

Fig



<svg>
1 0 obj
prologues:=0;
%% BoundingBox

Handbuch zu fig2vect

Dipl.-Ing. D. Krause

18. November 2009

Inhaltsverzeichnis

1	Nutzer-Handbuch	5
1.1	Überblick	5
1.1.1	Zweck	5
1.1.2	Warum fig2vect?	6
1.1.3	Lizenz	8
1.2	Installation	9
1.2.1	Installation unter Unix/Linux	9
1.2.2	Installation unter Windows	11
1.2.3	Installation der Ghostscript-Fonts für SVG	12
1.3	Benutzung	14
1.3.1	Treiber	14
1.3.2	Programm-Start	27
1.3.3	Optionen	27
1.4	Konfigurationsmechanismen	29
1.4.1	Konfigurationsdatei	29
1.4.2	Kommandozeilenargumente	29
1.4.3	Steuerkommentare	31
1.5	Konfigurationseinträge	33
1.5.1	Für Alle Treiber	33
1.5.2	MetaPost	41
1.5.3	PS/EPS	43
1.5.4	PDF	46
1.5.5	TeX	47
1.5.6	SVG	48
1.6	Praktische Tipps	51
1.6.1	Hintergrund-Rechteck	51
1.6.2	Umgang mit Füllmustern	52
1.6.3	Eingebundene Bilder	53
1.6.4	SVG-Bilder	54
1.7	Bitmap Images	55
1.7.1	Übersicht	55
1.7.2	Details für die verschiedenen Konvertierungsmöglichkeiten	57
2	Technische Details	59

Inhaltsverzeichnis

2.1	Mathematische Aspekte	59
2.1.1	X-Splines und Bezier-Splines	59
2.1.2	Kreisbögen	69
2.1.3	Pfeilspitzen	71
2.2	Notizen	78
2.2.1	Font-Handling	78

1 Nutzer-Handbuch

1.1 Überblick

1.1.1 Verwendungszweck

Das Programm `fig2vect` wandelt `*.fig`-Dateien – die mit `XFig`, `jFig` oder `WinFig` erzeugt wurden – in andere vektorbasierte Dateiformate um, siehe Abb. 1.1.

Mögliche Ausgabeformate sind METAPOST, PS/EPS, PDF, $\text{T}_{\text{E}}\text{X}$ und SVG. Die Ausga-

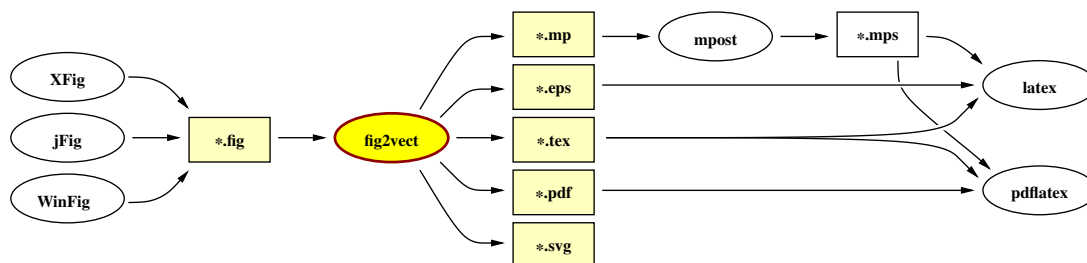


Abbildung 1.1: Verwendung des Programmes

beformate METAPOST, PS/EPS, PDF und $\text{T}_{\text{E}}\text{X}$ sind für die Verwendung mit $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}/\text{dvips}$ bzw. `pdf $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$` vorgesehen, SVG kann genutzt werden, um Graphiken per WWW zu publizieren.

1.1.2 Unterschiede zu Transfig

Hinweis: Alle Aussagen über die Software „transfig“ beziehen sich auf Version 3.2.4.

Sowohl transfig als auch fig2vect können genutzt werden, um *.fig-Dateien in andere Formate zu konvertieren. Zwischen beiden Programmen besteht eine Reihe von Unterschieden:

- **Unterschiedliche Zielgruppen**
Transfig (fig2dev) ist ein generelles Umwandlungsprogramm von Fig nach anderen Formaten und bietet eine große Anzahl Ausgabeformate an. Für eingebundene Images wird eine große Anzahl an Dateiformaten unterstützt.
Fig2vect wendet sich an L^AT_EX-Nutzer und bietet hauptsächlich solche Ausgabeformate an, die in diesem Umfeld genutzt werden. Für eingebundene Images können PNG-, JPEG- und NETPBM-Bilder verwendet werden; der EPS-Ausgabetreiber erlaubt zusätzlich die Einbindung von PS/EPS-Bildern, wenn diese eine „%%BoundingBox“-Zeile enthalten.
- **Interne Programmstruktur und Erweiterbarkeit**
Sowohl transfig als auch fig2vect können um zusätzliche Treiber erweitert werden. Ein transfig-Ausgabetreiber besteht dabei aus einem Satz von Funktionen, die jeweils nur die Daten für ein Graphikelement „zu sehen“ bekommen.
Ein fig2vect-Ausgabetreiber besteht aus einer Funktion, die Zugriff auf die gesamte Zeichnung erhält. Der Container mit den Graphikelementen kann beliebig oft durchmustert werden, z.B. um in einem ersten Pass Informationen zu sammeln und in einem zweiten Pass die Ausgabe vorzunehmen.
- **Füllmuster mit Vektorgraphik-Operationen realisiert.**
In fig2vect werden Füllmuster mit Vektorgraphik-Operationen realisiert anstelle von Bitmaps.
- **Spline-Behandlung**
X-Splines der Fig-Datei werden von transfig gleich beim Einlesen in Polylinien umgewandelt.
In fig2vect werden X-Splines durch Bezier-Splines angenähert, dabei wird jedes X-Spline-Segment durch ein oder mehrere Bezier-Spline-Segmente approximiert.
- **Bildabmessungen**
Einige transfig-Treiber verwenden die Papiergröße aus dem Kopfbereich der Fig-Datei, um die Bildabmessungen zu bestimmen.
Fig2vect bezieht die Bildabmessungen aus den Graphikelementen (mit Ausnahme der Texte).
- **Pfeilspitzen**
Für Linien mit Pfeilspitzen betrachtet fig2vect den angegebenen Endpunkt der

Linie als Endpunkt der Pfeilspitze und kürzt bei Bedarf die Linie so, daß die Pfeilspitze in gewünschten Punkt endet.

Liniendicken für Pfeilspitzen werden ignoriert, eine Pfeilspitze wird immer mit derselben Liniendicke gezeichnet wie die Linie selbst.

- **Konfiguration**

Transfig wird über Kommandozeilenparameter gesteuert. Fig2vect benutzt eine Konfigurationsdatei, es wird nur ein Kommandozeilenparameter benötigt, um eine bestimmte Konfiguration zu aktivieren.

- **Keine automatische Bildrotation**

Ist bei eingefügten Bildern der erste angegebene Polygon-Punkt nicht der linke obere Punkt, verhalten sich XFig und jFig unterschiedlich: XFig rotiert das Bild um 90° bzw. ein Vielfaches, jFig führt keine Rotation durch. Fig2vect verhält sich hier wie jFig und nimmt ebenfalls keine Rotation vor.

Um ein Bild rotiert einzubinden, wird empfohlen, das Bild in einem Graphikprogramm zu bearbeiten und dort die Rotation vorzunehmen. Das rotierte Bild wird dann in die Fig-Datei eingebunden und dabei zuerst der linke obere Eckpunkt des Bildbereiches angegeben. Damit ist die Fig-Datei portabel und kann mit XFig und jFig bearbeitet und mit transfig und fig2vect konvertiert werden.

1.1.3 Lizenzbestimmungen

Die Software unterliegt einer Lizenz im BSD-Stil:

- Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
 - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
 - Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
 - Neither the name of the Dirk Krause nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
- This software is provided by the copyright holders and contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed.
In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

1.2 Installation

1.2.1 Installation unter Unix/Linux

Voraussetzungen

Es wird eine Reihe von Bibliotheken benötigt, um fig2vect zu installieren:

- zlib
- libzip2
- dklibs
- libpng
- jpeglib
- netpbm

In der fig2vect FAQ¹ sind die Bibliotheken ausführlich und mit Bezugsquellen gelistet. Außer den dklibs sollten alle Bibliotheken über das Paketmanagement der Unix/Linux-Distribution installierbar sein. Zu beachten ist, daß nicht nur die Bibliotheken selbst als Binärdateien benötigt werden, sondern auch die zugehörigen Headerdateien. Bei manchen Distributionen befinden sich diese in separaten Paketen, die dann z.B. „zlib developer support“ heißen.

Download

Die Software kann von der Projekt-Homepage² bezogen werden.

Installationsvorgang unter Unix/Linux

Die üblichen Befehle

```
1 ./configure
2 make
3 make install
```

konfigurieren und erstellen die Software und nehmen die Installation vor.

¹<http://fig2vect.sourceforge.net/faq.html#h5uwhich>

²<http://fig2vect.sourceforge.net>

1 Nutzer-Handbuch

RPM vorbereiten

Um eine RPM-Erstellung – bzw. die Erstellung von Paketen für andere Softwaremanagement-Systeme – vorzubereiten, müssen die Dateien bei der Installation nicht in das Zielverzeichnis kopiert werden sondern in ein Package-Verzeichnis, z.B. „/tmp/packages/fig2vect“ unterhalb dessen dann die übliche Struktur „.../usr/local/...“ zu finden ist.

Dies kann mit

```
1 ./configure
2 make
3 make pp=/tmp/packages/fig2vect install
```

bewerkstelligt werden („pp“ bedeutet hier „package prefix“).

