studienarbeit:

2013-03-28	2013-04-05	2013-05-05
Abgabe der Erstfassung (Datum)	Rückgabe der korrigierten Erstfassung (Datum)	Abgabe der Endversion (Datum)

Bestätigung der Einreichung der abgeschlossenen Studienarbeit durch den Betreuer:

Ort, Datum	Unterschrift des Betreuers am LLS (Sky Sartorius)

Anmerkungen:

Technische Universität München
Fakultät für Maschinenwesen

WS 2012/2013

Carpet Plots in Parametric Trade Studies

Development of a Matlab Tool to Create Carpet Plots

Themensteller:

M.Sc. Sky Sartorius
Fakultät für Maschinenwesen
Technische Universität München
Arcisstraße 11, 80333 München

Bearbeitet von:

B.Sc. Matthias Oberhauser
Badgasteiner Straße 6a
81373 München
0177 52 89 560
Matrikelnummer: 03633836 , Fachsemester: 4

Abgabetermin: 05.05.2013

Abstract

Visualizing information is an important skill for engineers. It is not only needed in order to understand complex sets of data, it is also necessary to communicate design decisions to non-technical staff. The carpet plot method is a visualization technique well-known in aeronautical engineering. It helps transforming complex multi-variable datasets into an easy to use two dimensional illustration. Despite its advantages, the method is only used seldom, as it requires some algebra to create such plots and so fare there is no free tool available to support the user in the creation process.

In this thesis the advantages and restrictions as well as the mathematical background of the carpet plot method will be discussed. In order to help engineers create carpet plots, a Matlab class has been developed.

To test the capabilities of the developed carpet plot tool, a simple model of a short-haul twin engine airliner has been created. Using this model as the baseline design, multiple trade studies have been conducted to demonstrate features of the created tool.

The class can be downloaded at Mathwork's file exchange platform including a documentation and the source code at `http://www.mathworks.com/matlabcentral/fileexchange/41467-the-carpetplot-class`

Table of Contents

Nomenclature

$S_{Landing}$	Landing distance
α	Thrust lapse ratio $\frac{T}{T_{To}}$
β	Weight fraction $\frac{W}{W_{To}}$
λ	Taper ratio
σ	Density ratio
Λ	Sweep angle
a_{max}	Maximum input value of parameter a
a_{min}	Minimum input value of parameter a
b_{max}	Maximum input value of parameter b
b_{min}	Minimum input value of parameter b
C_{D0}	Zero lift drag
$C_{L_{max}}$	Maximum lift coefficient
$C_{L_{TO}}$	Lift coefficient in takeoff configuration
h	Altitude
i_a	Interval of the a values
i_b	Interval of the b values
K_0	Independent factor for x values in a cheater plot
K_1	Factor for x values in a cheater plot dependent on parameter a
k_1	Induced drag factor
K_2	Factor for x values in a cheater plot dependent on parameter b
n	Load factor
n_a	Number of input data elements of parameter a
n_b	Number of input data elements of parameter b
q	Dynamic pressure
R^2	Coefficient of determination
S_a	Obstacle clearance distance

t/c	Thickness ratio
T/W	Thrust-to-weight ratio
V	Velocity
W/S	Wing loading
W_{To}	Takeoff weight
CAx	Computer Aided technologies
DOC	Direct Operational Costs
FAR	Federal Aviation Regulations
JAR	Joint Aviation Requirements
MTOW	Maximum Takeoff Weight
OEW	Operational Empty Weight
Pax	Passengers
UML	Unified Modeling Language
WYSIWYG	What You See Is What You Get

List of Figures

List of Tables

1. Introduction

1.1. Overview

Engineers have to make design decisions based on empirical or theoretical data. To make the right choices, it is important for an engineer to fully understand the set of data and the influences and dependencies on it. Visualizing data is mandatory when it comes to engineering design decisions. These visualizations should not only visualize the data; they should also communicate information in a technically accurate way (Vitaly Friedman 2008). As design choices often have to be backed by decision makers or are part of a bidding process, it is also important to *sell* the information in a clear and easy to understand way. This can be a hard task especially when it comes to multi-dimensional data and multiple dependencies.
This thesis will focus on a well-known method for visualizing multi-variable data: The carpet plot. This technique is common in aircraft design studies. Yet, as there is no widely spread free tool available for creating carpet plots, it takes some effort to create individual plots. Therefore, many engineers do not use this technique so far.
The main goal of this thesis is to create an easy to use but powerful carpet plot tool that will support engineers in presenting their data visibly and clearly.
After some basics about visualization in chapter 1 of this thesis, section 2 will give details about the creation of carpet plots. The development of the carpet plot tool will be presented in chapter 3, including in particular a discussion of the requirements, implementation and features. In part 4 of this thesis, some trade studies on an aircraft design will show how the tool can be used in practice.
The well documented tool will be published under a free software license. Thus, this thesis might help spreading the carpet plot technique.

1.2. Trade Studies in Aircraft Design

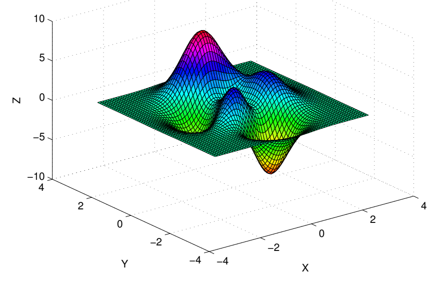
An aircraft model consists of a huge number of different design variables. Thus the aircraft design process is very complex and highly iterative. Even at the end of an aircraft design process in which proven methods have been used it is not certain that one will end up with an aircraft that meets all requirements or even the *optimal* vehicle.
In order to find more and possibly better configurations, it is necessary to explore the solution space. This is done by using trade studies. As a prerequisite, good knowledge of the aircraft design and a detailed aircraft model are required. When changing some parameters (e.g. T/W, W/S, aspect ratio) the impacts of these changes on other parameters can be validated. The results will help the designer understand the trade-offs and apply useful modifications to the baseline design if necessary.
It is also important to be able to *sell* the results to decision makers. For this purpose a good and clear visualization is mandatory. Yet, this can be a tough task as a trade study can consist of four or even five variables, so the effort in creating useful visualizations should not be underestimated.

1.3. Visualization of Multiple Variable Data

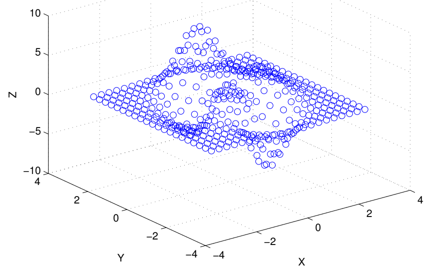
Sometimes trade studies require navigating through a three, sometimes four or even five dimensional solution space.

$$f(x, y) = z \tag{1.1}$$

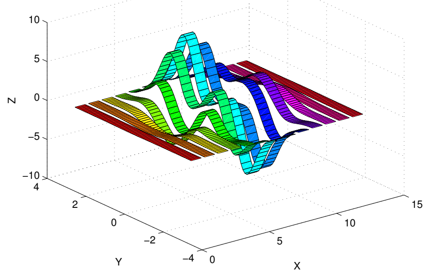
Visualizing a function like equation 1.1 with two variables leads to a three dimensional plot. A surface plot like in Fig. 1a is the most obvious presentation when it comes to visualizing a two variable function. But also mesh plots, scattered points or ribbon plots as shown in Fig. 1b and Fig. 1c are options, only to name some of many.



(a) Surface plot



(b) Scatter plot



(c) Ribbon plot

Fig. 1.1: Three dimensional plots

A common issue of all three dimensional plots is their lack of usability when it comes to fast interpolation or *paper-based* use. Although these three dimensional figures help imagine the data structure, engineers can hardly extract data points without the use of CAx systems. To make three dimensional plots more usable, the method of choice is projecting it to a two dimensional plain. A method well-known from cartography is the contour plot, shown in Fig. 2. Contour plots do a good job by connecting points with same z values with isolines. This results in an easy to read two dimensional illustration.

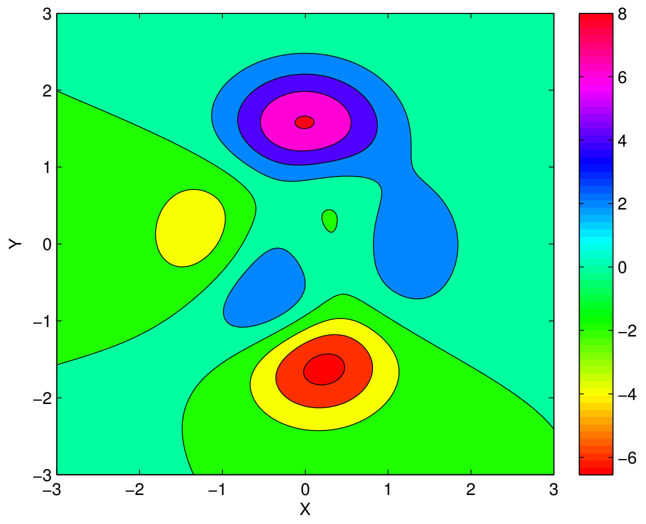


Fig. 1.2: A filled contour plot

Another way of visualizing three dimensional data is a parametric plot. Creating a parametric plot out of a simple surface like in Fig. 3, is achieved by cutting the surface at several positions in one axis. The projected intersection curves form an easy to read two dimensional plot.

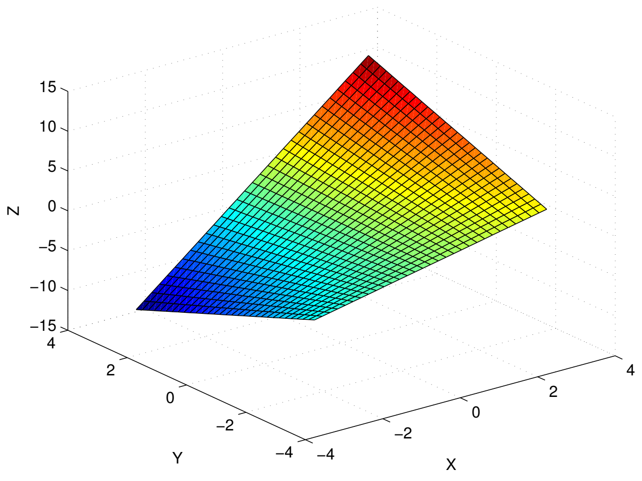
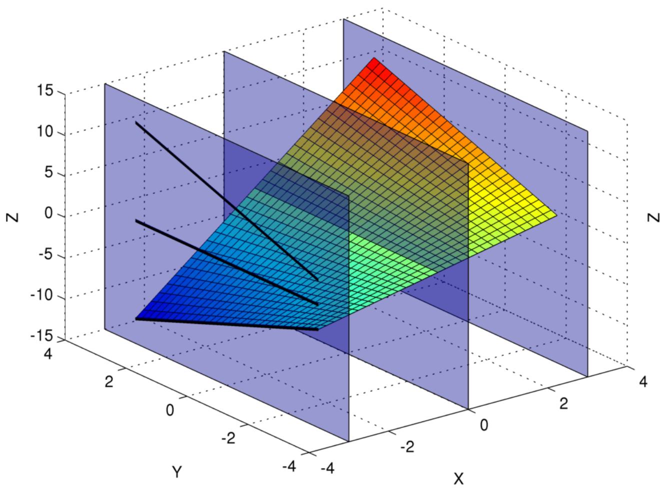


Fig. 1.3: A three dimensional surface plot

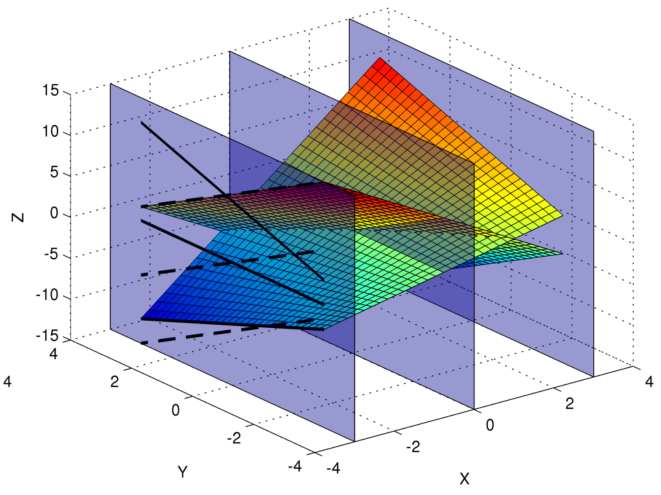
If one adds another independent function to the plot, like equation 1.2, it is also possible to visualize the additional equation to a parametric plot. This is done by projecting the second surface's intersection curves to the plot.

$$
\begin{aligned}
f(x, y) &= z_1 \\
f(x, y) &= z_2
\end{aligned}
\tag{1.2}
$$

This can be repeated with any other independent function as shown in Fig. 4a and Fig. 4b. The result is a very simple two dimensional plot that offers a high level of information density as presented in Fig. 5.



(a) One function



(b) Two functions

Fig. 1.4: Creation of a parameterized two dimensional plot from a three dimensional surface

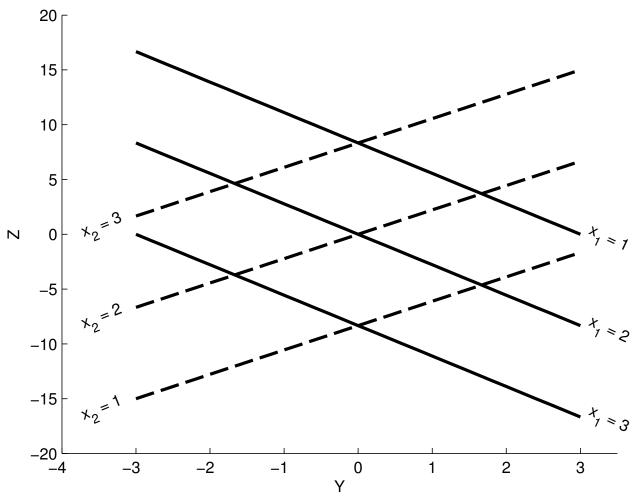


Fig. 1.5: Two dimensional parametric plot

As this introduction has outlined, parametric plots have various advantages. Among them, they are a well-known method and accompany most engineers throughout their careers. They are easy to create and use. However, they have their downsides when it comes to fast interpolation: they get crowded when another function is added and the three dimensional impression of the surface is lost.

2. The Carpet Plot

2.1. About the Carpet Plot Method

Carpet plots illustrate the behavior of two independent variables and their impact on each other in a compact, easy to read plot. The method makes it easy to interpolate intermediate values for both parameters (IHS ESDU 2004a, p. 1). Although it is a two dimensional plot, a three dimensional impression of the data can be achieved.

2.2. Construction of Four Variable Carpet Plots

In order to construct a carpet plot some algebra is needed. This section describes the theoretical background and the necessary steps to manually create a carpet plot. The four variable carpet plots, or real carpet plots, need two independent functions which share one variable and have at least one unique parameter each (Schneider 2000, p. 2).

$$\begin{aligned} f(x,a) &= y_1 \\ f(x,b) &= y_2 \end{aligned} \tag{2.1}$$

The carpet plot consists of the intersections of these two functions. The y variables of the two functions 2.1 have to be equated to find the intersections.

$$f(x,a) = f(x,b) \tag{2.2}$$

After solving equation 2.2 to x and inserting the solution into one of the equations in 2.1 two functions can be found that are dependent of both a and b. In most cases, equation 2.3 is already available in the first place and the step described above is not necessary. The equations of 2.3 can now be used as the trade parameters and can be assigned to a finite set of data.

$$\begin{aligned} x &= f(a,b) \\ y &= f(a,b) \end{aligned} \tag{2.3}$$

The following example shows two simple linear functions that will be transformed into a carpet plot using simple algebraic and some spreadsheet calculations. A parametric plot of the two functions in equation 2.4 is illustrated in Fig. 6.

$$\begin{aligned} y_1 &= a + x \\ y_2 &= \frac{x}{b} \end{aligned} \tag{2.4}$$

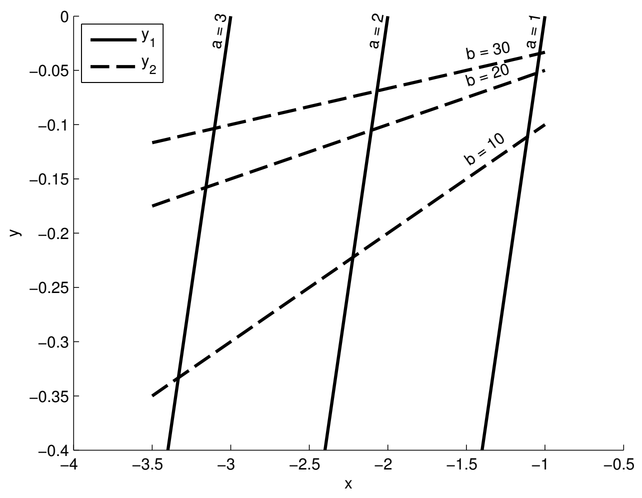


Fig. 2.6: Parametric plot of the two sample functions

After equating the two functions, the base form for creating the carpet plot can be found as seen in 2.5.

$$\begin{aligned} a + x &= \frac{x}{b} \\ x &= \frac{ab}{1-b} \\ y &= \frac{a}{1-b} \end{aligned} \tag{2.5}$$

Consequently, a table can be created covering the a and b values as well as the corresponding x and y values. Table 2.1 shows the values for a simple nine point carpet plot.