1.2.2 Installation unter Windows

Vom dklibs-Projekt auf Sourceforge³ werden u.a. Windows-Binaries für die dklibs-Libraries und darauf basierende Software angeboten.

Die Datei „dklibs-win32-*-user.exe“ installiert die Software und die Dokumentation, die „dklibs-win32-*-dev.exe“ darüber hinaus noch die Quellen sowohl für die dklibs-Libraries, die darauf aufbauende Software und alle anderen genutzten Bibliotheken. Die „dklibs-win32-*-dev.exe“-Datei ist wesentlich größer als die „dklibs-win32-*-user.exe“-Datei.

³<http://sourceforge.net/projects/dklibs>

1.2.3 Installation der Ghostscript-Fonts für SVG

Dieser Schritt muß nur ausgeführt werden, wenn SVG-Dateien erzeugt werden sollen und die in der Fig-Datei festgelegten PostScript-Fonts unbedingt übernommen werden sollen. Der SVG-Standard definiert, daß SVG-Viewer Texte darstellen können müssen, die die Angaben „mit Serifen“, „ohne Serifen“, „monospaced“, „normale Dicke“, „fett“, „aufrecht“ und „kursiv“ enthalten. Standardmäßig setzt fig2vect die Fontangaben aus der Fig-Datei in entsprechende SVG-Fontangaben um.

Wollen Sie mit dem Aussehen der erzeugten SVG-Datei möglichst nahe am Original bleiben, müssen Sie dem SVG-Viewer die verwendeten Fonts als SVG-Fontdateien bereitstellen. Hierzu müssen Sie zunächst die SVG-Fontdateien erzeugen. Sie benötigen:

- Das Programm FontForge⁴
- Die Ghostscript-Fonts
- Die Quellen für die Ghostscript-Fonts⁵, Datei „urw-fonts-1.0.7pre40-src.tar.bz2“ bzw. Datei mit höherer Versionsnummer.
- Die Dateien „comment.txt“, „crfonts.csh“ und „sfd2svg.pe“ aus dem scripts-Verzeichnis des fig2vect-Archives.

Gehen Sie folgendermaßen vor:

- Installieren Sie die Software FontForge.
- Erzeugen Sie ein temporäres Verzeichnis, z.B. „/tmp/xxx“.
- Entpacken Sie die Archive mit den Ghostscript-Fonts und deren Quellen in das temporäre Verzeichnis, so daß sich die *.pfb- und *.sfd-Dateien in diesem Verzeichnis befinden.
- Kopieren Sie die drei Dateien aus dem scripts-Verzeichnis ebenfalls in das temporäre Verzeichnis.
- Wechseln Sie in das temporäre Verzeichnis.
- Rufen Sie das Script crfonts.csh mit

```
1 ./crfonts.csh
```

auf. Damit werden die *.svg-Dateien erstellt.

⁴<http://sourceforge.net/projects/fontforge>

⁵<ftp://ftp.gnome.ru/fonts/sources>

- Bearbeiten Sie alle *.svg-Dateien mit einem Texteditor. Fügen Sie in jeder Datei zwischen dem „<metadata>...</metadata>“- und dem „<defs>...</defs>“-Bereich den kompletten Inhalt der Datei „comment.txt“ ein. Im „<metadata>...</metadata>“-Bereich finden Sie unter der Zeile „Created by FontForge...“ eine Zeile „By...“, die Ihren Namen bzw. Loginnamen enthält. Diese Zeile sollten Sie löschen (ebenfalls in jeder *.svg-Datei).
- Erstellen Sie ein Zielverzeichnis, in das Sie die SVG-Fonts installieren möchten. Kopieren Sie folgende Dateien in das Zielverzeichnis:
 - alle *.svg-Dateien aus dem temporären Verzeichnis,
 - die Archive mit den Ghostscript-Fonts und den Quellen und
 - die Dateien aus dem „scripts“-Verzeichnis der fig2vect-Distribution.

Hintergrund: Die Ghostscript-Fonts wurden vom Hersteller URW unter der GNU General Public License (GPL) bereitgestellt. Den vollen Wortlaut (englischsprachig) finden Sie in der Datei COPYING der Ghostscript-Font-Distribution. Diese Lizenz beinhaltet u.a., daß jeder Nutzer von GPL-lizenzierter Software auch das Recht hat, die Quellen der Software zu erhalten und für seine Zwecke anzupassen. Wer GPL-lizenzierte Software für andere Nutzer verfügbar macht, ist auch dafür verantwortlich, diesen Nutzern den Zugang zu den Quellen zu ermöglichen.

Stellen Sie – z.B. auf einem Multi-User-Computersystem oder auf einer Web-Site – anderen Nutzern die SVG-Fonts zur Verfügung, haben Sie allen entsprechenden Nutzern auch den Zugang zu den Quellen zu ermöglichen, die zur Herstellung dieser Fonts benötigt werden. Der Zugang zu den Quellen darf dabei für den Nutzer nicht komplizierter sein als der Zugang zur Software selbst. Zur Erfüllung dieser Forderung bietet es sich an, die Quellen gemeinsam mit den Fonts im selben Verzeichnis zu lagern.

Wenn Sie die Fonts in SVG-Dateien Ihres Webauftrittes verwenden wollen, sollten Sie einen Ordner, z.B. „<http://www.my-organisation.com/gs-fonts-svg>“ erstellen und dort sowohl die *.svg-Dateien mit den SVG-Fonts als auch die Archive mit den Ghostscript-Fonts und deren Quellen und die Dateien aus dem „scripts“-Verzeichnis der fig2vect-Distribution vorhalten.

Alle Aussagen in den vorangegangenen Absätzen in diesem Abschnitt über die GNU General Public License und die sich daraus für Sie ergebenden Rechte und Pflichten sind lediglich Interpretationen, die auf meinem aktuellen Kenntnisstand beruhen und durchaus fehlerhaft sein können. Rechtlich verbindlich ist ausschließlich der englischsprachige Originaltext mit den Lizenzbestimmungen, den Sie in der Datei COPYING der Ghostscript-Font-Distribution finden bzw. bei der Free Software Foundation erhalten können.

1.3 Benutzung

1.3.1 Treiber

Treiber-Auswahl

Die Auswahl des Treibers hängt davon ab,

- für welchen Verwendungszweck eine Ausgabedatei produziert werden soll und
- was in der Fig-Datei enthalten ist.

Wollen Sie eine Graphik für eine Webpräsenz erstellen, bietet sich das SVG-Format an. Besucher Ihrer Website benötigen u.U. ein spezielles Plug-In, um SVG-Dateien betrachten zu können.

Wollen Sie die Graphik in \LaTeX / $\text{pdf}\text{\LaTeX}$ -Dokumenten einsetzen, wird der MetaPost-Ausgabetreiber empfohlen, da die MetaPost-Ausgabe sowohl mit \LaTeX / dvips als auch mit $\text{pdf}\text{\LaTeX}$ verwendbar ist. Allerdings kann der MetaPost-Treiber nicht genutzt werden, wenn die Fig-Datei eingebundene Images enthält, in diesem Fall können Sie folgende Treiber benutzen:

- EPS, wenn Sie mit \LaTeX / dvips arbeiten. Enthält die Fig-Datei `special text`, müssen Sie den EPS-Treiber mit dem \TeX -Treiber kombinieren.
- PDF, wenn Sie mit $\text{pdf}\text{\LaTeX}$ arbeiten. Enthält die Fig-Datei `Text`, müssen Sie den PDF-Treiber mit dem \TeX -Treiber kombinieren, da der PDF-Treiber nur `non-special-Text` in den 14 PDF-Standardfonts linksbündig darstellen kann.

MetaPost-Treiber

Bei der Arbeit mit dem MetaPost-Treiber erstellen Sie zunächst mit `fig2vect` eine Datei `*.mp`. Diese Datei wird dann durch das Programm `mpost` (auf manchen Systemen auch einfach nur `mp`) in eine PostScript-Datei `*.0` umgewandelt. Diese Datei sollten Sie umbenennen zu `*.mps`.

Der Vorteil des MetaPost-Treiber ist, daß die `*.mps`-Datei sowohl mit $\text{\LaTeX}/\text{dvips}$ als auch mit $\text{pdf}\text{\LaTeX}$ verwendet werden kann. Nachteilig ist, daß MetaPost keine eingebundenen Images unterstützt. Mit MetaPost können sowohl „normaler“ Text als auch `special text` benutzt werden. Zum Setzen von `special text` wird \LaTeX genutzt, für normalen Text kann gewählt werden, ob der Textsatz mit \LaTeX oder durch einfache PostScript-Texte erfolgen soll.

Unter Unix/Linux verwenden Sie folgende Kommandofolge, um aus einer Datei `example.fig` eine Datei `example.mps` zu erstellen:

```
1 fig2vect -lmp example.fig example.mp
2 mpost --tex=latex example
3 mv example.0 example.mps
```

Die Kommandofolge in einer Windows-Eingabeaufforderung lautet:

```
1 fig2vect -lmp example.fig example.mp
2 mpost --tex=latex example
3 xcopy example.0 example.mps
4 del example.0
```

In der \LaTeX -Quelle binden Sie das Bild folgendermaßen ein:

```
1 \begin{figure}
2 {\centering
3 \includegraphics{example.mps}
4 \caption{Dies ist ein Bild}
5 \label{fig:example}
6 }
7 \end{figure}
```

Hinweis: Auf manchen Systemen wird muss MetaPost mit „`mpost --tex=latex ...`“ gestartet werden, auf anderen Systemen mit „`mpost -tex=latex ...`“.

Mit „`mpost --help`“ können Sie einen Hilfetext zu MetaPost anzeigen lassen.

PS/EPS-Treiber

Mit dem Kommando

```
1 fig2vect -leps example.fig example.eps
```

bzw.

```
1 fig2vect -lps example.fig example.ps
```

konvertieren Sie die Datei `example.fig` in eine Datei `example.eps` bzw. `example.ps`. Diese Datei können Sie mit

```
1 \begin{figure}  
2 {\centering  
3 \includegraphics{example}  
4 \caption{Dies ist ein Bild}  
5 \label{fig:example}  
6 }  
7 \end{figure}
```

in Ihre \LaTeX -Quelle einbinden.

Der EPS-Treiber verarbeitet auch eingebundene Images der Typen PNG, JPEG, NET-PBM und PS/EPS. Enthält die Fig-Datei `special text`, muß der EPS-Treiber mit dem \TeX -Treiber kombiniert werden, siehe Abschnitt 1.3.1 auf Seite 18.

PDF-Treiber

Mit dem Kommando

```
1 fig2vect -lpdf example.fig example.pdf
```

wird die Datei `example.fig` in eine PDF-Datei `example.pdf` konvertiert. Diese kann in der pdfL^AT_EX-Quelle mit

```
1 \begin{figure}
2 {\centering
3 \includegraphics{example}
4 \caption{Das ist ein Bild}
5 \label{fig:example}
6 }
7 \end{figure}
```

eingebunden werden. Wie man sieht kann dieselbe Quelle mit pdfL^AT_EX genutzt werden (dieser bindet die *.pdf-Datei ein) wie mit L^AT_EX (bindet die *.eps-Datei ein).

Ein Vorteil des PDF-Treiber ist, daß Alpha-Kanäle in eingebundenen PNG-Dateien in die PDF-Ausgabe übernommen werden. Damit können teilweise transparente Graphiken noch über Hintergründe (z.B. Farbverläufe) gelegt werden.

Nachteil des PDF-Treibers ist, daß dieser nur normalen linksbündigen Text selbst setzen kann, wobei alle Schriften durch die 14 Standard-PDF-Schriften substituiert werden. Werden mehr Textsatz-Features oder gar special text benötigt, muß der PDF-Treiber mit dem T_EX-Treiber kombiniert werden, siehe Abschnitt 1.3.1 auf Seite 21.

Kombination von EPS- und TeX-Treiber

In der Kombination von EPS- und TeX-Treiber übernimmt der TeX-Treiber den Textsatz von special text und wahlweise auch von normalem Text. Der EPS-Treiber kümmert sich um die Darstellung aller anderen Graphikelemente. Mit den Kommandos

```
1 fig2vect -leps .tex example .fig example .eps
2 fig2vect -ltex example .fig example .tex
```

werden zwei Dateien erzeugt. In der L^AT_EX-Quelle wird die Graphik mit

```
1 \begin{ figure }
2 { \centering
3 \input{example .tex }
4 \caption{Dies ist ein Bild }
5 \label{fig : example }
6 }
7 \end{ figure }
```

eingebunden. Falls eine Skalierung erforderlich ist, können Sie diese mit einer „resize-box“ vornehmen:

```
1 \begin{ figure }
2 { \centering
3 \resizebox { \linewidth } { ! } { \input{ example .tex } }
4 \caption{Dies ist ein Bild }
5 \label{fig : example }
6 }
7 \end{ figure }
```

vornehmen. Das erste Argument von `resizebox` legt die Bildbreite fest, das zweite Argument die Höhe der Box. Im Beispiel wurde mit dem Ausrufezeichen festgelegt, daß eine gleichmäßige Skalierung horizontal und vertikal erfolgen soll.

Abb. 1.2 auf der nächsten Seite zeigt das Zusammenspiel der Ausgaben von EPS- und TeX-Treiber:

- Die Datei `example.tex` – die von der L^AT_EX-Quelle mit einem `\input`-Befehl eingeschlossen wird – baut zunächst eine `picture`-Umgebung auf, für die jedoch keinerlei Platz auf der Seite reserviert wird. An den Ursprung dieser `picture`-Umgebung wird die Datei `example.eps` eingefügt.
- Eine zweite `picture`-Umgebung mit genau den Abmessungen der EPS-Datei wird erzeugt. Da für die erste `picture`-Datei kein Platz reserviert wurde, liegt sie exakt über dem eingebundenen Bild. In dieser `picture`-Umgebung werden die Labels für die special texts und wahlweise auch für normale Texte gesetzt.

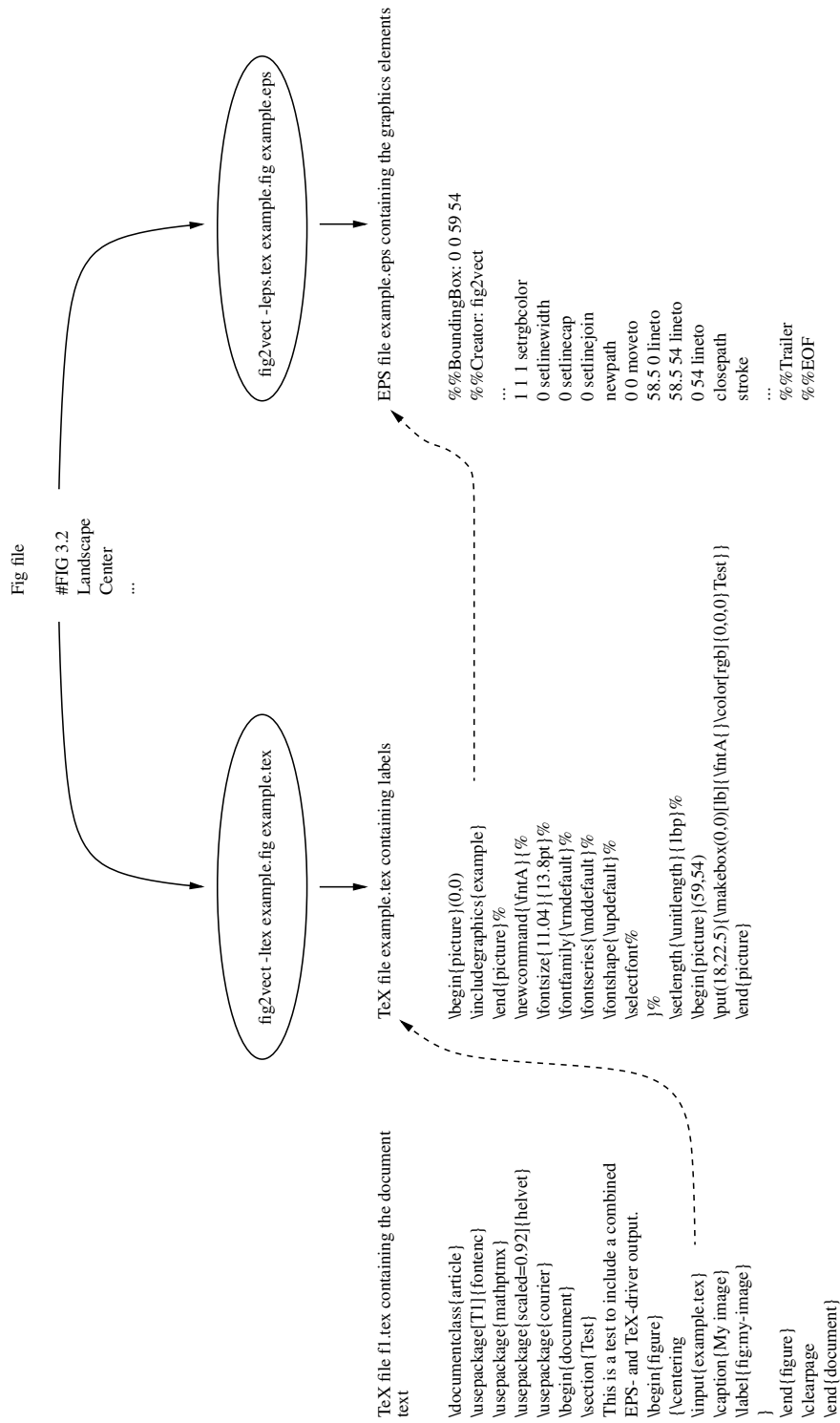


Abbildung 1.2: Kombination von EPS- und TeX-Treiber

1 Nutzer-Handbuch

Wollen Sie normalen Text auch vom $\text{T}_{\text{E}}\text{X}$ -Treiber setzen lassen, müssen die entsprechenden Einträge in der Konfigurationsdatei (siehe Abschnitt 1.4.1 auf Seite 29) folgendermaßen aussehen:

```
1 [ eps . tex ]
2 normal text      =      handling : tex , font : similar , size : fig , mbox
3 special text     =      font : similar , size : fig , mbox
4 skip all texts   =      yes
5 [ tex ]
6 normal text      =      handling : tex , font : similar , size : fig , mbox
7 special text     =      font : similar , size : fig , mbox
```

Für den EPS-Treiber legt „handling:tex“ im „[eps.tex]“-Abschnitt fest, daß der Textsatz durch $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ erfolgen soll, was der EPS-Treiber nicht selbst veranlaßt. Daher ignoriert der EPS-Treiber normale Texte. Für den $\text{T}_{\text{E}}\text{X}$ -Treiber konfiguriert „handling:tex“ im „[tex]“-Abschnitt, daß normaler Text mit berücksichtigt werden muß.