Nr.	**a**	**b**	**x**	**y**
1	1	10	-1.11	-0.11
2	1	20	-1.05	-0.05
3	1	30	-1.03	-0.03
4	2	10	-2.22	-0.22
5	2	20	-2.11	-0.11
6	2	30	-2.07	-0.07
7	3	10	-3.33	-0.33
8	3	20	-3.16	-0.16
9	3	30	-3.10	-0.10

Table 2.1: Values for a simple carpet plot

In the first step the rows 1 to 3, 4 to 6 and 7 to 9 have to be plotted. They are equivalent to the parameter a from 1 to 3. See Fig. 7.

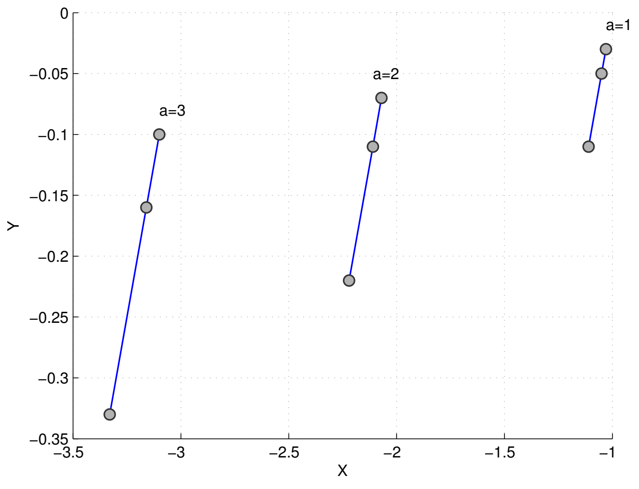


Fig. 2.7: The plotted a values of a carpet plot

Now all the data points with the same b values have to be connected. This can be done by plotting the rows 1, 4, 7 and 2, 5, 8 as well as 3, 6, 9. The result is the simple nine point carpet plot of Fig. 8

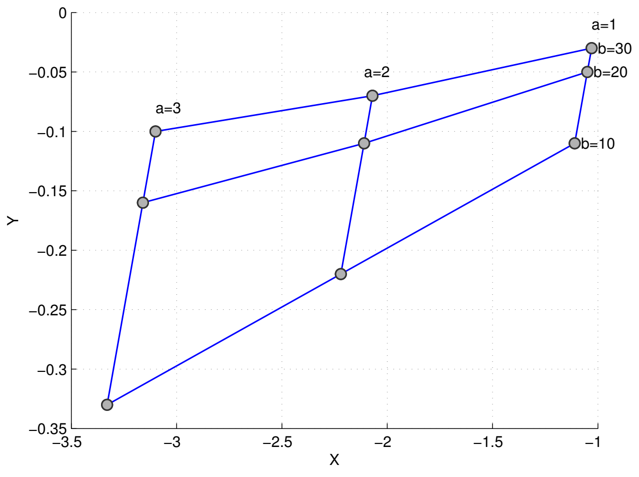


Fig. 2.8: A simple carpet plot

This is a simple linear example. The math will get more complex if there is more than one intersection. In that case a sanity check must be conducted to define the intersections in the desired data range and choose or dismiss solutions.

2.3. The Cheater Plot

If there is only one function with two variables, the prerequisites for a carpet plot discussed in chapter 2.2. are not met. Anyway, if a function has the form of equation 2.6, it is possible to create a carpet plot, a so-called cheater plot.

$$y = f(a, b) \tag{2.6}$$

As cheater plots do not have values for the x axis the x values have to be shifted to get a two dimensional plot. This can be done using equation 2.7.

$$x = K_0 + K_1 a + K_2 b \tag{2.7}$$

If the a and b values are equally spaced and K_1 and K_2 are calculated using equation 2.8, the intersections will line up vertically. This eases working with the plot. Note that it is common not to label the x axis (Powers 1997, p. 2).

$$\begin{aligned} i_a &= \frac{a_{max} - a_{min}}{n_a} \\ i_b &= \frac{b_{max} - b_{min}}{n_b} \\ K_1 &= \pm 1 \\ K_2 &= \pm \frac{i_a}{i_b} \end{aligned} \tag{2.8}$$

Fig. 9 shows a cheater plot with the K_1 and K_2 factors that were chosen in a way that the intersections would line up vertically. The arrows on the top left indicate the shifting of the x axis (IHS ESDU 1996).

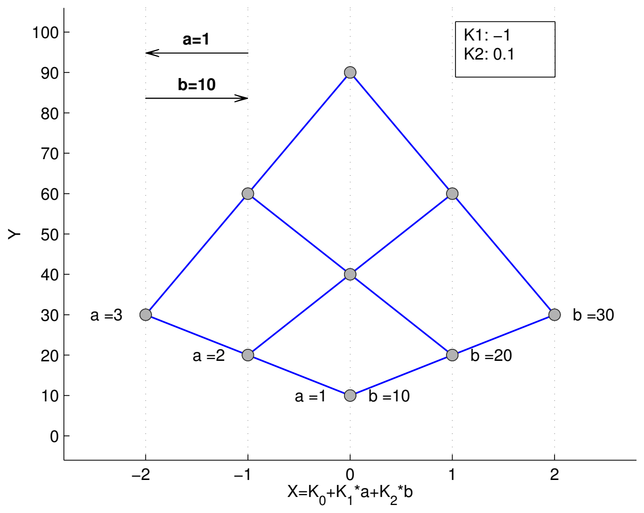
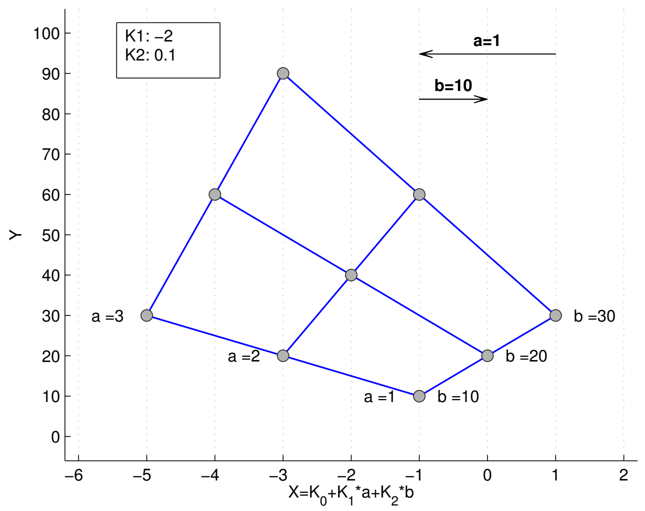
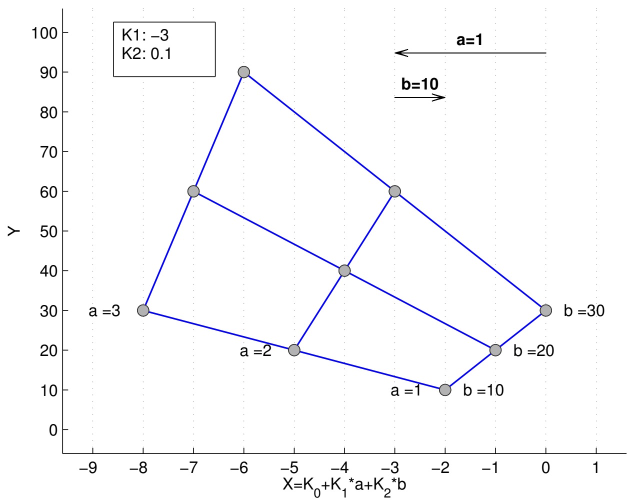


Fig. 2.9: A cheater plot

If the input data is not equally spaced or a vertical alignment is not desired, it is possible to change the appearance of the plot by changing the values of K_1 and K_2. By changing $K1$, the shifting (in x direction) of the a data points increases. This effect can be observed in Fig. 10.



(a) K_1 = -2



(b) K_1 = -3

Fig. 2.10: Effect on a cheater plot when changing the K_1 value

2.4. Folded Carpet Plots

The representation of carpet plots highly depends on the given input data. Some data produce hardly readable carpet plots which can be turned into a clearer visualization using some tweaks that will be explained in the following section. Some other data, as seen in Fig. 11, cannot be visualized as a carpet plot because data points are overlapping multiple times.

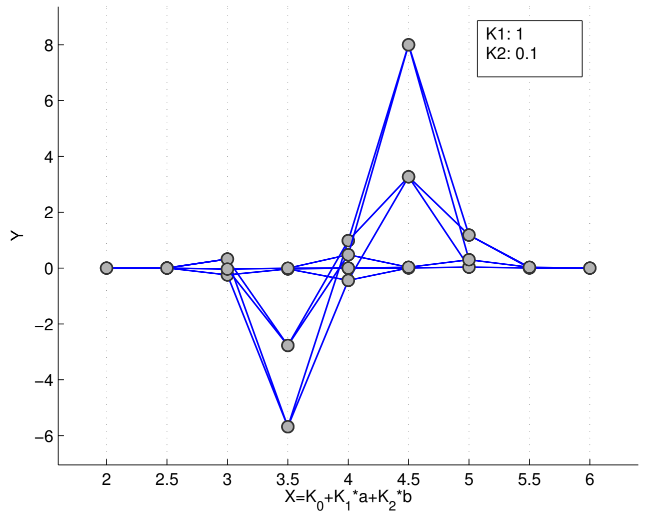
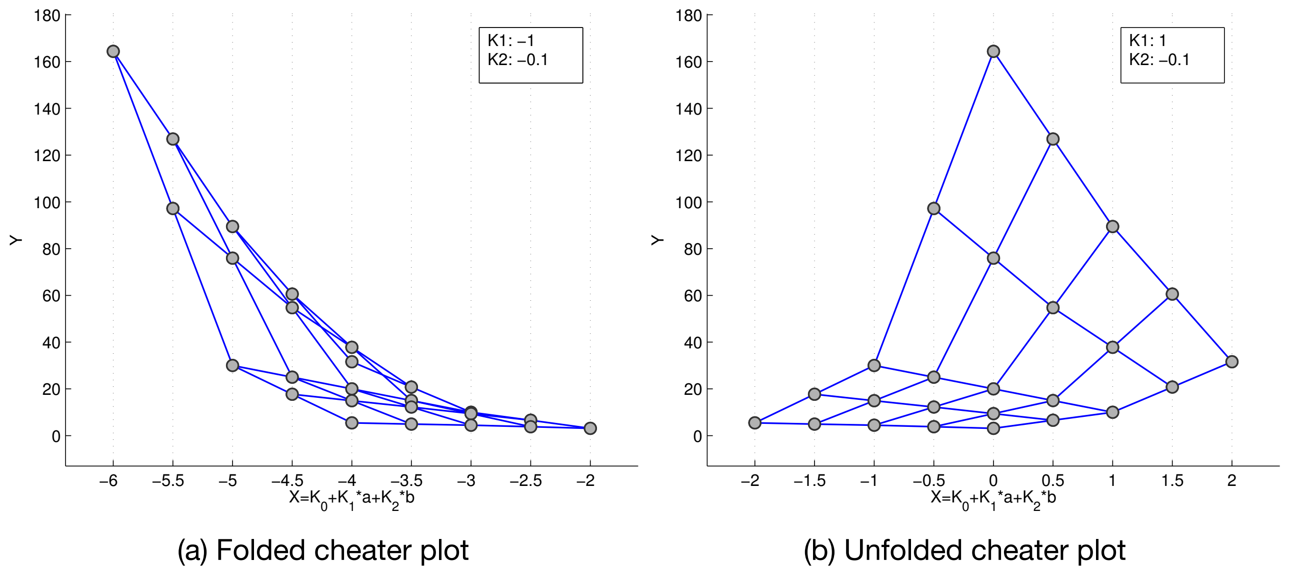


Fig. 2.11: Matlab's peaks function as a cheater plot

As the cheater plot's x axis is not fixed, it is possible to unfold the data. This is done by changing the K_1 and K_2 factors, ergo changing the plotting direction. If these factors are calculated using equation 2.6, the algebraic sign can be varied until a good, ideally unfolded representation is reached (IHS ESDU 2004a, p. 6).



(a) Folded cheater plot

(b) Unfolded cheater plot

Fig. 2.12: Algebraic sign's effect on K_1 and K_2

In the case of plots with four variables, also known as real carpet plots, the plotting direction is fixed and therefore unfolding a folded plot, as shown in Fig. 13, is not possible.

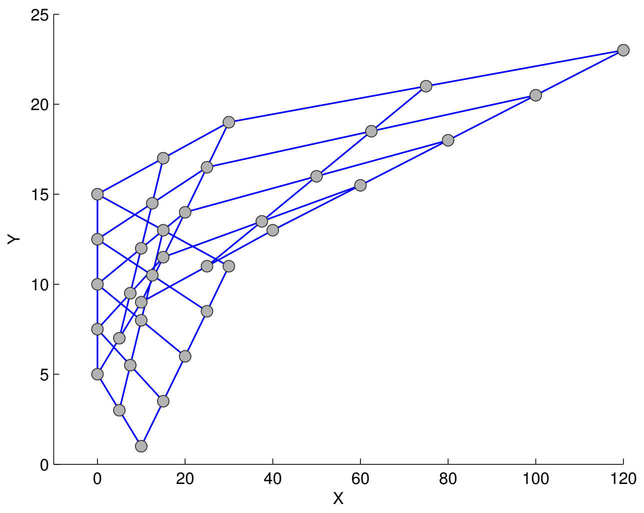
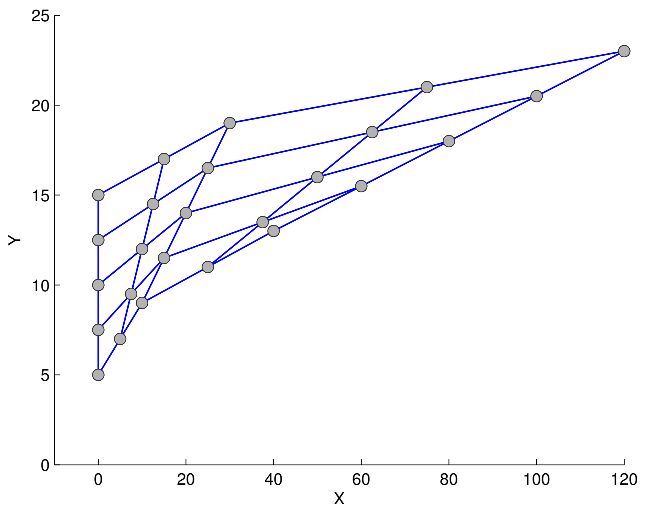
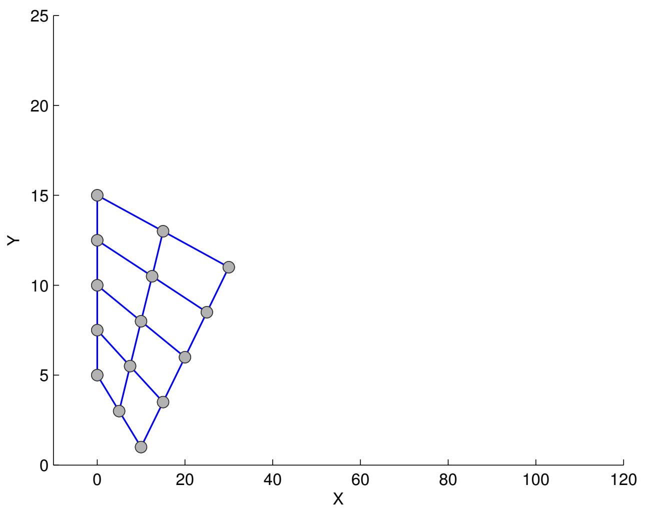


Fig. 2.13: A folded carpet plot

One way of making the plot readable is to plot the folded parts separately like in figure Fig. 14.