Soll der Textsatz für normalen Text durch den EPS-Treiber erfolgen, müssen die Abschnitte wie folgt geändert werden:

```
1 [ eps . tex ]
2 normal text      =      handling : none , font : fig , size : fig
3 special text     =      font : similar , size : fig , mbox
4 [ tex ]
5 normal text      =      handling : none
6 special text     =      font : similar , size : fig , mbox
```

Das „handling:none“ im „[eps.tex]“-Abschnitt besagt, daß normaler Text keine Sonderbehandlung bekommt, der EPS-Treiber diesen als normale PostScript-Labels ausgibt. Das „handling:none“ im „[tex]“-Abschnitt stellt für den $\text{T}_{\text{E}}\text{X}$ -Treiber ein, daß normaler Text ignoriert wird.

Kombination von PDF- und TeX-Treiber

In der Kombination von EPS- und TeX-Treiber übernimmt der TeX-Treiber den Textsatz von special text und wahlweise auch von normalem Text. Der PDF-Treiber kümmert sich um die Darstellung aller anderen Graphikelemente. Mit den Kommandos

```
1 fig2vect -lpdf.tex example.fig example.pdf
2 fig2vect -ltex example.fig example.tex
```

werden zwei Dateien erzeugt. In der L^AT_EX-Quelle wird die Graphik mit

```
1 \begin{figure}
2   {\centering
3   \input{example.tex}
4   \caption{Dies ist ein Bild}
5   \label{fig:example}
6   }
7 \end{figure}
```

eingebunden. Falls eine Skalierung erforderlich ist, können Sie diese mit einer „resize-box“ vornehmen:

```
1 \begin{figure}
2   {\centering
3   \resizebox{\linewidth}{!}{\input{example.tex}}
4   \caption{Dies ist ein Bild}
5   \label{fig:example}
6   }
7 \end{figure}
```

vornehmen. Abb. 1.3 auf der nächsten Seite zeigt das Zusammenspiel der Ausgaben von PDF- und TeX-Treiber:

- Die Datei `example.tex` – die von der L^AT_EX-Quelle mit einem `\input`-Befehl eingeschlossen wird – baut zunächst eine `picture`-Umgebung auf, für die jedoch keinerlei Platz auf der Seite reserviert wird. An den Ursprung dieser `picture`-Umgebung wird die Datei `example.pdf` eingefügt.
- Eine zweite `picture`-Umgebung mit genau den Abmessungen der PDF-Datei wird erzeugt. Da für die erste `picture`-Datei kein Platz reserviert wurde, liegt sie exakt über dem eingebundenen Bild. In dieser `picture`-Umgebung werden die Labels für die special texts und wahlweise auch für normale Texte gesetzt.

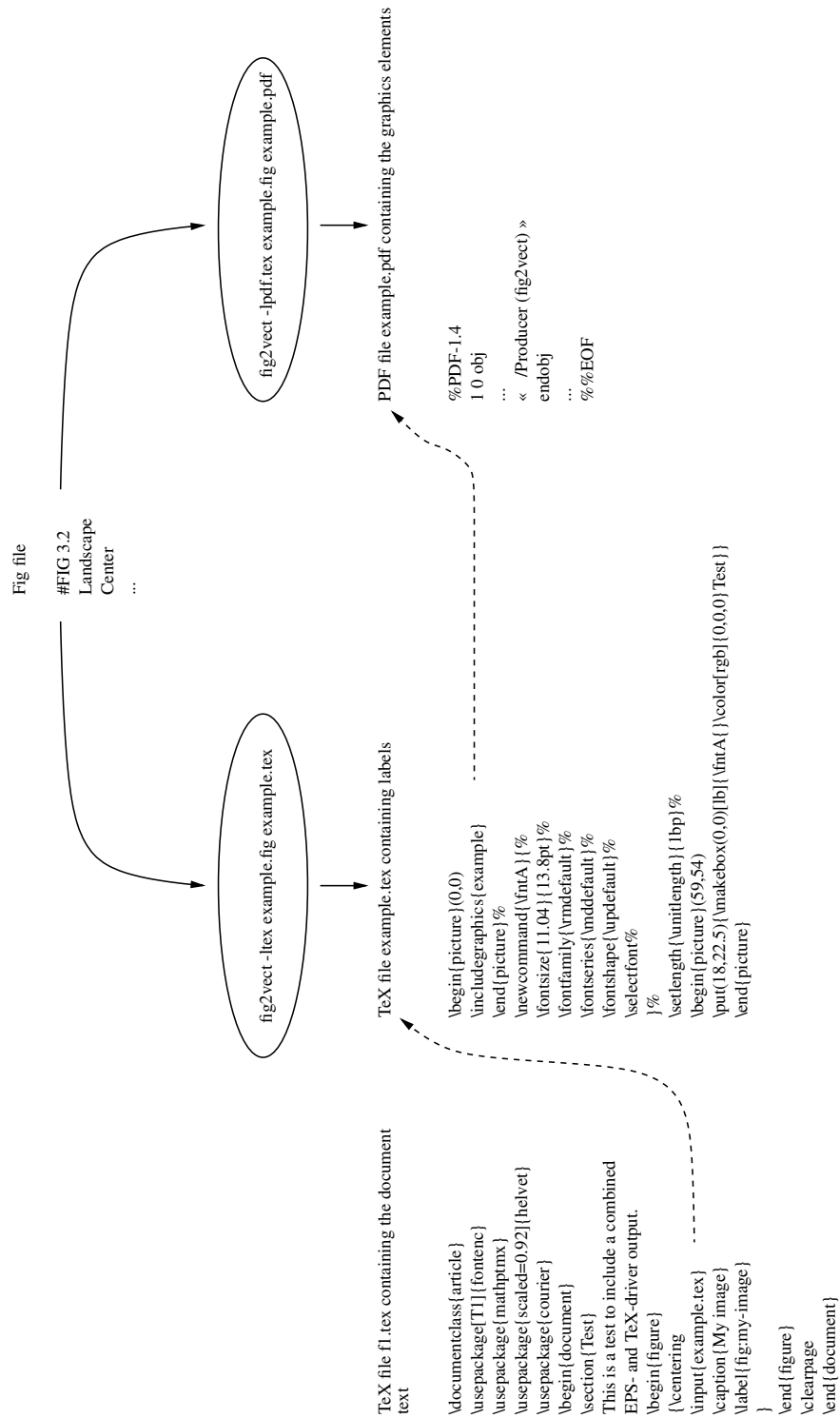


Abbildung 1.3: Kombination von PDF- und TeX-Treiber

Wollen Sie normalen Text auch vom $\text{T}_{\text{E}}\text{X}$ -Treiber setzen lassen, müssen die entsprechenden Einträge in der Konfigurationsdatei (siehe Abschnitt 1.4.1 auf Seite 29) folgendermaßen aussehen:

```

1 [ pdf . tex ]
2 normal text      =      handling : tex , font : similar , size : fig , mbox
3 special text     =      font : similar , size : fig , mbox
4 skip all texts   =      yes
5 [ tex ]
6 normal text      =      handling : tex , font : similar , size : fig , mbox
7 special text     =      font : similar , size : fig , mbox

```

Für den PDF-Treiber legt „handling:tex“ im „[pdf.tex]“-Abschnitt fest, daß der Textsatz durch $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ erfolgen soll, was der PDF-Treiber nicht selbst veranlaßt. Daher ignoriert der PDF-Treiber normale Texte. Für den $\text{T}_{\text{E}}\text{X}$ -Treiber konfiguriert „handling:tex“ im „[tex]“-Abschnitt, daß normaler Text mit berücksichtigt werden muß.

Soll der Textsatz für normalen Text durch den PDF-Treiber erfolgen, müssen die Abschnitte wie folgt geändert werden:

```

1 [ pdf . tex ]
2 normal text      =      handling : none , font : fig , size : fig
3 special text     =      font : similar , size : fig , mbox
4 [ tex ]
5 normal text      =      handling : none
6 special text     =      font : similar , size : fig , mbox

```

Das „handling:none“ im „[pdf.tex]“-Abschnitt besagt, daß normaler Text keine Sonderbehandlung bekommt, der PDF-Treiber diesen als normale PDF-Labels ausgibt. Das „handling:none“ im „[tex]“-Abschnitt stellt für den $\text{T}_{\text{E}}\text{X}$ -Treiber ein, daß normaler Text ignoriert wird.

Hinweis: Die Textausgabemöglichkeiten des PDF-Treibers sind begrenzt (nur normaler, linksbündiger Text in den 14 PDF-Standardschriften), daher sollte auch normaler Text durch den $\text{T}_{\text{E}}\text{X}$ -Treiber verarbeitet werden.

Eigenständige PDF-Dateien mit PDF- und TeX-Treiber

Im vorangegangenen Abschnitt wurde gezeigt, wie mit der Kombination von PDF- und TeX-Treiber ein Fig-Bild in ein L^AT_EX-Dokument eingebunden werden kann.

In diesem Abschnitt wird gezeigt, wie man eine PDF-Version eines Fig-Bildes erstellt, wobei der Textsatz pdfL^AT_EX überlassen wird.

An diesem Vorgang sind folgende Dateien beteiligt:

- die PDF-Version des Bildes ohne Texte,
- die TeX-Datei, die diese PDF-Version einbindet und dann die Text-Labels darüberlegt und
- die von pdfL^AT_EX erzeugte PDF-Datei.