(a) First part of the unfolded carpet plot



(b) Second part of the unfolded carpet plot

Fig. 2.14: Plotting parts of the carpet plot to unfold it

3. The Carpet Plot Tool

3.1. Intention

As seen in chapter 2.2., the construction of carpet plots requires some effort. For a productive work flow, it is necessary to automate the process. As of March 2013, there is no free and widely spread tool available to create and customize carpet plots. Personal Excel macros like Schneider's (Schneider 2000, p. 12) are one way but are far from a fast and flexible tool. In addition, those tools are mostly unpublished individual helpers.
Due to the lack of carpet plot tools, the barriers of using the technique, especially for users not familiar with programming, are rather high. This thesis should not only result in an easy to use but also in a powerful tool for creating carpet plots. Thus, this thesis may be able to contribute to spreading this technique.

3.2. Requirements

Software requirements can be summed up in two main categories, functional requirements and non-functional requirements (Eide 2005). The functional as well as the non-functional requirements were derivated from use cases, existing tools, published carpet plots as well as personal experience and deliberation. The Gasturb software (Joachim Kurzke 2013) offers the possibility to create parametric studies for gas turbines which can be visualized with a carpet plot and integrated filled contour plots. The visualization of the Gasturb software as well as numerous of the cited ESDU data items along with others were used as a measure for graphical elements and features.
In this process, two main use cases were identified that are illustrated using UML interaction diagrams. In Fig. 15 the first use case is shown. After preparing the input data, the user will change parameters and observe the effects of changes interactively. It also should be possible to change the plot using a WYSIWYG plot tool.

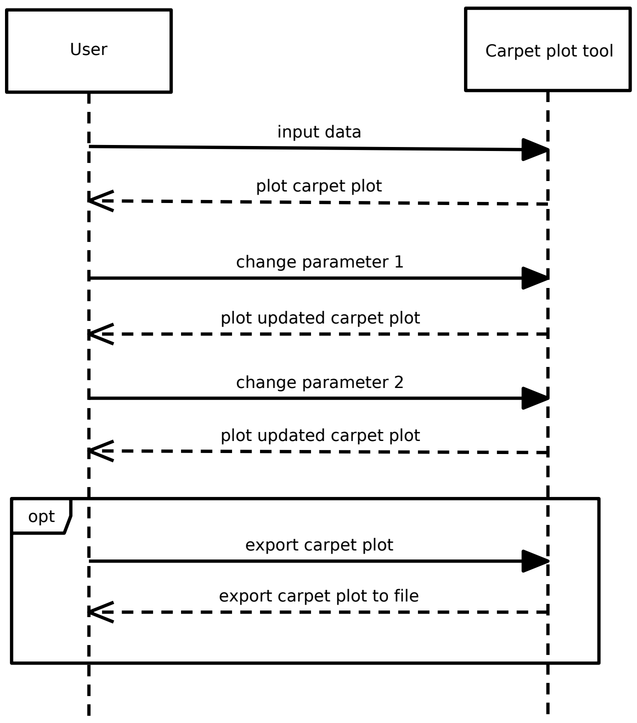


Fig. 3.15: UML interaction diagram for the first use case

The second identified use case, shown in Fig. 16, describes a user who writes a script in order to create the data and visualize it by use of the carpet plot tool. Changing the position of graphic elements manually using a WYSIWYG plot environment should not be necessary. The user then can save the script and should be able to create the same plot repeatedly. In this case, immediate visual feedback is not necessary and might even have a negative impact on the performance.

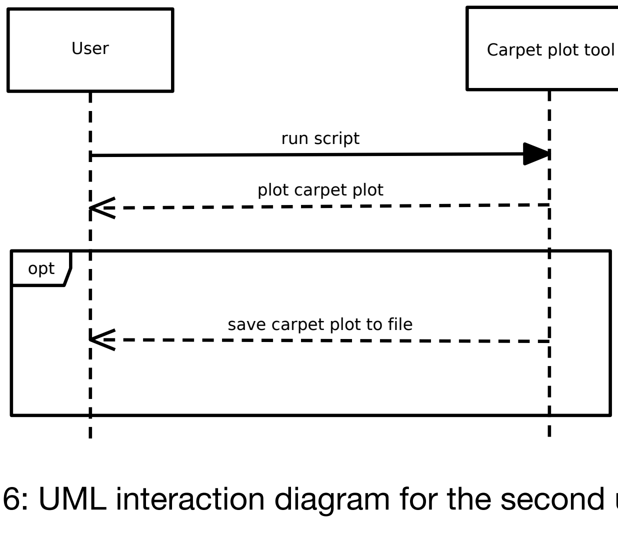


Fig. 3.16: UML interaction diagram for the second use case

In both use cases the export of the carpet plot as an image file is optional. In some cases, visual feedback is enough whereas in other cases publishing the carpet plot as an image file format is required.
Based on these identified use cases and other methods, table 3.1 shows the functional requirements for the carpet plot tool. The non-functional ones are listed in table 3.2.

Feature	Description	Priority
Carpet plot	Creating a four variable carpet plot from a set of input data points that cover the whole carpet.	must have
Cheater Plot	Creating a three variable cheater plot from a set of input data points that cover the complete carpet.	
Scattered input data	Support for creating a cheater and a carpet plot out of scattered data points.	
Labeling	Labeling of axes and plot lines.	
Customizability	The visualization of the carpet should be customizable manually as well as by a script or macro.	should have
Rotation of labels	The line labels should rotate according to the slope of the line.	
Spreadsheet support	Data from Microsoft Excel or similar spreadsheet applications should be importable.	
File export	The plot should be exportable to a common vector or raster graphics file format.	
Constraints	Constraints can be added to the carpet plot.	
Non numeric inputs	Data with non numeric data points like infinite values resulting from a division by zero should be ignored by the carpet plot tool.	
Custom intervals	The axis interval should be changeable.	
Cheater: point alignment	The intersections of the cheater plot should line up vertically. See chapter 2.3.	
Cheater: optimal x-axis	The k-factors should spread out the carpet as wide as possible. See chapter 2.3.	
Interpolation of points	Points in the carpet should be visualized with the corresponding a and b lines.	nice to have
Interpolation of plots	Interpolation of plots (IHS ESDU 2004b, p. 14).	
Filled contours	Filled contour lines to represent another variable dependent of a and b.	
Coordinate transformation	Any x/y-plot could be transformable to the carpet plot's coordinate system.	
Graphical user interface	An easy to use graphical user interface will help inexperienced users to work with the tool.	

Table 3.1: Functional requirements for the carpet plot tool

Feature	Description	Priority
Free software	The tool must be made public and the source code must be made accessible through an open source license.	must have
Matlab Implementation	The tool must be implemented in Mathworks Matlab.	
Documentation	The tool as well as the source code should be well documented to help users and developers.	should have
Usability	The use of the tool should be easy and intuitive.	
Performance	The tool should be able to handle a huge amount of data without memory issues and should create carpet plots in a reasonable time.	nice to have

Table 3.2: Non-functional requirements for the carpet plot tool

3.3. Implementation

The implementation is based on the requirements identified in chapter 3.2. The implementation in Matlab is one of the "must haves": Matlab is a powerful software package for numeric calculations. As the name MATrix LABoratory indicates, its strength is the handling of large vectors and matrices (Angermann, Beuschel, Rau & Wohlfarth 2005, p. 1). The tool offers a variety of extensions, so-called toolboxes. Its capabilities and the built in aerospace toolbox make it a popular tool among aeronautical engineers. Matlab is backed by an active community that offers support and a number of free scripts and extensions on `http://www.mathworks.com/matlabcentral/`.
Matlab offers different ways of extending its features:

- Script
- Function
- Class
- Toolbox

Scripts are files with Matlab code that can be executed. It does not provide the possibility to hand over input or get output parameters.
The function approach offers the possibility to create files containing a user defined function with one or more input and output parameters. The file can contain several sub-functions which cannot be called from outside the function.

With object-oriented programming, however, classes can be created that contain multiple methods and properties. During run-time, more than one instance of a class can be created. A toolbox is nothing more than a collection of multiple functions, scripts, or classes bundled in one folder. Many built-in and commercial toolboxes offer graphical user interfaces.
Because of the requirements identified in chapter 3.2., an object-oriented approach has been chosen. There are several reasons for that. Especially the need of high customizability makes it necessary to offer a lot of changeable properties. But as a huge number of input parameters clutter a function's input, this would have a negative influence on the usability. Not only will the class offer various properties, there is also a large number of methods. In the conventional approach, a new script file for every public accessible method would have to be introduced. This would lead to a number of different files containing scripts that could not communicate with each other. Thus, with the object-oriented approach, there is only one file containing all the methods and properties.
Fig. 17 illustrates the simple structure of the class. The carpet plot class contains all public methods to create and modify the plot. Child objects of the class are the two axes. They do not have any methods but all necessary properties to change the style of the lines and labels of the different axes.

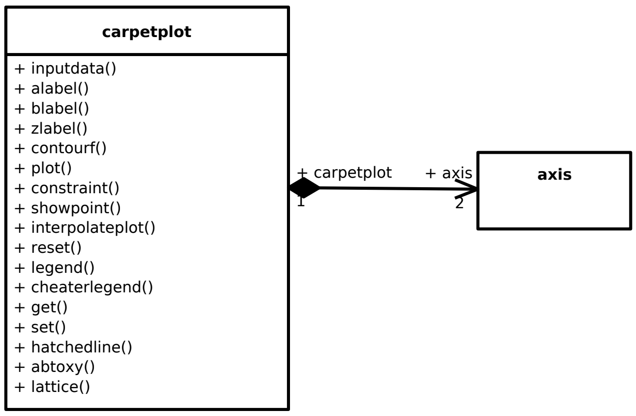


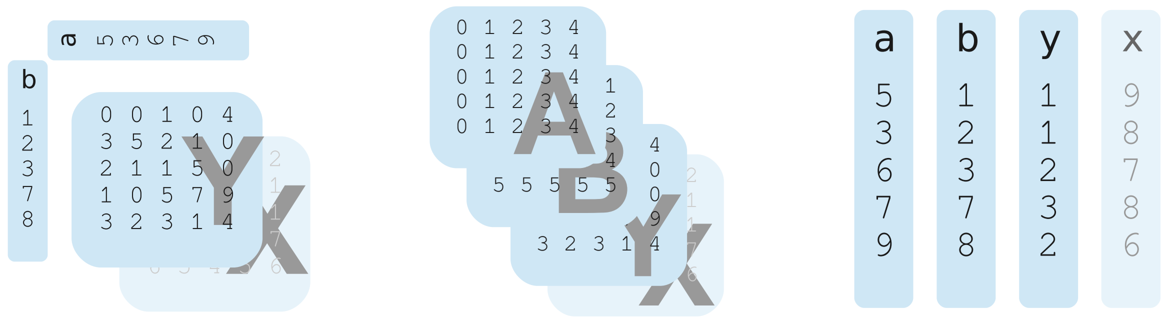
Fig. 3.17: UML class diagram of the carpet plot class without properties

Because of this, updating the carpet plot during run time is easy as long as the carpet plot instance is present in the workspace. Working with multiple plots also benefits from the object-oriented approach, as new instances of the carpet plot object can be created at any time.

3.4. Features

In the following section, the main features of the newly developed Matlab class will be demonstrated. As it is a flexible tool, the creation of more plots may be possible with the current implementation and the further development can continue by the author or the Matlab community.
In Fig. 18, the different forms of input data are illustrated. The exact input methods cover every data point of the carpet plot and can be plotted directly. On the other hand, the scattered data input relies on interpolation of a complete matrix before plotting. Both, the

scattered data input as well as the exact input, can be generated in Matlab or imported from a data file using the import feature of Matlab.



(a) Exact input with vectors (b) Exact input with matrices (c) Scattered input

Fig. 3.18: The possible data inputs for the Matlab carpet plot class

Fig. 19 shows a few lines of Matlab code that creates four matrices containing sample data for a carpet plot. All following examples will be based on this sample data.

```
% Create exact input data for a carpet plot
a = linspace(0,50,10);
b = linspace(1,50,12);
[A,B] = meshgrid(a,b);
X = A.^2.*B;
Y = A.^2-B.^2;
```

Fig. 3.19: Creation of input data in Matlab

If the input data is imported or created using Matlab, a simple carpet plot can be created in one line. This meets the requirement of simple usability described in chapter 3.2. Simply by calling the class constructor (Fig. 20), the carpet will be plotted. Saving the object in a variable is not necessary but important for further customization.

```
o = carpetplot(a,b,X,Y);
```

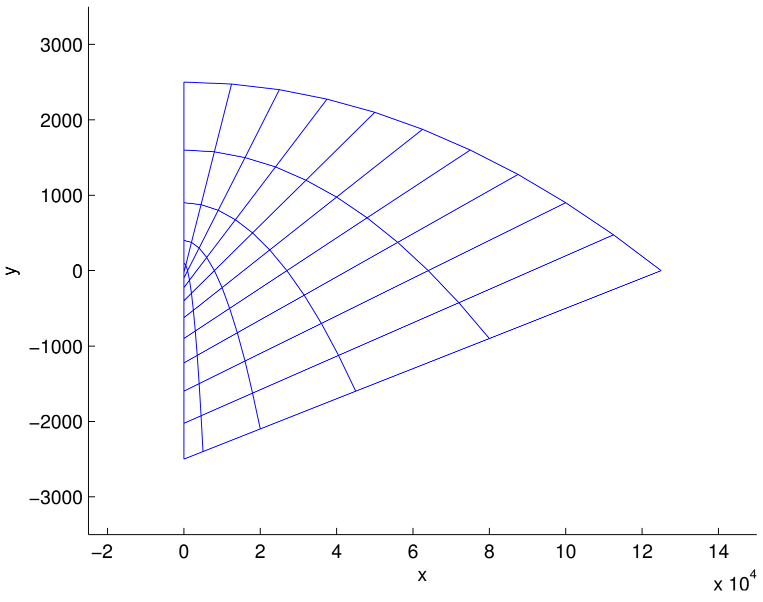


Fig. 3.20: A simple four variable carpet plot

By use of the same method, a cheater plot can be created. As the constructor supports a varying number of input arguments, this can be done by not using the x parameter. Fig. 21 shows the resulting cheater plot, the x axis has disappeared and the intersections line up vertically.

```
o = carpetplot(a,b,Y);
grid on;
```

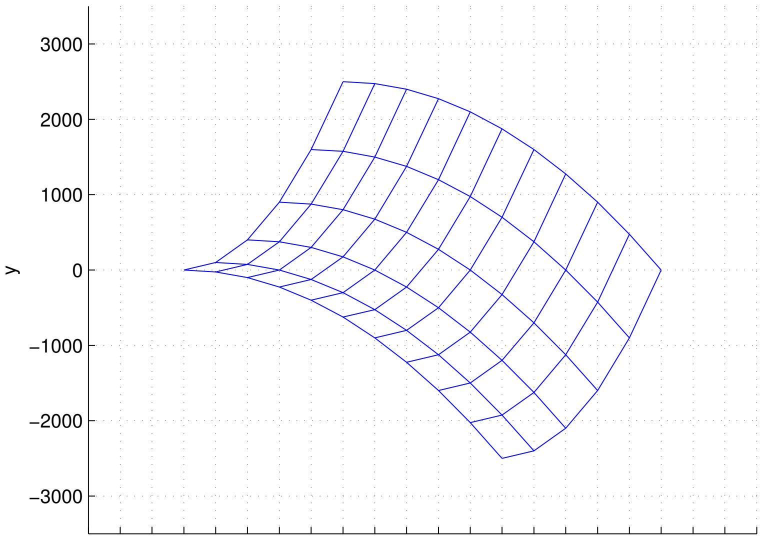


Fig. 3.21: A simple cheater plot

Another must have requirement, identified in table 3.1, is the labeling of the carpet plot. Fig. 22 shows how both axes can be labeled at once with one function call.

```
o = carpetplot(a,b,X,Y);
label(o,'A-Values',...
        'B-Values')
```

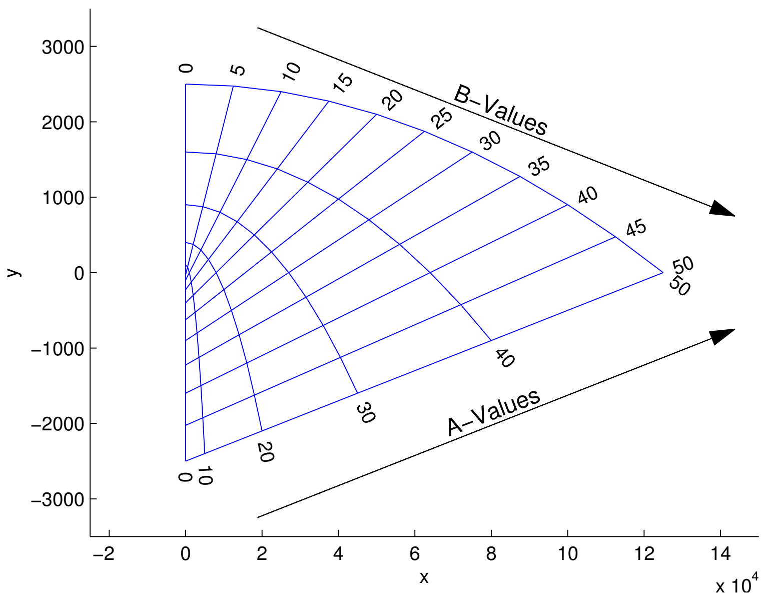


Fig. 3.22: A labeled carpet plot

To further change the appearance of the carpet plot, get and set methods have been implemented. These methods offer access to a variety of the plot's properties. Furthermore, the properties of every graphic object are accessible trough handles.

```
o = carpetplot(a,b,X,Y);
label(o,'A-Values','B-Values')
set(o,'bTick',[0:10:50],'blabelspacing',0.4);
set(get(o,'texthandles'),'color','b','FontWeight','bold')
```

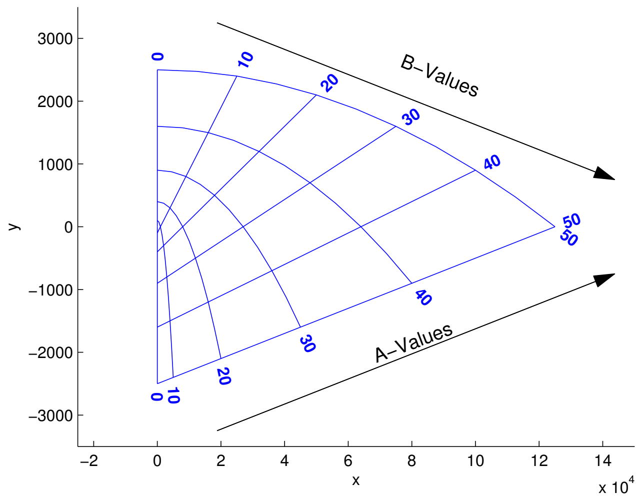
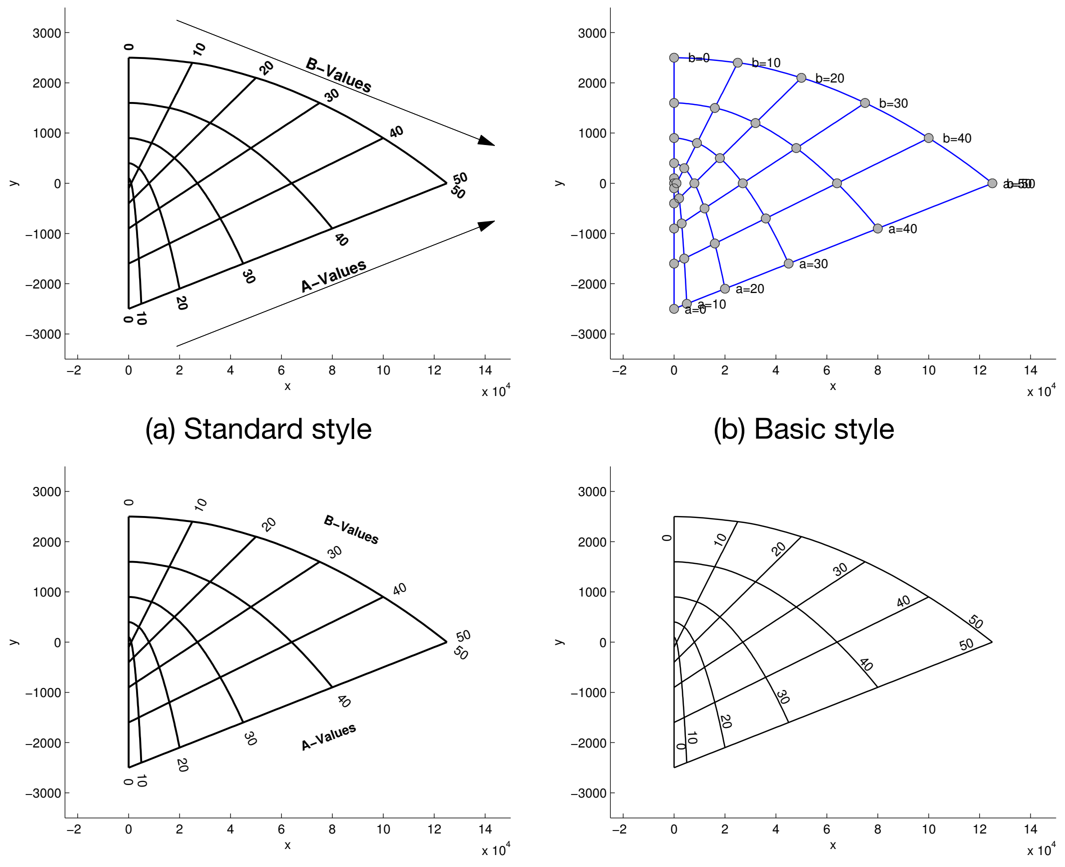


Fig. 3.23: A customized labeled carpet plot

For a more convenient change of the overall appearance of the plot, different styles (Fig. 24) have been implemented.



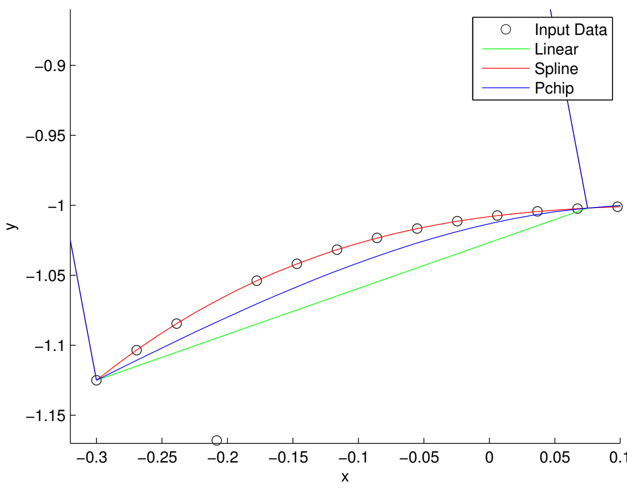
(a) Standard style (b) Basic style

(c) Clean style (d) Minimal style

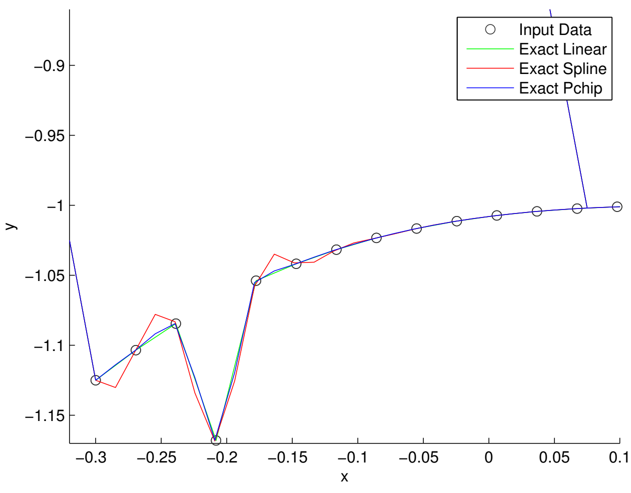
Fig. 3.24: Available styles for the carpet plot class

Another important requirement is the curve fitting which is needed in order to produce smooth lines. There are two major methods: The first one interpolates the line intersections using a linear table lookup algorithm. It then connects these intersections using one of three different curve fitting styles. The default style is ’linear.’ It connects the intersections with straight lines. This might result in a roughish presentation. The other two styles, ’Pchip’ and ’Spline’, use piecewise cubic interpolation or spline interpolation to connect the intersections with a smoother line.
If the intersections only represent a small part of the actual input data, the second major method can be used. In this case, all input data points will be considered for the lines. The intersections will not be interpolated, as the lines should intersect in any case due to the large amount of data points. The connection of the input data points can be controlled using a linear (’elinear’), a spline interpolation (’espline’) or a piecewise cubic interpolation (’epchip’) method.
Fig. 25 shows the effects of the different curve fitting methods. In this dataset an anomaly was inserted. As one can see in Fig. 25b, only the exact curve fitting methods cover this data anomaly whereas the methods shown in Fig. 25a that only rely on the intersections do not.
Each user has to decide individually which curve fitting method fits his needs best, because it highly depends on the number of input data points and the desired accuracy of the visualization. For example, creating a carpet plot based on noisy experimental data with outliners using a method that just interpolates the intersections may be better, although

there might be a huge set of data points. Whereas the presentation of carpet plots based on analytic calculations will benefit from an exact curve fitting approach.



(a) Intersection curve fittings



(b) Full data curve fittings

Fig. 3.25: Comparison of different curve fitting methods

Not only the curve fitting parameter influences the smoothness of the carpet plot lines. A proper rendering and file export method is necessary as well. The most simple way to export the plot to a file is taking a screen shot. Nevertheless, it is recommended to use Matlab's built-in file export functions, or a third party add-on. Especially *export_fig* is recommended as it supports vector graphics, anti aliasing and a variety of graphic file formats (Oliver Woodford 2012).

Some additional features that were not a top priority are shown in Fig. 26, among them the possibility to add a filled contour plot to the carpet as well as adding constraints in different forms like a hatched line (Rob McDonald 2007).

```
o = carpetplot(a,b,Y);
label(o,'a-axis','b-axis')
set(o,'style','standard','blabelspacing',0.4,'barrowspacing',0.4);

showpoint(o,27,37);
colormap cool;
contourf(o,a,b,A.*B);

constraint(o,'y<1200','fill')
constraint(o,'y<1200','hatchedline','r-',-45)
```

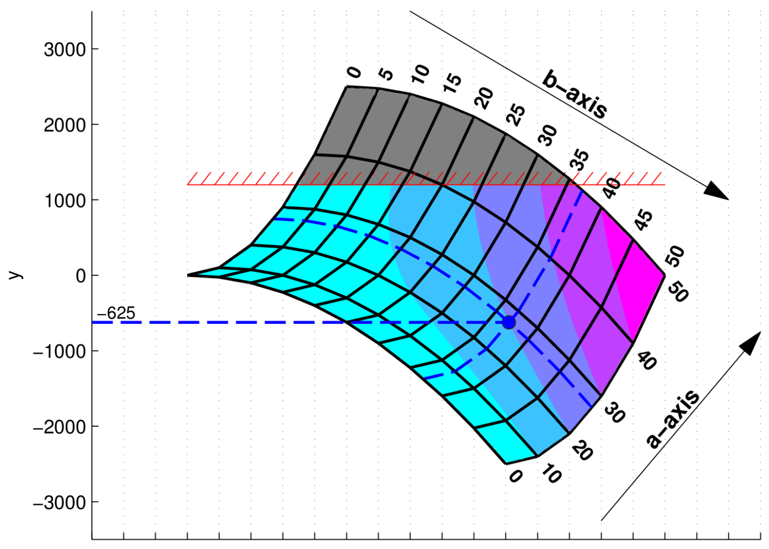


Fig. 3.26: Demonstration of additional carpet plot features

The carpet plot class also offers support for the handling of multiple plots. By using Matlab's 'hold on' function, it is possible to plot another carpet into the figure. By assigning a z coordinate to every carpet, it is also possible to interpolate a whole carpet. In addition, the showpoint function also works with multiple plots as illustrated in Fig. 27.

```
hold on;

o1 = carpetplot(a,b,X,Y,10);
o2 = carpetplot(a,b,X.*0.8+150000,Y.*0.8+2000,20);
o3 = carpetplot(a,b,X.*0.7+600000,Y.*0.7+5000,40);

oi = interpolateplot(o1,o2,o3,30,'linear');
set(get(oi,'linehandles'),'LineStyle','--','Color',[0.5 0.5 0.5])

showpoint(o1,o2,o3,oi,27,37,'g-.');
```

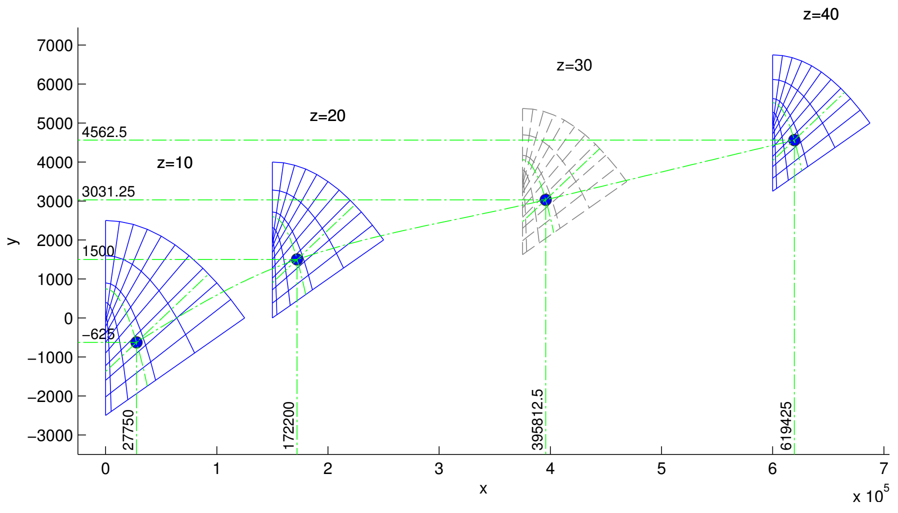
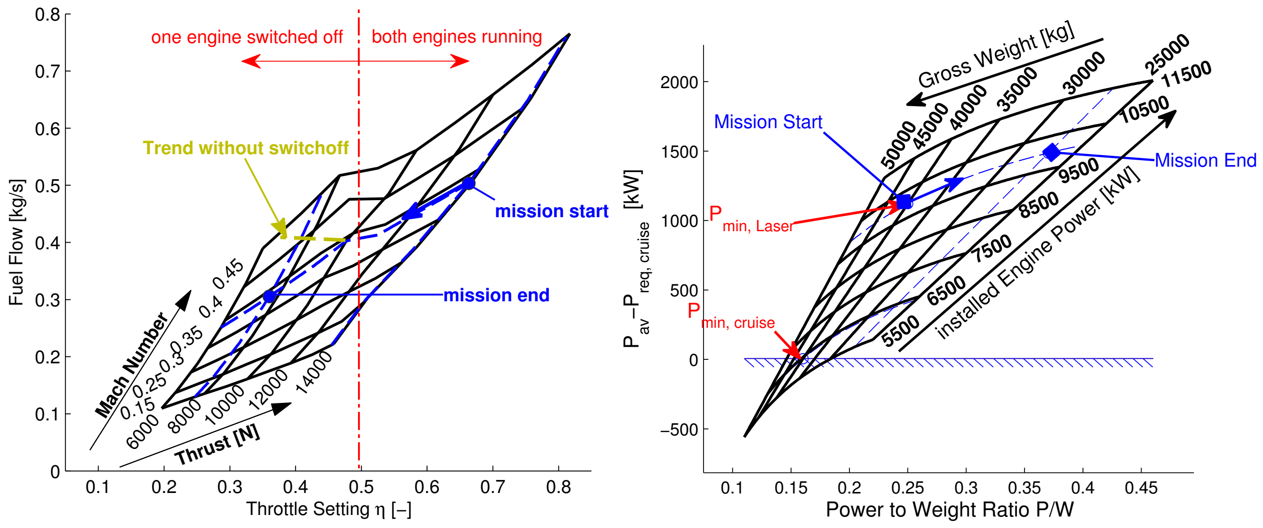


Fig. 3.27: Demonstration of additional carpet plot features for multiple carpet plots

Fig. 28 shows two trade studies performed by Philipp Juretzko using an early version of the Matlab class. The project conducted by students of the Technical University of Munich was created as a response to the 2012/13 AIAA Graduate Team Aircraft Design Competition request for proposals (RFP) titled “High Altitude Long Endurance (HALE) Unmanned Aerial System (UAS) for Missile Defense with Directed Energy (DE) Laser Weapon." As one can see, besides the implemented functions like labeling, constraints or curve fitting, additional elements had been inserted using Matlab’s WYSIWYG plot tool box.