Wichtig ist hierbei, daß die beiden PDF-Dateien unterschiedliche Namen haben müssen. Es bietet sich an, entweder den Namen der Zielfile aus dem Basisnamen der Fig-Datei zu bilden, indem ein „s“ (für standalone) angehängt wird

```
1 fig2vect -l pdf.tex example.fig example.pdf
2 fig2vect -l tex.full -o preamble.file =... example.fig examples.tex
3 pdflatex examples.tex
```

oder den Namen der PDF-Datei ohne Texte durch Anhängen eines „i“ (für incomplete) zu bilden. In diesem Fall muß dem TeX-Treiber der Dateiname mitgeteilt werden, der in der \includegraphics-Anweisung verwendet werden soll (standardmäßig wird der Basisname der Quelldatei eingetragen).

```
1 fig2vect -l pdf.tex example.fig examplei.pdf
2 fig2vect -l tex.full -i examplei -o preamble.file =... \
3 example.fig example.tex
4 pdflatex example
```

Kombination von MetaPost- und TeX-Treiber

In der Kombination von MetaPost- und TeX-Treiber übernimmt der TeX-Treiber den Textsatz für alle Texte, der MetaPost-Treiber die Darstellung aller anderen Graphikelemente. Mit den Kommandos

```
1 fig2vect -lmp.tex example.fig example.mp
2 mpost --tex=latex example
3 mv example.0 example.mps
4 fig2vect -ltex -i example.mps example.fig example.tex
```

werden example.mps und example.tex erstellt, mit

```
1 \begin{figure}
2 {\centering
3 \input{example.tex}
4 \caption{Dies ist ein Bild}
5 \label{fig:example}
6 }
7 \end{figure}
```

wird das Bild eingebunden. Eine Skalierung mit „resizebox“ ist möglich.

MetaPost kann zwar sowohl normalen Text als auch special text verarbeiten, die Erstellung der *.mpx-Datei, der L^AT_EX-Lauf und die Verarbeitung der *.dvi-Datei durch MetaPost sind allerdings zeitaufwendig, insbesondere bei einer größeren Anzahl an Bildern. Dieser Zeitaufwand kann durch die Kombination von MetaPost- und TeX-Treiber reduziert werden.

SVG-Treiber

Mit dem Kommando

```
1 fig2vect -lsvg example.fig example.svg
```

wird die Datei example.fig in die Datei example.svg konvertiert. Diese kann z.B. mit

```
1 <noembed>  
2   
3 </noembed>  
4 <embed src="example.svg" type="image/svg+xml" width="840" height="220">
```

in eine HTML-Seite eingebunden werden.

Der SVG-Treiber kann keinen special text verarbeiten und ignoriert das Flipping von eingebundenen Images.

1.3.2 Programm-Start

Mit

fig2vect Optionen Verzeichnis
fig2vect Optionen Eingabedatei Ausgabedatei

wird das Programm gestartet.

Bei der Verwendung des MetaPost-Ausgabe-Treibers können auch

fig2vect Optionen
fig2vect Optionen Eingabedatei

verwendet werden.

Wird kein Dateiname angegeben, arbeitet das Programm als Filter, d.h. die Standardeingabe wird verarbeitet und das Ergebnis auf die Standardausgabe ausgegeben.

Wird ein Verzeichnis angegeben, wird das Verzeichnis nach „*.fig“-Dateien durchsucht, diese werden zu gleichnamigen „*.mp“-Dateien bzw. „*.eps“-, „*.pdf“, „*.tex“- oder „*.svg“-Dateien umgewandelt. Hierbei kann eine make-Verarbeitungsweise aktiviert werden, dann werden die Zeitpunkte der letzten Änderung an Quell- und Zieldatei geprüft, eine Umwandlung erfolgt nur, wenn notwendig.

1.3.3 Optionen

Programm-Optionen

- *-l Konfiguration*
wählt eine bestimmte Konfiguration aus der Konfigurationsdatei aus.
- *-o key=value*
überschreibt einen Eintrag der Konfigurationsdatei.
- *-m*
aktiviert make-Verarbeitungsweise.
- *-m-*
deaktiviert make-Verarbeitungsweise.
- *-a*
wählt automatisch einen passenden Namen für die Ausgabedatei, falls keiner angegeben wurde. Der Name basiert auf dem Dateinamen der Eingabedatei und dem gewählten Ausgabetreiber.

Hilfe und Versionsanzeige

- *-h*
--help

1 Nutzer-Handbuch

gibt einen Hilfetext aus.

- -v
--version
zeigt die Versionsnummer an.

Permanente Optionen

- -c ...
--configure ...
speichert permanente Konfigurationsoptionen.
- -u
--unconfigure
löscht permanente Konfigurationsoptionen.
- -C
--show-configuration
zeigt die permanenten Konfigurationsoptionen an.
- -r
--reset
deaktiviert die permanenten Konfigurationsoptionen für einen Programmlauf.

1.4 Konfigurationsmechanismen

1.4.1 Konfigurationsdatei

Dateiname

Standardmäßig wird die Konfigurationsdatei „\$prefix/etc/fig2vect/fig2vect.cfg“ bzw. „%WINDIR%\fig2vect\fig2vect.cfg“ genutzt.

Jeder Nutzer kann eine eigene Konfigurationsdatei anlegen, diese wird unter „\$HOME/.defaults/fig2vect.cfg“ bzw. „%USERPROFILE%\defaults\fig2vect.cfg“ gesucht. Die nutzereigene Konfigurationsdatei wird *anstelle* der systemweit gültigen Konfigurationsdatei verwendet. Um zu prüfen, an welchen Orten die Konfigurationsdatei gesucht wird und welche Datei verwendet wird, können Sie die Option „--/log/stderr/level=debug“ verwenden.

Aufbau der Konfigurationsdatei

Die Konfigurationsdatei besteht aus einzelnen Abschnitten, die jeweils eine Konfiguration beschreiben. Jeder Abschnitt wird durch den Konfigurations-Namen in eckigen Klammern eingeleitet, daran schließen sich Konfigurationszeilen an. Jede Konfigurationszeile enthält einen Eintrag, der aus einem Namen und einem Wert besteht, die durch ein Gleichheitszeichen getrennt sind. Der Name kann aus mehreren Teilen bestehen. Der Konfigurationsname wird gebildet, indem an den Treibernamen ein weiterer Name – durch Punkt getrennt – angehängt wird.

Um die einzelnen Konfigurationsabschnitte nicht zu lang werden zu lassen, wird ein Vererbungsmechanismus genutzt.

Wird beispielsweise

```
i  fig2vect -l mp.pdf test.fig test.mp
```

einggegeben (siehe Listing 1.1 auf der nächsten Seite), wird zunächst der allgemeine Konfigurationsabschnitt „[*]“ verarbeitet. Anschließend wird der Konfigurationsabschnitt verarbeitet, der dem Treibernamen entspricht, also „[mp]“. Die hier getroffenen Einstellungen überschreiben die bisher vorgenommenen Einstellungen. Zuletzt wird der Abschnitt „[mp.pdf]“ eingelesen, dabei werden wiederum die bisherigen Einstellungen überschrieben.

Bei der Vererbung ist zu beachten, daß ausschließlich der erste Punkt den Treibernamen vom restlichen Namen trennt. Weitere Punkte erzeugen keine zusätzlichen Vererbungsschritte.

1.4.2 Kommandozeilenargumente

Einzelne Konfigurationseinträge können auf der Kommandozeile überschrieben werden, die Syntax hierfür ist

Listing 1.1: Beispiel-Konfigurationsdatei

```
1 [*]
2 tex command           = latex
3 embed fonts          = no
4 normal text          = handling:none , font:fig , size:fig , mbox
5 special text         = handling:tex , font:fig , size:fig , mbox
6 web palette          = no
7 spline segments     = 8
8 verbose output       = yes
9 pattern repeat       = 4
10 remove zero borders = yes
11 lighten look        = yes
12 min iteration steps = 16
13 max iteration steps = 2048
14 arrowhead linejoin  = mitered
15 fill patterns       = contiguous
16
17 [mp]
18 normal text         = handling:tex , font:fig , size:fig , mbox
19 special text        = handling:tex , font:tex , size:tex , mbox
20 use metapost arrowheads = yes
21
22 [mp.ams]
23 latex font setup    = ams
24
25 [mp.pdf]
26 latex font setup    = pdf
27
28 [mp.unusual]
29 normal text         =          handling:none , font:fig , size:fig , mbox
30 special text        =          handling:tex , font:fig , size:fig , mbox
31 latex font setup    =          pdf
```

```
1 -o name=value
```

Um beispielsweise eine MetaPost-Datei ohne Verwendung von MetaPost-Pfeilspitzen zu erzeugen, wäre das Kommando

```
1 fig2vect -lmp -o use.metapost.arrowheads=no example.fig example.mp
```

Wie man sieht, werden die Namensbestandteile für den zu überschreibenden Konfigurationseintrag durch Punkte getrennt.

1.4.3 Steuerkommentare

Eine weitere Möglichkeit, Konfigurationseinträge zu überschreiben, sind Steuerkommentare in der Fig-Datei. Ein Steuerkommentar kann entweder für die Datei als Ganzes gelten – in diesem Fall muß er im globalen Kommentarbereich stehen – oder nur für ein Graphikobjekt – in diesem Fall muß er dem Graphikobjekt unmittelbar vorangehen. Ein Steuerkommentar ist eine Zeile mit folgenden Bestandteilen:

- zwei Rauten (Kommentarzeichen) zwischen denen sich optional Leerzeichen/Tabulatoren befinden dürfen,
- Liste der Treiber, für die der Steuerkommentar gültig ist,
- Doppelpunkt
- Konfigurationseintrag.

Der Beispieleintrag

```
1 ## pdf: tiled patterns = no
```

legt für den PDF-Treiber fest, daß nur eine große Pattern-Kachel verwendet werden soll. Mit

```
1 ## *: font scale factor = 0.95
```

wird für alle Treiber ein Skalierungsfaktor für Schriftgrößen festgelegt.