(a) Engine power trade study

(b) Fuel Consumption trade study

Fig. 3.28: Performance trade studies for the HALE UAS Tarantula project

4. Case Study: Design of a Short Haul Narrow Body Commercial Airliner

4.1. Introduction to the Case

In this case, the use of the carpet plot class for Matlab will be demonstrated by conducting trade studies. Therefore, a rough preliminary design for a short haul narrow body aircraft, similar to Boeing's 737 or Airbus' A320, will be designed.

Parameter	**Value** [a]	**Value** [b]
Range	$2500\,\mathrm{nmi}$	$4630\,\mathrm{km}$
Wing span	$122\,\mathrm{ft}$	$37.2\,\mathrm{m}$
MTOW	$170000\,\mathrm{lbf}$	$77110,7\,\mathrm{kg}$
Passengers	$180\,\mathrm{pax}$	$180\,\mathrm{pax}$
Cruise Speed	$0.78\,\mathrm{Mach}$	$0.78\,\mathrm{Mach}$
Cruise altitude	$37000\,\mathrm{ft}$	$11277.6\,\mathrm{m}$

[a]Using the gravitational FPS system
[b]Using SI units

Table 4.1: Requirements for the airliner design

The plane should be powered by high bypass ratio turbofan engines. In addition, it should have the capacity to transport 180 passengers to a destination $2\,500\,\mathrm{nmi}$ away at a cruising speed of $0.78\,\mathrm{Mach}$. All these requirements listed in table 4.1 were derived from the current capabilities of Airbus' A320 and Boeing's 737 family (L. Jenkinson, P. Simpkin & D. Rhodes 2013).

4.2. Constraint Diagram

> "Most of the design requirements which a customer specifies for an aircraft are performance capabilities, so in most cases it is performance analysis which answers the question, 'Will this aircraft meet the customer's needs?'" (Brandt, Whitford, R. Stiles & J. Bertin 2004, p. 123)

There are six key parameters in aircraft design identified by Raymer (Raymer 2002, p.70):

1. Thrust to weight ratio T/W
2. Wing loading W/S
3. Aspect ratio
4. Sweep Λ
5. Taper ratio λ
6. Airfoil t/c

The first two, the thrust to weight ratio and the wing loading, are the most important parameters influencing the aircraft's performance. Therefore, the goal of a constraint analysis is to find initial values for T/W and W/S which fulfill the performance constraints defined by the customer as well as national and international regulations. For large turbine-powered airplanes the airworthiness standards of JAR 25 and FAR 25 apply as a prerequisite to get a certification (JAR-25 Large Aeroplanes 1994) (Federal Aviation Administration n.d.).
The values for the design parameters, except for T/W and W/S, are illustrated in table 4.2. They were derived from similar existing aircrafts.

Name	**Value**
Aspect Ratio	9.5
Sweep Λ	25
Taper Ratio λ	0.25
Airfoil t/c	120

Table 4.2: Design parameters

The distribution of the runway lengths worldwide is illustrated in Fig. 29. The data was gathered from the Sandel Avionics database which covers the runway lengths of most of the airports worldwide (Sandel Avionics ST3400 TAWS Database Information Cycle 1210 2012). A 55 percentile resulting in a required runway length of $5\,000\,\mathrm{ft}$ was chosen, as a higher percentile would have led to the inclusion of many short general aviation airports that are not relevant for commercial airlines.

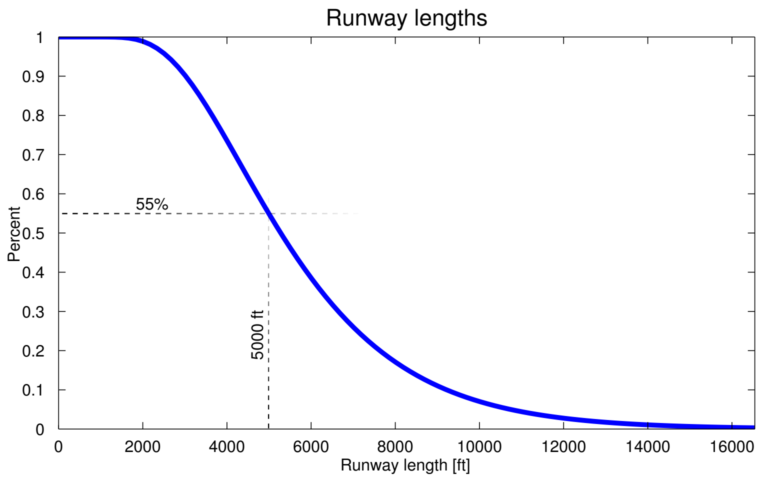


Fig. 4.29: Cumulated distribution plot of runway lengths worldwide

The landing field length requires clearing a 50 ft obstacle and the required W/S can be calculated using Brandt's equation as shown in 4.1 (Brandt et al. 2004, p. 91).

$$S_{Landing} = 80\left(\frac{W}{S}\right)\left(\frac{1}{\sigma C_{L_{max}}}\right) + S_a \tag{4.1}$$

The takeoff distance which is required to clear a 50 ft obstacle depends on T/W as well as on W/S. In the initial sizing process, the relation can be roughly estimated using a takeoff parameter (TOP) which is a statistical approach (Brandt et al. 2004, p. 88).

$$TOP = \frac{\left(\frac{W}{S}\right)}{\sigma C_{L_{TO}} \left(\frac{T}{W}\right)} \tag{4.2}$$

For the cruise and the second segment climb constraint, Brandt's "Master Equation" for constraint diagrams will be used (Brandt et al. 2004, p. 176). For the cruise case, the term $\frac{\partial V}{\partial t}$ for acceleration and $\frac{\partial h}{\partial t}$ for altitude change can be set to zero. For the second segment climb, the corresponding values have to be inserted.

$$\left(\frac{T}{W}\right) = \frac{\beta}{\alpha}\left\{\frac{q}{\beta}\left[\frac{C_{D0}}{\frac{W_{To}}{S}} + k_1\left(\frac{n\beta}{q}\right)^2 \frac{W_{To}}{S}\right] + \frac{1}{V}\frac{\partial h}{\partial t} + \frac{1}{g}\frac{\partial V}{\partial t}\right\} \tag{4.3}$$

The constraint diagram, shown in Fig. 30, results in the following values for the baseline design:

- $\frac{T}{W} = 0.16$
- $\frac{W}{S} = 68\,\frac{\text{lb}}{\text{ft}^2}$

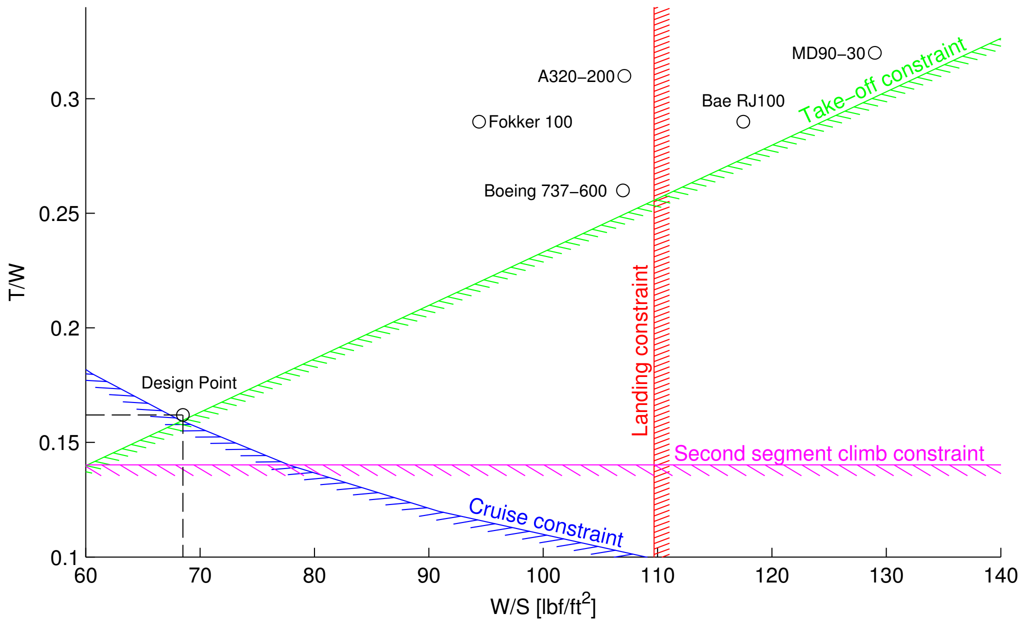
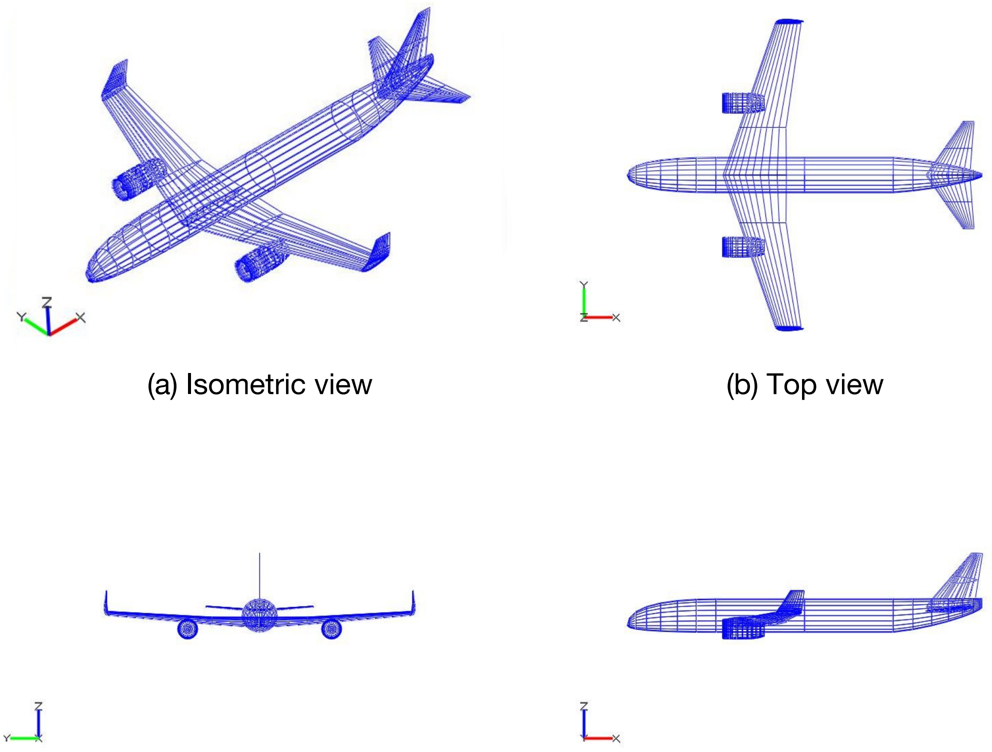


Fig. 4.30: Constraint diagram

4.3. Preliminary Design

In Fig. 31, a first design sketch conducted using openVSP (NASA 2013) illustrates the layout of the aircraft's baseline design. The airframe is similar to those of the majority of commercial passenger planes - a low-wing cantilever monoplane with a single vertical stabilizer and a fuselage mounted empennage.



(a) Isometric view

(b) Top view

(c) Front view

(d) Left side view

Fig. 4.31: Design sketch airliner using OpenVSP

For the engine, a rubber engine model based on historical data was created. The data points in Fig. 32 have been linearized using linear regression and will be used for all following calculations.

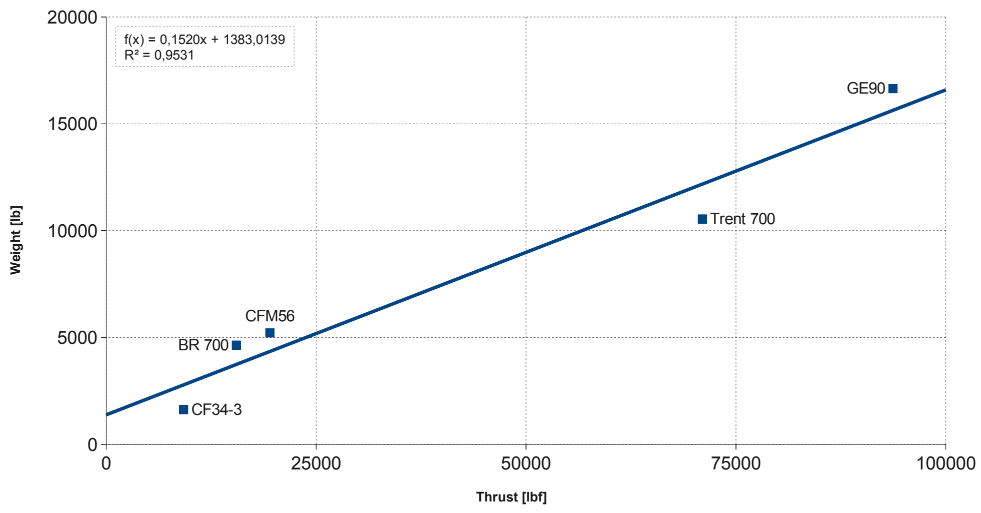


Fig. 4.32: Rubber engine model

4.4. Trade Studies

First of all the "granddaddy" of all trade studies (Raymer 1992, p. 534), the T/W - W/S carpet plot, is conducted. In this carpet plot, the lowest operational empty weight (OEW) that meets all constraints can be found.
The weights were estimated using Raymer's statistical group weights method (Raymer 1992, Table 15.2). As one can see in Fig. 33, the new design point moved to $T/W = 0.25$ and $W/S = 109\,\frac{\mathrm{lb}}{\mathrm{ft}^2}$, the estimated OEW at this point is $87\,675\,\mathrm{lbf}$. Obviously, the design point could have been moved farther to the left without influencing the OEW, but as shown in Fig. 30, this new design point is similar to comparable aircrafts and therefor has been chosen. With this new design point, all following trade studies in this thesis will focus on this T/W and W/S area.

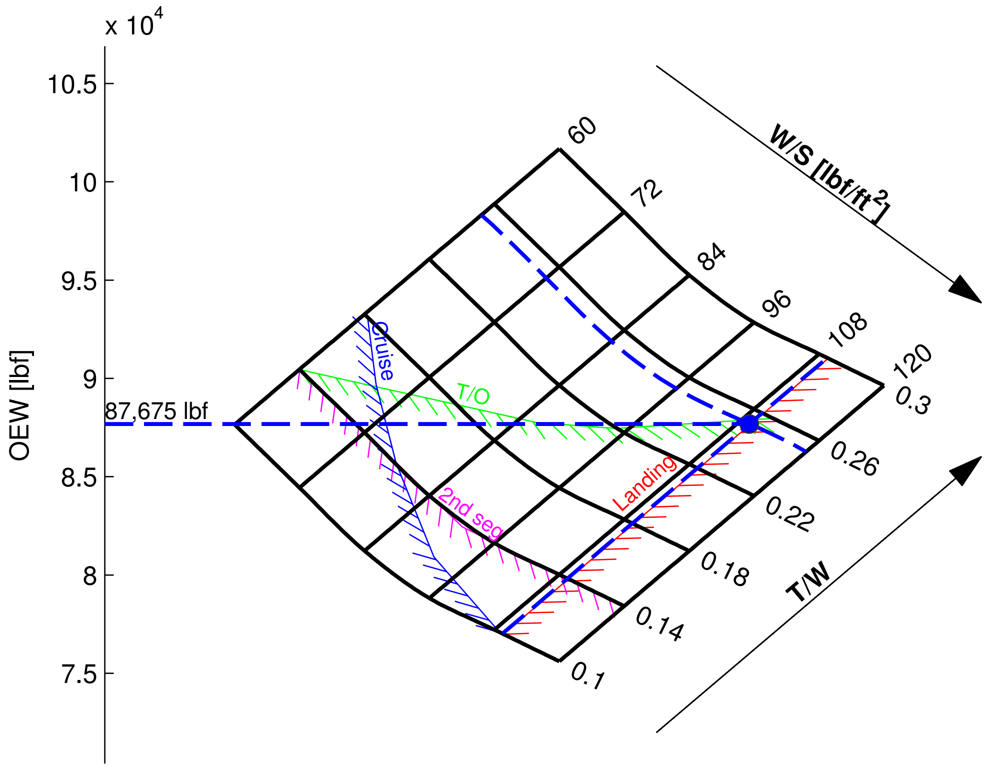
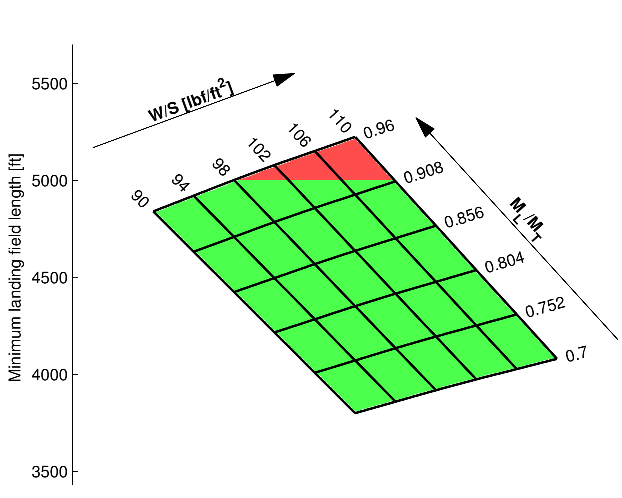
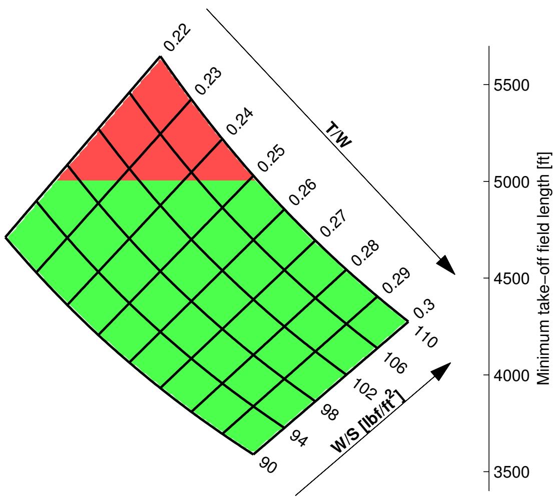


Fig. 4.33: Constraint diagram with OEW

In Fig. 34, field performance studies were performed with the previously identified $5\,000\,\mathrm{ft}$ runway length requirement. The trade studies have been conducted with a fixed MTOW of $170\,000\,\mathrm{lbf}$ and a stage length of $2\,500\,\mathrm{nmi}$. Both trade studies depend on the wing loading. The landing field also depends on the mass fraction (M_L/M_T) which is the landing mass divided by the takeoff mass (Lloyd R. Jenkinson & James F. Marchman III 2003, p. 86). Fig. 34a shows that the mass fraction has a huge impact on the required landing field. In comparison, the impact of the wing loading is rather limited. The thrust-to-weight-ratio has a huge impact on the takeoff field performance presented in Fig. 34b. In combination with a possible reduction of the wing loading, a more powerful engine would result in a significantly shorter takeoff length.



(a) Landing field trade study



(b) Takeoff length trade study

Fig. 4.34: Runway length trade studies

Using Breguet's range equation, a trade study shows the effect of wing loading and thrust-to-weight ratio on the aircraft's maximum range. Again, the MTOW is fixed at $170\,000\,\mathrm{lbf}$. Fig. 35 shows that the aircraft's range is well above the projected $2\,500\,\mathrm{nmi}$ of table 4.1.

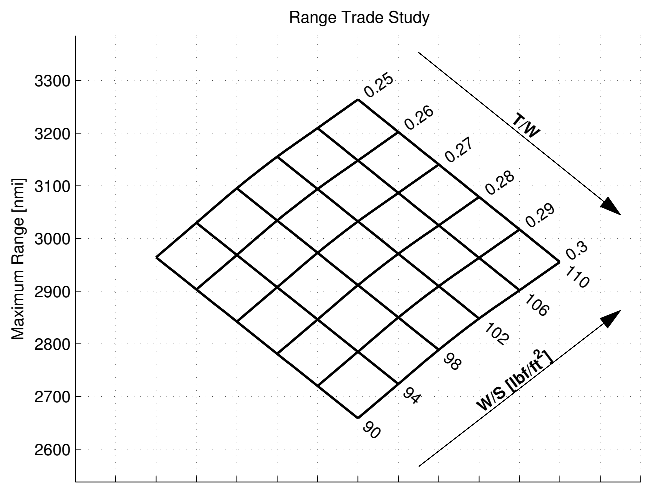
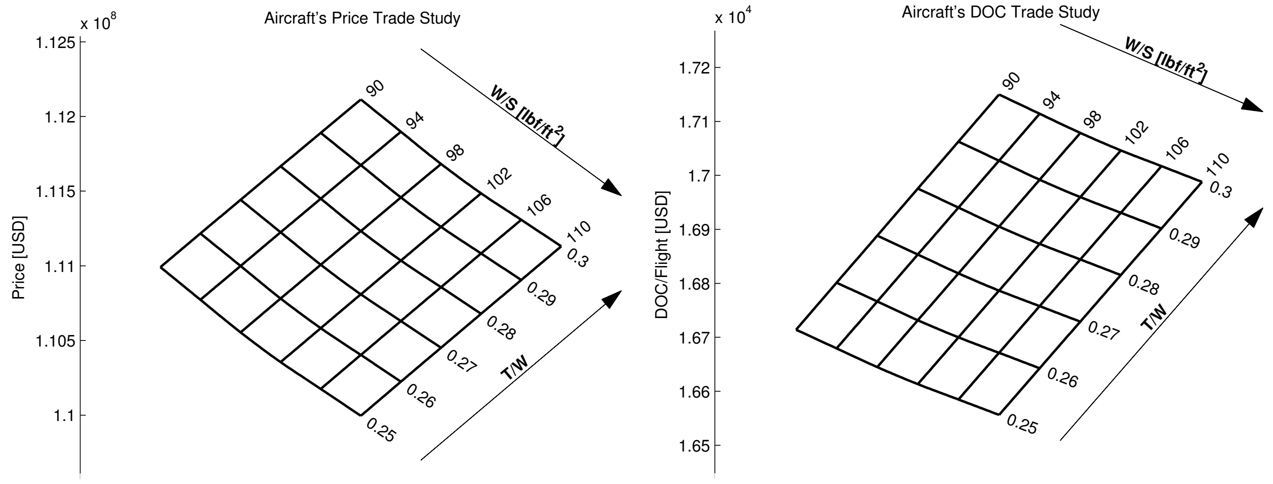


Fig. 4.35: Range trade study

For the economic trade studies in Fig. 36, a simple parametric cost estimation based on historical data was used. The stage length of $2\,500\,\mathrm{nmi}$ is fixed.



(a) Aircraft price (b) Direct operational costs

Fig. 4.36: Economic trade studies

The seat mile costs are an important indicator for an airline's decision to purchase a new airplane. These costs depend on design variables as well as the seating configuration. Fig. 37 illustrates the impact on the seat mile costs of the three different classes listed in table 4.3.

Name	**Passengers**	**Classes**
Economy	180	1
Mixed	150	2
Executive	120	1

Table 4.3: Class configurations

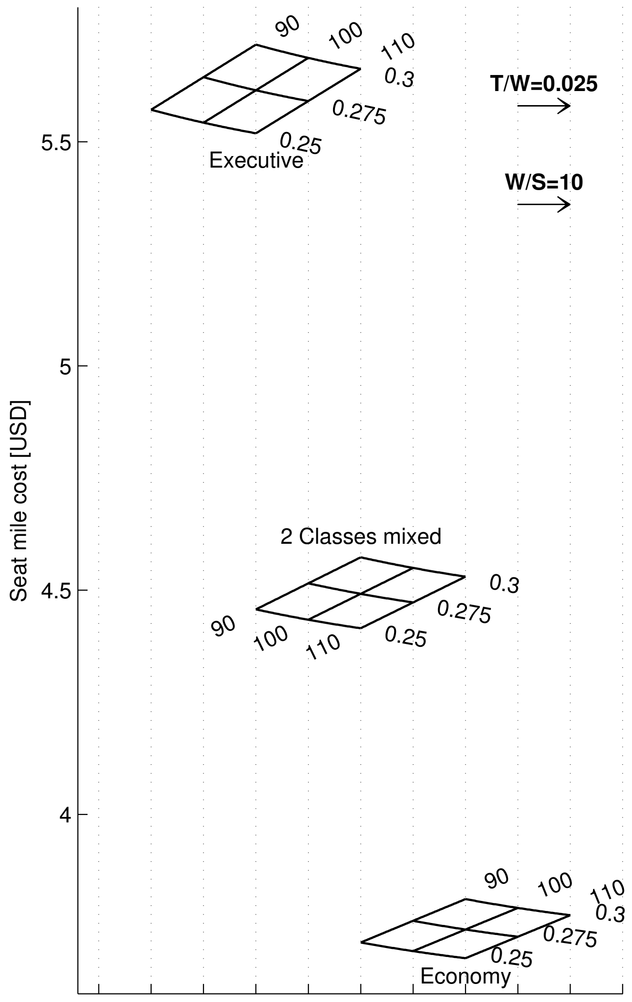


Fig. 4.37: Seat mile costs with different seat configurations

In conclusion, the trade studies resulted in a major change of the design point from $T/W = 0.16$ and $W/S = 68\,\frac{\text{lb}}{\text{ft}^2}$ to $T/W = 0.25$ and $W/S = 109\,\frac{\text{lb}}{\text{ft}^2}$ which, compared to similar aircrafts, is a much more realistic design point, as shown in Fig. 30. This change was necessary after conducting the T/W - W/S carpet plot trade study in Fig. 33 which clearly presented the chance to reduce the OEW. The other trade studies conducted in this thesis focused on minor changes of the T/W and W/S design point; using the carpet plot visualization helped getting a better understanding of the design space. Especially the economic trade studies may be useful to communicate and justify design decisions to a non technical audience.

5. Conclusion and Outlook

Although visualizing data is a very important part of an engineer's work, only a very limited number of methods are known and used in practice. When starting this thesis for instance, like many other aircraft engineers, the author was not aware of the carpet plot technique. But as shown in chapter 1.3., carpet plots have numerous advantages when it comes to multi-dimensional data.
The developed Matlab carpet plot class tried to cover numerous features and graphical elements discussed in the available literature. Thus, people currently using the carpet plot method will be able to migrate to the Matlab class without loosing necessary features. What is more, people new to the carpet plot technique are being offered a huge set of features and possibilities.
The trade studies conducted in chapter 4.4. had a major influence on the development of the tool. Not only did the practical use of the class reveal some bugs, it also helped understand the user interaction and therefore created much benefit to the usability of the tool. The case studies also showed that after creating the data, which can be a hard task, the creation of a first carpet is fairly easy. Sometimes further design modifications, like text or arrow positions, are possible but can be achieved in a few lines of code or manually with Matlab's WYSIWYG plot editor.
The tool has been made public on Mathwork's file exchange at `http://www.mathworks.com/matlabcentral/fileexchange/41467-the-carpetplot-class`. According to the comments and feedback from the Matlab community, further improvements and bug fixes will be made. As the class has a modular design and the source code is well documented, additional features could be added easily.

Bibliography

Angermann, Anne, Beuschel, Michael, Rau, Martin & Wohlfarth, Ulrich (2005): Matlab - Simulink - Stateflow: Grundlagen, Toolboxen, Beispiele, 4 Aufl., Wissenschaftsverlag, Oldenbourg.

Brandt, Whitford, R. Stiles & J. Bertin (2004): Introduction to Aeronautics: A Design Perspective, 2. Aufl., AIAA, Cranfield Institute of Technology.

Eide, Petter LH (2005): Quantification and traceability of requirements, Norwegian University of Science and Technology. Faculty of Information Technology, Mathematics and Electrical Engineering. Department of Computer and Information Science. Pp **130**.

Federal Aviation Administration (n.d.): e-CFR: TITLE 14–aeronautics and space.

IHS ESDU (1996): ESDU 74035 subsonic lift-dependent drag due to the trailing vortex wake for wings without camber or twist.

IHS ESDU (2004a): ESDU 04008 use of carpet plots to represent functions of two variables.

IHS ESDU (2004b): ESDU 04012 examples of construction of carpet plots from experimental data.

JAR-25 Large Aeroplanes (1994).

Joachim Kurzke (2013): gasturb.
URL: *http://www.gasturb.de*

L. Jenkinson, P. Simpkin & D. Rhodes (2013): Butterworth-heinemann - jenkins - civil jet aircraft design - data sets.
URL: *http://www.elsevierdirect.com/companions/9780340741528/default.htm*

Lloyd R. Jenkinson & James F. Marchman III (2003): Aircraft Design Projects, Elsevier Ltd., Boston.

NASA (2013): Open vehicle sketch pad (OpenVSP).
URL: *http://www.openvsp.org/*

Oliver Woodford (2012): export_fig.
URL: *http://www.mathworks.com/matlabcentral/fileexchange/23629-exportfig*

Powers, Sidney (1997): The generation of carpet plots.

Raymer, Daniel (2002): Enhancing Aircraft Conceptual Design using Multidisciplinary Optimization, dissertation, KTH, Stockholm.

Raymer, Daniel P. (1992): Aircraft Design: A Conceptual Approach, 2. Aufl., American Institut of Aeronautics, Reston.

Rob McDonald (2007): Hatched lines and contours.
URL: *http://www.mathworks.de/matlabcentral/fileexchange/29121-hatched-lines-and-contours*

Sandel Avionics ST3400 TAWS Database Information Cycle 1210 (2012).

Schneider, Markus (2000): How to generate carpet plots.