Soll ein Eintrag für mehrere Treiber gültig sein, werden die Treibernamen durch Kommata getrennt, z.B.

```
1 ## mp, pdf: fill pattern = none
```

Steuerkommentare werden eingefügt, indem Sie beim Bearbeiten eines Objektes in das Feld „Comments“ den Text, z.B.

```
1 # mp, pdf: fill pattern = none
```

einfügen. Zu beachten ist, daß ein Kommentarzeichen geschrieben werden muß, das zweite Kommentarzeichen wird durch XFig bzw. jFig vorangestellt, wenn die Datei gespeichert wird.

1 Nutzer-Handbuch

Steuerkommentare auf Dateiebene werden mit einem Texteditor manuell in die Datei eingefügt, sie müssen zwischen Zeile 8 (transparente Farbe) und 9 (Auflösung/Koordinatenursprung) eingefügt werden.

Die Datei `doc2.fig`⁶ kann als Beispiel dienen, hier wurden nachträglich Steuerkommentare für ECMA-Script (JavaScript) in SVG-Dateien eingefügt.

⁶<http://fig2vect.sourceforge.net/doc2.fig>

1.5 Konfigurationseinträge

1.5.1 Konfigurationseinträge für alle Treiber

Allgemeines Aussehen

- **lighten look** = *boolean:lighten*
legt fest, ob beim Konvertieren alle Liniendicken halbiert werden (wie es die Fig Format Specification vorschlägt) oder nicht.
- **web palette** = *boolean:web-palette*
legt fest, ob eine Web-optimierte Farbpalette zum Einsatz kommt oder alle Farben so belassen werden, wie in der Fig-Datei vorgegeben.
- **use cs setting** = *boolean:cs-setting*
legt fest, ob die Koordinatensystem-Angabe aus der Fig-Datei verwendet werden soll oder nicht. Die Fig-Spezifikation ist hier (für mich) nicht ganz eindeutig. Auf der einen Seite gibt es die Möglichkeit, den Koordinatenursprung links oben oder links unten zu wählen, an einer anderen Textstelle steht jedoch, daß der Koordinatenursprung immer links oben liegt. Der Standardwert „no“ für diese Option geht davon aus, daß der Koordinatenursprung immer links oben liegt.
- **verbose output** = *boolean:verbose*
legt fest, ob zusätzliche Kommentare in die Ausgabedatei geschrieben werden.
- **accept unknown paper size** = *boolean:accept*
legt fest, ob beliebige Namen für Papiergrößen verwendet werden dürfen („yes“ oder „no“). Da fig2vect die Bildgröße nicht aus dem Namen der Papiergröße bildet sondern aus den Graphikelementen berechnet, können beliebige Papiergrößen akzeptiert werden.
- **remove background rectangle** = *boolean:remove*
legt fest, daß ein Hintergrund-Rechteck nur für die Berechnung der Bildabmessungen verwendet wird, aber nicht mit gezeichnet wird. Ein Hintergrund-Rechteck kann genutzt werden, um die Bildabmessungen explizit festzulegen. Dies ist u.a. sinnvoll, wenn eine Fig-Datei auch Texte enthält, da diese nicht zur Berechnung der Bildabmessungen herangezogen werden.
Ein Hintergrund-Rechteck ist ein Polygon (Typ 2), Sub-Typ Box (2) in Layer 999. Die Strichfarbe muß auf -1 („Default“) bzw. 7 („weiß“) – siehe auch Option „background rectangle color“ – gesetzt sein, der Füll-Stil auf -1 (keine Füllung) und die Linienbreite auf 0. Die Füllfarbe ist egal.
- **background rectangle color** = *string:colordef*
erlaubt die Angabe, welche Strichfarbe ein Hintergrund-Rechteck hat. Die Standard-einstellung „default“ verlangt die Farbe -1 („Default“), der Wert „white“ verlangt



Abbildung 1.4: spline segments = 2 / 8

7 (weiß).

Diese Option ist insbesondere bei der Arbeit mit jFig interessant, da dort die Strichfarbe -1 nicht in den Objekteigenschaften eingestellt werden kann.

Sie können hier nur „default“ oder „white“ angeben, keine anderen Farben.

Zahlenangaben

- **color digits** = *integer:digits*
legt fest, wieviele Nachkommastellen in Farbangaben verwendet werden (Standard: 3).
- **position digits** = *integer:digits*
legt fest, wieviele Nachkommastellen in Koordinatenangaben verwendet werden (Standard: 2).
- **additional trigonometric digits** = *integer:digits*
legt fest, wieviele Nachkommastellen zu Koordinaten hinzukommen, wenn die Koordinaten durch trigonometrische Berechnungen zustande kamen (Standard: 3, somit insgesamt 5 Nachkommastellen).

Spline-Behandlung

- **spline segments** = *integer:segments*
legt fest, wieviele Bezier-Spline-Segmente benutzt werden, um jeweils ein X-Spline-Segment zu approximieren. Der Standardwert hierfür ist 2, eine Erhöhung führt zu einer besseren Qualität der Approximation. Erhöhungen über 8 hinaus führen in den meisten Fällen nicht mehr zu einer sichtbaren Qualitätserhöhung. Abb. 1.4 zeigt zwei geschlossene Splines (approximiert und interpoliert), es wurden 2 bzw. 8 Bezier-Spline-Segmente für jedes X-Spline-Segment benutzt.

Pfeilspitzen

- **arrowhead linejoin** = *string:linejoin*
legt fest, wie Ecken in Pfeilspitzen gezeichnet werden. Mögliche Werte sind „mit-



Abbildung 1.5: arrowhead linejoin = mitered / rounded / beveled

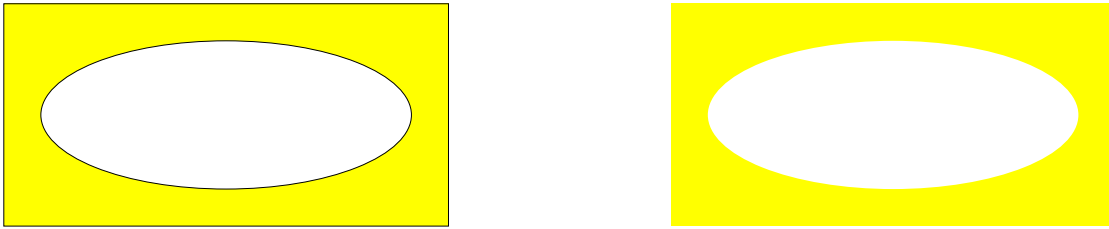


Abbildung 1.6: remove zero borders = no / yes

ered“ (Standard), „rounded“ oder „beveled“.

Abb. 1.5 zeigt die drei Varianten an zwei verschiedenen Pfeilspitzen.

- **min iteration steps** = *integer:number*

max iteration steps = *integer:number*

Werden Pfeilspitzen an offene Splines gelegt, muß das entsprechende Spline-Segment etwas gekürzt werden, damit die Linie nicht unter der Pfeilspitze hervorschaut. Für die Berechnung beim Kürzen wird ein Iterationsverfahren verwendet, die hier genannten Optionen legen die minimale bzw. maximale Anzahl an Iterationsdurchläufen fest. Wird die maximale Anzahl an Iterationsschritten ohne Erfolg durchlaufen, wird das Spline-Segment nicht gekürzt sondern stattdessen die Pfeilspitze entfernt.

Füllmuster, Füllungen und Strichmuster

- **remove zero border** = *boolean:remove*

legt fest, ob Umrandungen von gefüllten bzw. gemusterten Objekten entfallen, wenn als Liniendicke 0 angegeben ist (siehe Abb. 1.6).

- **fill patterns** = *boolean:filling*



Abbildung 1.7: fill patterns = yes / contiguous

1 Nutzer-Handbuch

legt fest, ob Füllmuster verwendet werden dürfen („yes“ oder „no“). Wird hier „contiguous“ angegeben, werden die Füllmuster so gelegt, daß gleiche Füllmuster verschiedener Objekte nahtlos ineinander übergehen (siehe Abb. 1.7 auf der vorherigen Seite).

- **pattern repeat** = *integer:distance*
legt die Patterngröße in $\frac{1}{80}$ Zoll fest (dieselbe Einheit, die Fig für die Linienbreite benutzt). Der Standardwert 4 sorgt für ein identisches Aussehen in XFig und der Ausgabedatei.
- **dashpattern dot length** = *string:length*
legt fest, ob als Länge für Punkte in Dashpattern 0 („0“) oder die Liniendicke („linewidth“, Standard) verwendet wird.