Vitaly Friedman (2008): Data visualization and infographics.
URL: *http://www.smashingmagazine.com/2008/01/14/monday-inspiration-data-visualization-and-infographics/*

Appendices

A Carpet Plot Class Help File

This is the help file of the Matlab carpet plot class which is accessible inside Matlab.

```
help carpetplot
```

There are more help files for every method and property available for further instructions of the usage.

```matlab
% CARPETPLOT is a class for creating carpet plots and cheater plots
%
% INTRODUCTION
%----------------------------------------------------------------------
%
% Carpet plots are a way to illustrate three (Fig. 2) or four (Fig. 1)
% variables in an easy to read two dimensional plot. The carpet plot
% class offers the possiblity to handle different input data, add
% labels, show interpolated points inside the plot and some further
% functions.
%
% For a quick start enter showdemo carpetplot.
%
%
%   |           _____________________ 4      |     .  .  .__.__.__.__.__.__.40
%   |          /      /      /      /        |     .  . /. /. /. /. /. /. /.
%   |         /      /      /      /         |     .  .  ./ ./ ./ ./ ./ ./ .
%   |        /______/______/_____ / 3        |     .  .__.__.__. _.__.__.30.
%   |       /      /      /      /           |     . /. /. /. /. /. /. /.  .
%  Y|      /      /      /      /    A      Y|     ./ ./ ./ ./ ./ ./ ./ .  .
%   |     /______/______/______/ 2           |     .__.__.__.__. _.__.20.  .
%   |    /      /      /      /              |    /. /. /. /. /. /. /.  .  .
%   |   /      /      /      /               |   / ./ ./ ./ ./ ./ ./ .  .  .
%   |  /______/______/______/ 1              |  /__.__.__.__.__.__. 10  .  .
%   | 0.1    0.2    0.3    0.4               | 1   2  3  4  5  6  7  .  .  .
%   |            B                           |     .  .  .  .  .  .  .  .  .
%   |_____________________________________   |_______________________________________
%                      X
%    Fig 1: A four variable carpet plot.      Fig 2: A cheater plot. The intersections
%                                             line up vertically and there is no
%                                             x-axis.
%
% DESCRIPTION
%----------------------------------------------------------------------
% obj = carpetplot(a,b,x) creates a cheater plot object and plot's it.
% The input data can be eather scattered data or matrices.
%
% obj = carpetplot(a,b,x,z) creates a cheater plot object with a
% z-coordinate useful for plot interpolation.
%
% obj = carpetplot(a,b,x,y) creates a four variable carpet plot object.
% The input data can be eather scattered data or matrices.
%
% obj = carpetplot(a,b,x,z) creates a four variable carpet plot
% object with a z-coordinate useful for plot interpolation.
%
% obj = carpetplot(...,'PropertyName',PropertyValue,...) changes the
% appareance by changing the line properties of the carpet plot.
%
% obj = carpetplot(...,lineSpec) plots the carpet plot with the
% given lineSpec.
%
% USAGE
%----------------------------------------------------------------------
% The carpetplot constructor creates a carpetplot object and plots the
% carpet. Note that the size of the input vectors and/or matrices must
% match.
%
% The appereance as well as the plot's input data can be changed using
% different methods and properties that are part of the created object.
% The changes will update the plot automatically if they are done using
% the set() and get() methods.
%
% EXAMPLE 1
%----------------------------------------------------------------------
% A simple cheater plot with matrix data input.
%
%   a = 1:0.25:2;
%   b = 1:20:100;
%   [A B] = meshgrid(a,b);
%   Y = A.*B;
%
%   o = carpetplot(A,B,Y);
%   label(o,'A-Axis','B-Axis')
```

```
%
% EXAMPLE 2
%--------------------------------------------------------------------
% A four variable carpet plot using scattered data input.
%
%   a = [1 1 1 2 2 2 3 3 3];
%   b = [10 20 30 10 20 30 10 20 30];
%   X = a.^2+a;
%   Y = a+b;
%
%   o = carpetplot(a,b,X,Y);
%   label(o,'A-Axis','B-Axis')
%
%
% CARPETPLOT Properties:
%--------------------------------------------------------------------
% The properties should be accessed using get() and set() methods.
%
% Properties for data handling:
%    k0              - X translation of the whole plot (three-variable-plots only)
%    k1              - X translation of a values (three-variable-plots only)
%    k2              - X translation of b values (three-variable-plots only)
%    curvefitting    - Curve fitting method
%    atick           - Interval of the a values
%    btick           - Interval of the b values
%    zvalue          - Z coordinate of the carpet plot
%
% Properties for visualization:
%   All these values influence the vizualisation of labels, lines and
%   arrows. They will automatically refresh the plot or its
%   labels.
%   If you want to change the appereance of the plot fast, check out style
%
%   style            - Set a pre defined style. Changes most of the
%                      properties beneath
%
%   aarrowflipped    - Flip the position of the a-axis arrow
%   barrowflipped    - Flip the position of the b-axis arrow
%   aarrowspacing    - Space between the a-axis arrow and the plot
%   barrowspacing    - Space between the b-axis arrow and the plot
%
%   alabelflipped    - Flip the position of the a-axis label
%   blabelflipped    - Flip the position of the b-axis label
%   alabelspacing    - Space between the a-axis label and the plot
%   blabelspacing    - Space between the b-axis label and the plot
%
%   asuffix          - Suffix for all a-axis values
%   bsuffix          - Suffix for all b-axis values
%   aprefix          - Prefix for all a-axis values
%   bprefix          - Prefix for all b-axis values
%
%   atextspacing     - White spaces before or after the a-axis values
%   btextspacing     - White spaces before or after the b-axis values
%   atextrotation    - Defines if the a-axis values will be rotated
%   btextrotation    - Defines if the b-axis values will be rotated
%   atextflipped     - Flip the position of the a-axis values
%   btextflipped     - Flip the position of the b-axis values
%   atextspacing     - Space between the a-axis text and the carpet
%   btextspacing     - Space between the b-axis text and the carpet
%
%   zalignement      - Position of the plot's z-label
%
% Handles
%   In order to further customize the carpet plot, it is possible to access all
%   graphic handles directly. Changes applied to the handles may be lost after
%   replotting the carpet.
%
%   alines           - Handles of a-value lines
%   blines           - Handles of b-value lines
%   lines            - Handles of all lines
%
%   aextraplines     - Handles of a-value extrapolated lines
%   bextraplines     - Handles of b-value extrapolated lines
%   extraplines      - Handles of all extrapolated lines
```

```matlab
%
%   amarkers         - Handles of a-value markers
%   bmarkers         - Handles of b-value markers
%   markers          - Handles of all markers
%
%   alabeltext       - Handle of the a-axis caption
%   blabeltext       - Handle of the b-axis caption
%   labels           - Handles of both axes captions
%
%   atext            - Handles of the a-axis labels
%   btext            - Handles of the b-axis labels
%   text             - Handles of the labels
%
%   aarrow           - Handle of the a-arrow
%   barrow           - Handle of the b-arrow
%   arrows           - Handle of the arrows
%
%   zlabeltext       - Handle of the z-label
%
% CARPETPLOT Methods:
%---------------------------------------------------------------------
%    inputdata       - Change the input data of the carpet plot
%    reset           - Reset all changes that were made manually
%    refresh         - refresh the carpet plot and/or its labels
%
%    alabel          - Add labels to the carpet plot's a-axis
%    blabel          - Add labels to the carpet plot's b-axis
%    labels          - Label both axes at once
%    zlabel          - Add a label to the carpet plot (z-axis)
%
%    legend          - Add a legend to a carpetplot
%    cheaterlegend   - Adds arrows to a cheater plot to indicate x-axis intervals
%
%    set             - Set method for carpet plot properties
%    get             - Get method for carpet plot properties
%
%    contourf        - Insert a filled contour plot to the carpet plot
%    plot            - Insert a plot into the carpet plot
%    hatchedline     - Insert a hatchedline into the carpet plot
%    constraint      - Add a constraint to the carpet plot
%    showpoint       - Show a point in the carpet plot
%    interpolateplot - Interpolate a carpet plot
%    lattice         - Creates a lattice plot out of multiple cheater plots
%
%    abtoxy          - Transform to the carpet plot's coordinate system
%
%
% Matthias Oberhauser
% matthias.oberhauser@tum.de
%
% Revision History:
% 22 April 2013 v. 1.0
% 01 May 2013 v. 1.01
%       - Added lattice() method + bug fixes for lattice plots
%       - zlabels will update automatically
%       - Added documentation for refresh method
%       - Updated documentation
```

B Demo Plots

This is a demo file published as a HTML file using Matlab. It shows some example plots covering most of the features of the carpet plot class. It can be accessed via the help file or by typing the following line in Matlab:

```
showdemo carpetplot
```

Carpetplot Class Example Plots

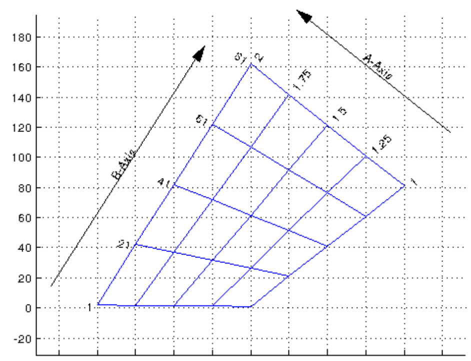
These are some examples plots made with the carpetplot class.

Contents

Simple Example

In the beginning a simple example. More details about creating and labeling a carpet plot will be illustrated in the following examples.

```
set(gcf,'Renderer','OpenGL')

a = 1:0.25:2;
b = 1:20:100;
[A B] = meshgrid(a,b);
Y = A.*B;

o = carpetplot(A,B,Y);
label(o,'A-Axis','B-Axis')

grid on;
```



Input Data

The carpetplot class supports different kind of input data. The data can be exact (a matrix containing all datapoints) or scattered data. In this case the data points will be interpolated using TriScatteredInterp.

There are two different types of carpet plots:

The cheater plot

```
obj = carpetplot(a,b,Y)
obj = carpetplot(a,b,Y,z)
```

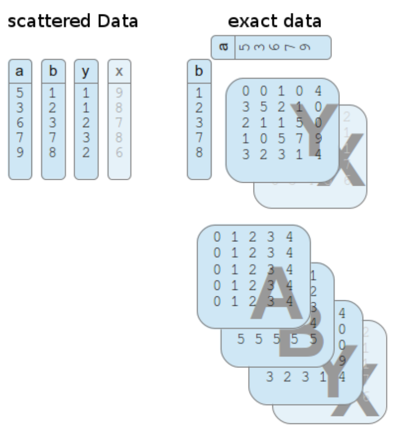
The (real) carpet plot

```
obj = carpetplot(a,b,X,Y)
obj = carpetplot(a,b,X,Y,z)
```

In case of exact data A and B can be eather vectors or meshgrids.

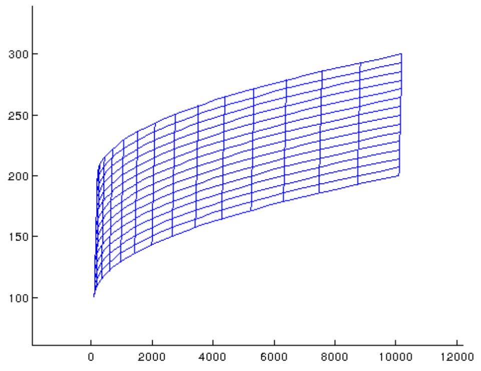
If the input data are scattered data points a,b,(x) and y must be vectors with the corresponding coordinates of the datapoints.

The z parameter is only used for the interpolation of whole carpet plots or lattice plots.



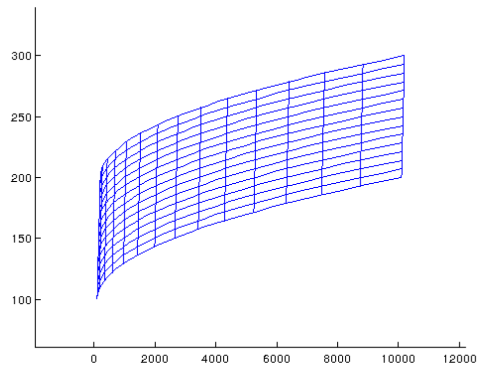
A plot using exact data input and Meshgrids

```
a = linspace(1,100,30);
b = linspace(100,200,20);
[A B] = meshgrid(a,b);
X = A.^2+B;
Y = A+B;
o = carpetplot(A,B,X,Y);
```



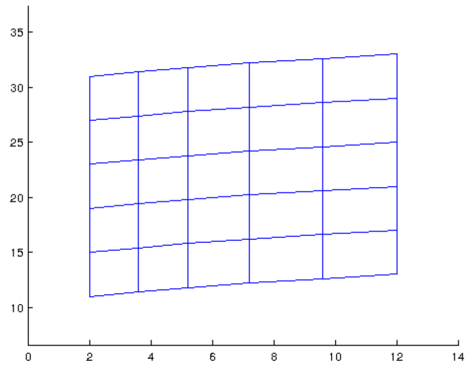
Using the vectors for a and b has the same effect

```
clf;
warning off;
a = linspace(1,100,30);
b = linspace(100,200,20);
[A,B] = meshgrid(a,b);
X = A.^2+B;
Y = A+B;
o = carpetplot(a,b,X,Y);
```



Scattered data

```
clf;
a = [1 1 1 2 2 2 3 3 3];
b = [10 20 30 10 20 30 10 20 30];
X = a.^2+a;
Y = a+b;
o = carpetplot(a,b,X,Y);
```



Curve Fitting

In the default case the interpolations of the carpet plot will be calculated and connected with a straight line.

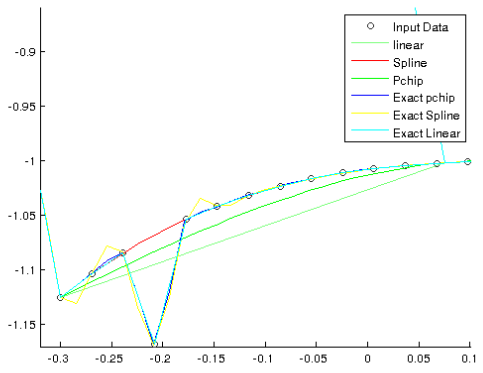
This **linear** curve will produce a cornered carpet if you do not have a lot of intersections.

To avoid this use the **spline** and **pchip** methods to produce curved lines with the given intersections.

Alternatively if you have a lot of data points but only wan't to vizualize a few intersections it is recommended to use the exact methods **epchip** or **espline** pr **elinear** as these methods consider all input datapoints for plotting.

The following example illustrates the different effects of the curve fitting methods.

```
a = linspace(-.5,1,50);
b = linspace(-1,6,14);
[A,B] = meshgrid(a,b);
Y = A.^3+B;
X = A-B*.2;
Y(1,4) = Y(1,4)-0.1;
clf;
hold on;

oinput = carpetplot(A,B,X,Y,'o','Color',[0.2 0.2 0.2]);
set(oinput,'aTick',linspace(-.5,1,50),'bTick',linspace(-1,6,14))
ospline = carpetplot(A,B,X,Y,'-','Color',[1 0 0]);
olinear = carpetplot(A,B,X,Y,'-','Color',[0.5 1 0.5]);
set(olinear,'curveFitting','linear','aTick',linspace(-.5,1,5))
set(ospline,'curveFitting','spline','aTick',linspace(-.5,1,5))
opchip = carpetplot(A,B,X,Y,'-','Color',[0 1 0]);
set(opchip,'curveFitting','pchip','aTick',linspace(-.5,1,5))
oepchip = carpetplot(A,B,X,Y,'-','Color',[0 0 1]);
set(oepchip,'curveFitting','epchip','aTick',linspace(-.5,1,5))
oespline = carpetplot(A,B,X,Y,'-','Color',[1 1 0]);
set(oespline,'curveFitting','espline','aTick',linspace(-.5,1,5))
oelinear = carpetplot(A,B,X,Y,'-','Color',[0 1 1]);
set(oelinear,'curveFitting','elinear','aTick',linspace(-.5,1,5))

legend([oinput olinear ospline opchip oepchip oespline oelinear] ...
    ,'Input Data','linear','Spline','Pchip','Exact pchip','Exact Spline','Exact Linear');

xlim([-0.32 0.1]); ylim([-1.17 -0.86])
```



As you can see all curve fitting methods have their trade offs. **pchip** and **spline** don't consider the points between the intersections.

linear just draws a straight line.

The Exact Methods consider all points but are only usable if you have a lot of input data. **espline** especially with the data anomaly swings out a lot.

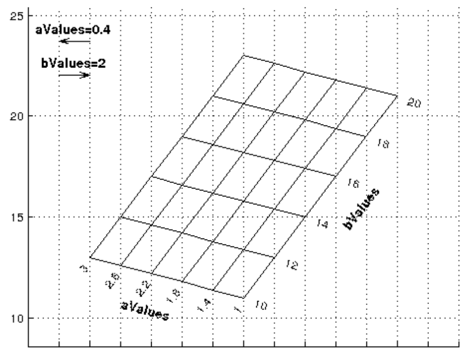
Cheater Plots

Cheater plots do not have an x axis. The x-values for the vizualisation will calculated using

```
X = k0 + a*K1 + b*K2
```

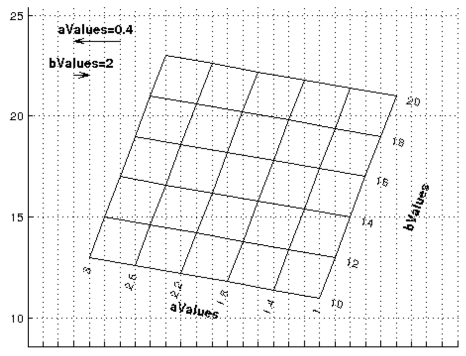
The K values controll the plotting direction of the plot and as default the intersections align vertically. The allignement does only work if the a and b values are equally spaced.

```
clf;
aValues = [1 1 1 2 2 2 3 3 3];
bValues = [10 15 20 10 15 20 10 15 20];
Y = aValues+bValues;
o = carpetplot(aValues,bValues,Y);
label(o);
set(o,'style','clean','alabelspacing',0.15);
[arrowH,textH] = cheaterlegend(o,'northwest');
grid on;
```



Use get and set to change the K Values. The Intersections will not allign vertically anymore.