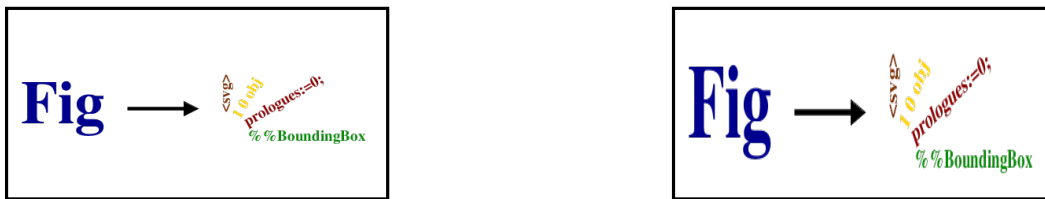


Abbildung 1.8: keep bitmap aspect ratio = yes / no

Eingebundene Bitmaps/Bilder

- keep bitmap aspect ratio = *boolean:keep***
 legt fest, ob beim Einfügen des Bildes das Höhe-Breite-Verhältnis beibehalten werden soll oder das Bitmap exakt in das vorgegebene Polygon eingepaßt wird (mit eventuellen Verzerrungen durch unterschiedliche Skalierung in x- und y-Richtung).
 Abb. 1.8 zeigt die Auswirkung dieser Option für eine Fig-Datei, in die eine PNG-Datei mit dem Fig2vect-Logo eingebunden wurde. Es wurde ein zusätzliches Rechteck hinterlegt, um die Polygon-Abmessungen zu zeigen, die für das eingebundene Bild reserviert wurden.
- remove bitmap border = *boolean:remove***
 legt fest, ob die Umrandung für Bilder entfernt werden soll. Mit


```
1 remove bitmap border = no
```

 wird eine Umrandung gezeichnet, mit


```
1 remove bitmap border = yes
```

 wird das Zeichnen verhindert (Standard).
 Abb. 1.9 auf der nächsten Seite zeigt die Auswirkung dieser Option, das hinterlegte Rechteck wurde entfernt.
 Über das XFig-GUI können die Linieneigenschaften der Umrandung nicht geändert werden, hierzu sind manuelle Änderungen an der Fig-Datei erforderlich.
- fill bitmap background = *boolean:fill***
 legt fest, ob das Polygon vor dem Zeichnen des Bitmaps noch mit der angegebenen Füllfarbe bzw. dem angegebenen Füllmuster gefüllt wird. Auch die Füllung des Bitmaps kann nicht über das XFig-GUI modifiziert werden, hier sind ebenfalls manuelle Änderungen der Fig-Datei erforderlich (z.B. Position 6: Farbe, Position 9: Sättigung). Abb. 1.10 auf der nächsten Seite zeigt die Auswirkung dieser Option, es wurde Farbe 11 (hellblau) und Sättigung 30 (starke Aufhellung) verwendet.



Abbildung 1.9: remove bitmap border = yes / no



Abbildung 1.10: fill bitmap background = no / yes

Textsatz

- **font scale factor** = *float:factor*
legt einen Skalierungsfaktor für alle Texte fest. Der Bereich für sinnvolle Werte erstreckt sich über 0,9...0,92...0,95.
Werden Schriftgrößen unverändert übernommen (Faktor 1,0), erscheint Text im Verhältnis zu anderen Bildelementen größer als in XFig angezeigt.
- **normal text** = *string:text-handling*
legt fest, wie normaler (non-special) Text zu verarbeiten ist. Der Wert *text-handling* besteht aus Key-Value-Paaren, die durch Kommata getrennt sind, Key und Value sind durch Doppelpunkt getrennt. Folgende Keys können auftreten:
 - handling
legt fest, ob der Text durch T_EX/L^AT_EX gesetzt werden soll („tex“) oder nicht („none“).
 - font
legt fest, welcher Font benutzt wird:
 - * „fig“ benutzt genau den PostScript-Font, der in der Fig-Datei festgelegt wurde,
 - * „similar“ erlaubt (für handling:tex) die Auswahl eines L^AT_EX-Fonts, der in seinen Merkmalen (serif/sansserif, upright/italic, normal/bold) mit dem in der Fig-Datei angegebenen Font übereinstimmt,
 - * „tex“ überläßt (für handling:tex) die Font-Auswahl komplett L^AT_EX.
 - size
wählt die Schriftgröße aus, entweder wie in der Fig-Datei angegeben („fig“) oder überläßt die Auswahl L^AT_EX („tex“).
 - mbox
tritt nur ohne Value auf. Ist dieser Key enthalten, wird das Text-Label in eine \mbox-Struktur gepackt.
- **special text** = *string:text-handling*
legt fest, wie special Text zu behandeln ist. Das *text-handling* wird genauso angegeben wie unter „normal text“, mit Ausnahme des „handling“.
- **tex command** = *string:tex-command*
legt fest, ob L^AT_EX („latex“) oder T_EX („tex“) benutzt wird.
- **latex font setup** = *string:setup-name*
wählt eine vordefinierte L^AT_EX-Preamble aus, die bestimmte Fonts einstellt. Mögliche Setup-Namen sind: „pdf“, „ams“, „newcent“, „pdf12“, „ams12“ und „newcent12“.

Hinweis: Es wird empfohlen, eine eigene, zum L^AT_EX-Dokument passende Font-Preamble zu erstellen und mittels „`preamble file`“ zu benutzen.

- **preamble file** = *file-name*
gibt den Dateinamen für eine abgerüstete Preamble vor, die nur dieselben Font-Einstellungen enthält wie das L^AT_EX-Dokument.
Diese Preamble darf kein „`\begin{document}`“ enthalten.
- **repeat error messages** = *boolean:repeat*
legt fest, ob ein Treiber die Meldung, daß kein special text verarbeitet werden kann, mehrmals ausgibt oder nur beim ersten Auftreten von special text.
- **skip all texts** = *yes*
legt fest, daß es keiner Meldungen bedarf, wenn der Treiber keinerlei Text ausgibt.
Diese Option ist insbesondere im Zusammenhang mit

```
1 normal text = handling:tex
```


sinnvoll, um Warnungen zu unterdrücken, daß Text nicht verarbeitet wird.
- **utf-8** = *boolean*
schaltet den UTF-8-Support ein oder aus. Alternativ kann hier „auto“ gewählt werden. In diesem Fall wird untersucht, ob die Umgebungsvariable LANG auf „utf-8“ endet.
Ist der UTF-8-Support eingeschaltet, werden alle „normalen“ (d. h. nicht-special) Texte UTF-8-dekodiert.
Die Verarbeitung der gefundenen Unicode-Zeichen hängt vom Treiber und dem für normale Texte gewählten Text-Handling ab. Wird L^AT_EX für das Text-Handling genutzt („`handling:tex`“), versucht fig2vect, passenden L^AT_EX-Code zur Darstellung des Unicode-Zeichens zu finden und auszugeben.
Erfolgt das Text-Handling direkt durch den Treiber („`handling:none`“) werden Unicode-Zeichen im Bereich 0x00000000...0x000000FF direkt ausgegeben und alle anderen Zeichen werden ignoriert.



Abbildung 1.11: use metapost arrowheads = yes / no

1.5.2 Konfigurationseinträge für den MetaPost-Treiber

Pfeilspitzen

- **use metapost arrowheads** = *boolean:mp-arrowheads*
legt fest, ob MetaPost-Pfeilspitzen benutzt werden. MetaPost unterstützt nicht alle Pfeilspitzen-Typen von Fig, daher werden ggf. automatische Umwandlungen vorgenommen. Vorteilhaft ist aber, daß an gekrümmte Linien (Splines, Kreisbögen) auch gekrümmte Pfeilspitzen gelegt werden, was insbesondere bei kleinen Krümmungsradien vorteilhafter aussieht (siehe Abb. 1.11).

Textsatz

- **normal text** = *string:text-handling*
special text = *string:text-handling*

– handling

- * **handling:tex**
setzt Texte um in „\btex ... \etex“.
- **font:fig**
Es wird der \LaTeX -Font verwendet, der dem PostScript-Font der Fig-Datei entspricht.
- **font:similar**
Es wird der \LaTeX -Font verwendet, der in seinen Merkmalen dem PostScript-Font der Fig-Datei entspricht.
- **font:tex**
Es wird der \LaTeX -Standardfont verwendet.
- * **handling:none**
setzt Texte um in „label "...“.
- **font:fig**
Es wird der PostScript-Fontname des Fonts verwendet, der in der Fig-Datei angegeben ist, z.B. „label "... infont "Times-Roman“.

· **font:tex**

Es wird der \LaTeX -Fontname des Fonts verwendet, der in der Fig-Datei angegeben ist, z.B. „label “. . . “ infont "ptmr"“.

Hinweis: Im Zusammenhang mit „handling:none“ wird „font:fig“ empfohlen, da die Verwendung von \LaTeX -Fontnamen zu Problemen führen kann.

• **embed fonts** = *boolean:embedding*

legt fest, ob MetaPost in der PostScript-Ausgabe die benutzten Fonts mit ausgeben soll („yes“) oder nicht („no“). Zu diesem Zweck wird in die *.mp-Datei entweder „prologues:=1“ oder „prologues:=0“ geschrieben. Das Einbetten von Fonts ist nur für Bilder sinnvoll, die eigenständig für sich genommen betrachtet werden.

Für Bilder, die in \LaTeX -Dokumenten benutzt werden, sollte das Einbetten der Fonts der entsprechenden Anwendung (dvips bzw. pdf \LaTeX) überlassen werden und nicht durch MetaPost vorgenommen werden.