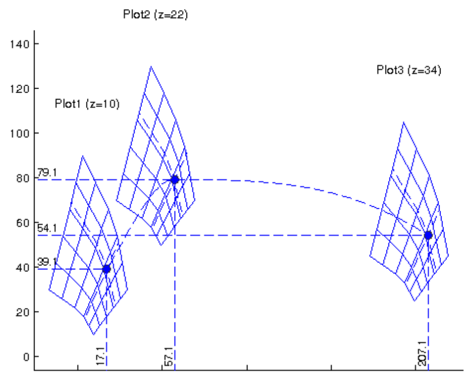
```
delete(arrowH,textH)

set(o,'K1',get(o,'K1')*3)

cheaterlegend(o,'northwest');
grid on
```



Multiple Plots

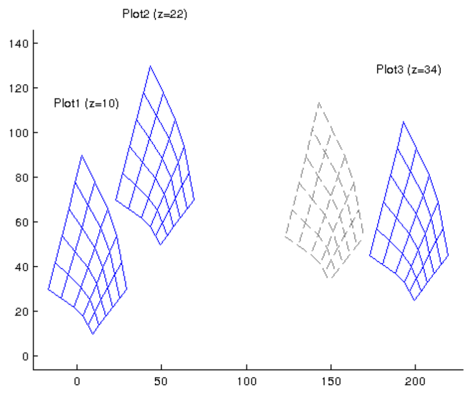
The carpetplot class supports multiple plots. They can just be plotted into one figure by using the `hold on` command.

```
clf;
a =[1;2;3;1;2;3];
b =[10;10;10;30;30;30];
x = b-a.^3;
y = a.*b;

hold on;
o1 = carpetplot(a,b,x,y,10);
o2 = carpetplot(a,b,x+40,y+40,22);
o3 = carpetplot(a,b,x+190,y+15,34);

zlabel(o1,'Plot1 (z=10)');
zlabel(o2,'Plot2 (z=22)');
zlabel(o3,'Plot3 (z=34)');
[hlines hmarkers htext] = showpoint(o1,o2,o3,1.7,23);
snapnow; %Only needed for publishing
```



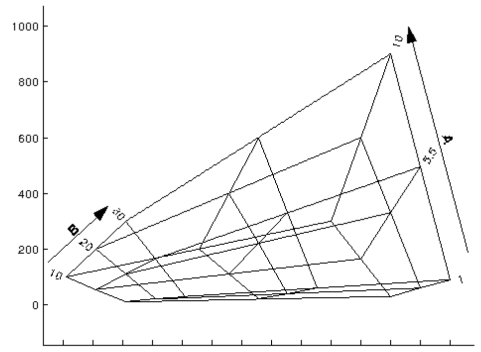
A whole plot can be interpolated. The showpoint lines and text can be deleted using the handles.

```
delete(hlines); delete(hmarkers); delete(htext);
oi = interpolateplot(o1,o2,o3,30);
set(get(oi,'aLines'),'color',[0.5 0.5 0.5],'LineStyle','--');
set(get(oi,'bLines'),'color',[0.5 0.5 0.5],'LineStyle','--');
snapnow; %Only needed for publishing
```



Altough cheater plot's do not have an x-axis multiple carpets can be plotted using the lattice method. The x-axis shifting of the plot will represent the plot's z-value.

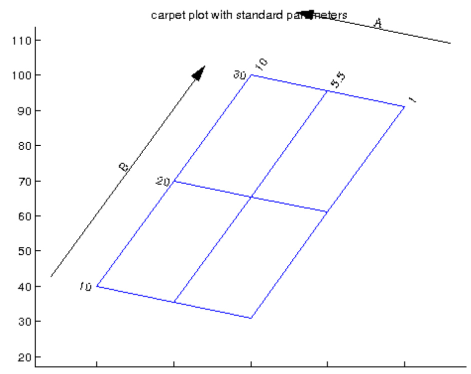
```
clear;
clf;

a = linspace(1,10,3);
b = linspace(10,30,3);
[A B] = meshgrid(a,b);
X1 = A.*B;
X2 = (A.*B).*2;
X3 = (A.*B).*3;


hold on;
o1 = carpetplot(A,B,X1,-124);
set(o1,'style','standard');
o2 = carpetplot(A,B,X2,500);
set(o2,'k0',20,'style','standard')
o3 = carpetplot(A,B,X3,3000);
set(o3,'k0',40,'style','standard')

alabel(o3); blabel(o1);

lLines = lattice(o1,o2,o3,'lines');
set(lLines,'LineWidth',0.5);
```



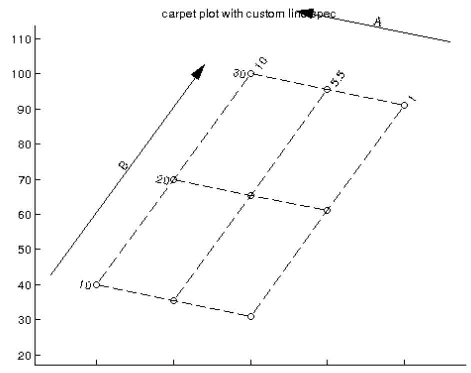
Styles

The carpetplot class supports the possibility to customize the carpet plot with built in styles as well as with custom parameters.

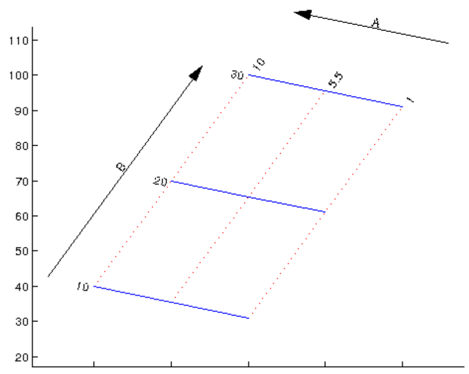
```
hold off;
clear;
clf;

a = linspace(1,10,3);
b = linspace(10,30,3);
[A B] = meshgrid(a,b);
X = A+3.*B;

o = carpetplot(A,B,X);
label(o);
title('carpet plot with standard parameters')
snapnow; %Only needed for publishing
```



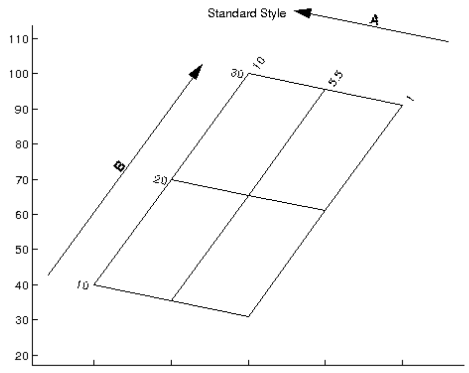
```
o = carpetplot(A,B,X,'ko--');
label(o);
title('carpet plot with custom line spec')
snapnow; %Only needed for publishing
```

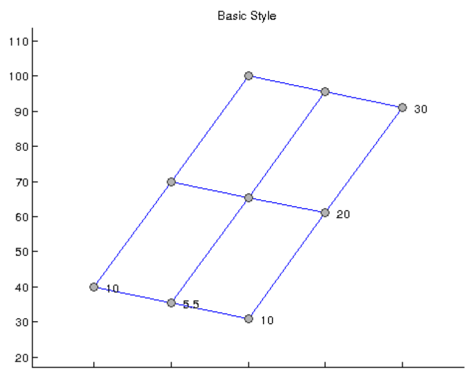


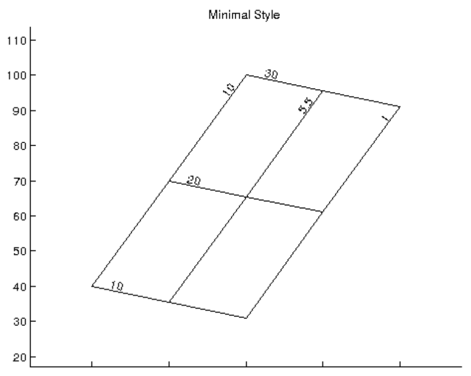
```
o = carpetplot(A,B,X);
label(o);
set(o.alines,'color',[1 0 0],'linestyle',':')
snapnow; %Only needed for publishing
```



```
o = carpetplot(A,B,X); label(o);
set(o,'style','standard');
title('Standard Style');
snapnow; %Only needed for publishing
set(o,'style','basic');
title('Basic Style');
snapnow; %Only needed for publishing
set(o,'style','minimal');
title('Minimal Style');
snapnow; %Only needed for publishing
```





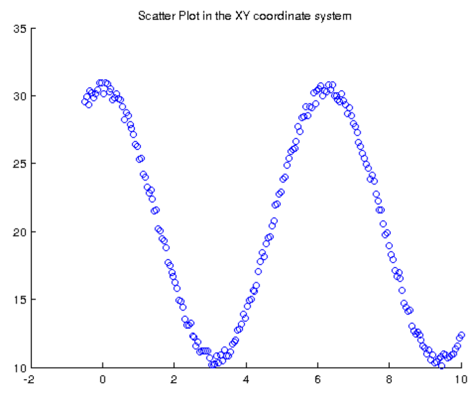


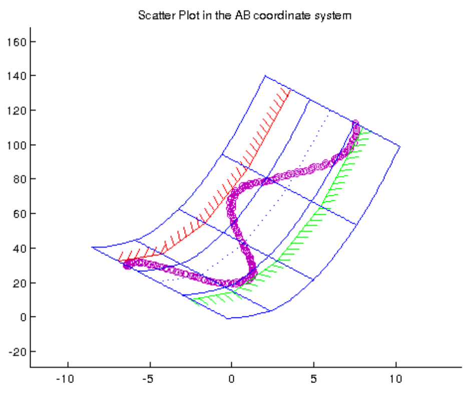
Transform coordinate Systems

The carpetpot class has three build in functions to draw hatchedlines, line plots, and filled contours into the a b coordinate systems. If there are other plots needed it is possible to transform any vectors or matrixes into the carpet plot's coordinate system

```
x = linspace(-0.5,10,200);
rng(0,'twister');
y = 10*cos(x)+ rand(1,200)+20;

hold off;
scatter(x,y)
title('Scatter Plot in the XY coordinate system');
snapnow; %Only needed for publishing

a = linspace(-.5,10,5);
b = linspace(-1,40,4);
[A,B] = meshgrid(a,b);
Y = A.^2+B;
X = A-B*.2;

o = carpetplot(A,B,X,Y);
set(o,'curvefitting','pchip');
hold on;
[x, y] = o.abtoxy(x,y);
scatter(x,y)
title('Scatter Plot in the AB coordinate system');

hatchedline(o,linspace(-.5,10,5),ones(1,5)*32,'r-',-45);
hatchedline(o,linspace(-.5,10,5),ones(1,5)*10,'g-',45);
plot(o,linspace(-.5,10,5),ones(1,5)*20,'b:');
```

```
snapnow; %Only needed for publishing
```

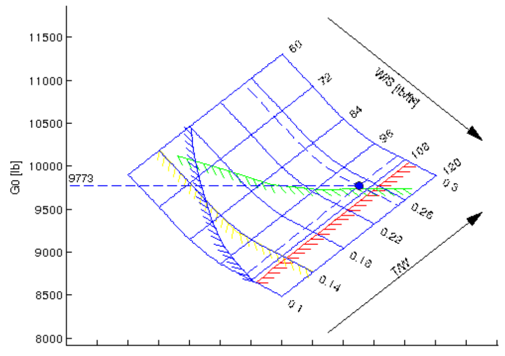




Constraint Plot with fixed Weight trade study

This is a constraint plot common in aircraft design. The fixed weights of planes with different Wingloading [W/S] and Thrust/Weight-Ration [T/W] had been take in into account. The data was created using spreadsheet calculation.

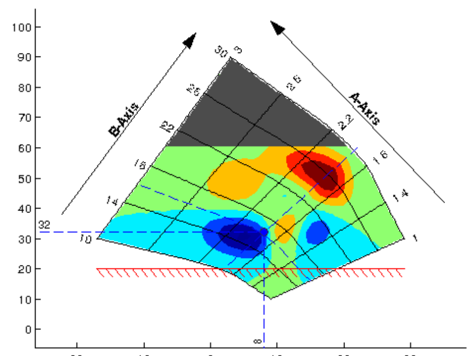
```
clear;
clf;

% Weight Estimations for different T/W and W/S
TW =  [0.1000    0.1200    0.1500    0.2000    0.2500    0.3000    0.1000    0.1200    0.1
500    0.2000    0.2500 0.3000    0.1000    0.1200    0.1500    0.2000    0.2500    0.3000
];
WS =  [60    60    60    60    60    60    90    90    90    90    90    90   120   120
120   120   120   120];
G0 =  10^4*[0.9901    1.0042    1.0252    1.0603    1.0953    1.1304    0.8957    0.9097
  0.9308    0.9658    1.0009 1.0359    0.8485    0.8625    0.8835    0.9186    0.9537    0
.9887];

% Constraint data
TW_Land = [ 0    0.0500    0.1000    0.1100    0.1200    0.1300    0.1400    0.1500    0.1
600    0.1700    0.1800 0.1900    0.2000    0.3000];
WS_Land =[109.6875  109.6875  109.6875  109.6875  109.6875  109.6875  109.6875  109.6875
109.6875  109.6875  109.6875 109.6875  109.6875  109.6875];
TW_takeoff =[0    0.0250    0.0500    0.0750    0.1000    0.1250    0.1500    0.1750    0.
2000    0.2250    0.2500 0.2750    0.3000    0.3250];
WS_takeoff =[0   10.7250   21.4500   32.1750   42.9000   53.6250   64.3500   75.0750   85.
8000   96.5250  107.2500 117.9750  128.7000  139.4250];
TW_Cruise =[0.1000    0.1100    0.1200    0.1300    0.1400    0.1500    0.1600    0.1700
  0.1800    0.1900    0.2000];
WS_Cruise =[109.0134   99.1031   90.8445   83.8565   77.8668   72.6756   68.1334   64.1256
   60.5631   57.3755   54.5068];
TW_secondSeg =[0.1402    0.1402    0.1402    0.1402    0.1402    0.1402    0.1402    0.140
2    0.1402];
WS_secondSeg =[60    70    80    90   100   120   130   140   150];

% Create the object and plot it
o = carpetplot(TW,WS,G0);



%Add some labels
alabel(o,'T/W');
blabel(o,'W/S [lb/ft²]');
ylabel('G0 [lb]');

set(o,'curveFitting','pchip','bLabelSpacing',0.2);

hold on;
LandConstr = hatchedline(o,TW_Land,WS_Land,'r-');
TakeOffConstr = hatchedline(o,TW_takeoff,WS_takeoff,'g-');
secSegmentConstr = hatchedline(o,TW_secondSeg,WS_secondSeg,'y-');
CruiseConstr = hatchedline(o,TW_Cruise,WS_Cruise,'b-',-120);
o.showpoint(0.25,105);
```



Contourf Plot

This example uses the simple peaks( ) contour and the fill style for a constraint

```
clear;
clf;
% Generate some Input Data
a =[1;2;3;1;2;3];
b =[10;10;10;30;30;30];
x = b-a.^3;
y = a.*b;

% Create the object and Plot
plotObject = carpetplot(a,b,x,y);
label(plotObject,'A-Axis','B-Axis')

% Change the curve Fitting and style
set(plotObject,'curvefitting','pchip','style','standard','blabelspacing',0.2,'barrowspacin
g',0.2);

% Add the contourf
hold on;
contourf(plotObject,1:0.1:3,10:1:30,peaks(21));

% Add some Constraints
const = constraint(plotObject,'y<60 ','fill',[0.3 0.3 0.3]);
const = constraint(plotObject,'y>20','hatchedline','r-',45);

showpoint(plotObject,2,16);

% Move the label a little bit
snapnow; %Only needed for publishing
```



Published with MATLAB® R2012b

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht.

München am 05.05.2013

B.Sc. Matthias Oberhauser