1.5.3 Konfigurationseinträge für den PS/EPS-Treiber

PS-Level und PS-Struktur

- **ps level** = *integer:level*
legt den PostScript-Level fest, entweder 1, 2 oder 3.
- **dsc comments** = *integer:level*
legt fest, welchem PostScript-Level die DSC-Kommentare in etwa entsprechen sollen.
DSC-Kommentare dienen zum Dokumentenmanagement, um beispielsweise zu bestimmte Seiten oder Seitenbereiche zu drucken oder Fonts und Prozeduren lokal im Drucker vorzuhalten anstatt sie mit jedem Druckauftrag mitzusenden. Das Dokumentenmanagement ist häufig getrennt vom PostScript-Interpreter, z.B. beim Drucken über Netzwerk mit LPD-Drucksystem muß das PostScript-Level passend für den Drucker gewählt werden, wohingegen das DSC-Level passend zur Dokumentenmanagement-Software gewählt werden muß.
Wird anstelle einer Zahlenangabe „yes“ gegeben, wird das DSC-Level passend zum PostScript-Level gewählt.
Bereiten die DSC-Kommentare bei der Weiterverarbeitung Probleme, schalten Sie diese mit „no“ einfach aus.
- **ps showpage** = *boolean:show-page*
legt fest, ob am Ende der EPS-Ausgabe ein showpage-Operator geschrieben wird. Für eigenständige Bilder wird dieser Operator benötigt. Bei Bildern, die in Dokumente eingebunden werden, sollte der Operator unterbleiben. Der showpage-Operator darf nur in *.ps-Dateien auftreten, nicht in *.eps-Dateien.
- **page size** = *integer:w integer:h [integer:llx integer:lly integer:urx integer:ury]*
legt Papierbreite und -höhe fest, optional auch den Druckbereich.
Diese Option ist nur für PS-Ausgabe verfügbar, nicht für EPS.
- **page alignment** = *string:alignment*
legt fest, wie die Graphik im Druckbereich positioniert wird (Standard: horizontal und vertikal zentriert). Mit „top“ bzw. „bottom“ kann die vertikale Ausrichtung geändert werden, mit „left“ bzw. „right“ die horizontale Ausrichtung.
Diese Option ist nur für PS-Ausgabe verfügbar, nicht für EPS.
- **ps setpagedevice** = *boolean:set-page-size*
legt fest, ob die „Papiergröße“ mittels setpagedevice-Operator an die Bounding-Box angepaßt wird. Für eigenständige Bilder – insbesondere wenn diese mit GhostScript in ein anderes Format konvertiert werden sollen – sollte dieser Operator gesetzt werden. Bei Bildern, die in Dokumente eingebunden werden, sollte

der Operator unterbleiben. Der `setpagedevice`-Operator darf nur in *.ps-Dateien auftreten, nicht in *.eps-Dateien.

Virtueller Speicher im PostScript-Interpreter

- **bitmap image dictionary** = *boolean:use-dict*

Für eingebundene Images werden Strings benötigt, die mit „def“ definiert werden. Weiterhin wird eine Definition für einen gefilterten Datenstrom angelegt.

Mit

```
1 bitmap image dictionary = yes
```

wird veranlaßt, daß die EPS-Ausgabe einschließlich der Definitionen in einen Konstrukt

```
1 ... dict begin
2 % Zeichenbefehle
3 end
```

eingeschlossen werden. Wird mit dem „end“-Befehl das Dictionary vom Dictionary-Stack entfernt, existiert kein Verweis mehr auf das Dictionary, somit ist dieses nicht mehr benutzbar. Da Referenzen zu den definierten Strings nur noch im nicht mehr benutzbaren Dictionary vorhanden sind, kann auch nicht mehr auf die Strings zugegriffen werden, diese sind also ebenfalls nicht mehr benutzbar. Damit ist es der garbage collection gestattet, sowohl das Dictionary als auch die Strings zu zerstören und den Speicherplatz wieder freizugeben.

Hinweis: Diese Option darf nicht verwendet werden, wenn die Fig-Datei auch PS/EPS-Images einschließt. Die Größe des Dictionary wird anhand der Breiten der eingeschlossenen Images berechnet (ein Dictionary-Element für jede auftretende Breite zuzüglich ein Element für den gefilterten Datenstrom). PS/EPS-Bilder können weitere Definitionen enthalten, damit würde die berechnete Dictionary-Größe überschritten.

- **force garbage collection** = *boolean:gc*

legt fest, ob am Ende der EPS-Ausgabe mit „1 vmreclaim“ eine garbage collection angefordert werden soll. Dies ist nur sinnvoll, wenn mit

```
1 bitmap image dictionary = yes
```

der Speicher für Bitmap-Images als „ab jetzt ungenutzt“ markiert wurde.

Encoding für eingebundene Bitmaps

- **ps run-length encoding** = *boolean:allow*

legt fest, ob Run-Length-Encoding genutzt werden darf oder nicht.

- **separated rgb channels = *boolean:separate***

legt fest, wie die RGB-Daten angeordnet werden, wenn Run-Length-Encoding genutzt wird.

Ist *separate* aktiviert, werden zuerst alle Rot-Werte, dann alle Grün-Werte und abschließend alle Blau-Werte geschrieben. Bei deaktivierten Channels folgt Punkt für Punkt jeweils Rot-Wert, Grün-Wert und Blau-Wert.

Die separierten Kanäle erlauben eine Run-Length-Komprimierung bei gleichfarbigen Flächen. Ist die Option deaktiviert, können nur graue Flächen mit gleichem Grauwert run-length-komprimiert werden.

Ein Nachteil der separierten Kanäle ist der wesentlich höhere Speicherbedarf im PostScript-Interpreter. Es werden $M_s = 3 \cdot w \cdot h$ Bytes für jedes Bitmap-Image benötigt anstelle von $M_n = w$ Bytes beim kombinierten Datenstrom. Hierbei ist w die Bildbreite in Pixeln und h die Bildhöhe.

1.5.4 Konfigurationseinträge für den PDF-Treiber

Allgemeine Optionen

- **plain text streams** = *boolean:plain*
legt fest, ob die Streams für die Graphik und die Füllmuster als ASCII-Text oder komprimiert angelegt werden.
- **full screen** = *boolean:full-screen*
legt fest, ob beim Öffnen der PDF-Datei auf Full-Screen-Darstellung umgeschaltet wird.
- **arc bezier steps** = *integer:steps*
legt fest, durch wieviele Bezier-Spline-Elemente ein Viertelkreis dargestellt wird (Standard: 2).
- **allow pdf page attributes** = *boolean:attributes*
erlaubt bzw. verbietet das Schreiben von PDF-Seitenattributen. Diese Attribute werden benötigt, wenn die Fig-Datei PNGs mit Alphakanal referenziert und die Ausgabedatei als eigenständiges Bild betrachtet werden soll.

Füllmuster, Füllungen und Strichmuster

- **tiled patterns** = *boolean:tiled*
legt fest, ob Füllmusterkacheln so groß wie das gesamte Bild angelegt werden (no) oder kleinstmöglich (yes).
Bei den kleinstmöglichen Kacheln kommt es durch Rundungseffekte mitunter zu Irritationen an den Fügstellen der Kacheln, insbesondere wenn Linien schräg über die Kachelgrenzen verlaufen.
Die Verwendung einer großen Füllmusterkachel führt zu einer besseren Bildschirmdarstellung, erhöht aber auch die Dateigröße.

Eingebundene Bitmap-Images

- **interpolate images** = *boolean:interpolate*
legt fest, ob für eingebundene Images eine Interpolation aktiviert wird. Dies ist insbesondere dann sinnvoll, wenn die Images stark vergrößert werden.
- **flip direction** = *string:flip-dir*
legt die Flip-Richtung fest, entweder „horizontal“ oder „diagonal“.

1.5.5 Konfigurationseinträge für den TeX-Treiber

- **full tex file** = *boolean:full-flag*
legt fest, ob ein komplettes L^AT_EX-Dokument erstellt wird, das mit pdfL^AT_EX verarbeitet werden kann oder nur ein Abschnitt, der mit „\input“ in ein L^AT_EX-Dokument eingebunden wird.

1.5.6 Konfigurationseinträge für den SVG-Treiber

SVG-Kopfbereich und SVG-Struktur

- **svg version** = *string:version*
legt die SVG-Version fest, entweder „1.0“, „1.1“ oder „1.2“. Empfohlen wird, den Standard „1.0“ zu verwenden, da hier die Unterstützung durch Browser-Plug-Ins am verbreitetsten ist.
- **embedded svg** = *boolean:embedded*
legt fest, ob SVG produziert wird, das in XML-Dokumente eingebettet wird. Standard ist die Erzeugung eigenständiger SVG-Dateien.
- **wh specification** = *string:wh-spec*
legt fest, in welcher Einheit Länge und Breite der Graphik in die SVG-Datei geschrieben werden. Mögliche Einstellungen sind Zoll („inches“), PostScript-Points („points“) oder Pixel („pixels“).

Objekt-Attribute

In SVG-Dateien gibt es verschiedene Möglichkeiten für die Festlegung von Objekteigenschaften:

- Angabe aller Attribute einzeln, z.B.

```
1 <rect fill="yellow" stroke="black" stroke-width="0.9" .../ >
```

- Angabe eines style-Attributes, z.B.

```
1 <rect style="fill: yellow; stroke: black; stroke-width: 0.9;" .../ >
```

- Angabe eines class-Attributes und Nutzung von CSS-Stylesheets

```
1 <defs><style type="text/css"><![CDATA[  
2   .c1 {  
3     fill: yellow;  
4     stroke: black;  
5     stroke-width: 0.9;  
6   }  
7 ]]></style></defs>  
8 <rect class="c1" .../ >
```

Insbesondere bei vielen Objekten mit gleichen Eigenschaften führt die Verwendung des class-Attributes zu geringeren Dateigrößen.

Werden für ein Objekt mehrere Angaben genutzt, sind style- und class-Attribut höher priorisiert als die einzeln angegebenen Attribute. Dies muß beachtet werden, wenn

1.5 Konfigurationseinträge

Animation und Event-Handler genutzt werden, da diese meist einzelne Attribute ändern. Wurden die Objekteigenschaften über ein style- oder class-Attribut angegeben, hat das anschließende Setzen bzw. Ändern einzelner Attribute keine Auswirkungen auf die Darstellung. Sollen Animationen oder Event-Handler genutzt werden, müssen die Objekteigenschaften als einzelne Attribute angegeben werden.

- **prepare for modifications** = *boolean:mods*
erzwingt die Angabe aller Attribute einzeln. Findet fig2vect Hinweise auf die Verwendung von ECMA-Script, wird dieser Modus automatisch verwendet.
- **use css** = *boolean:use-css*
legt fest, ob Objekteigenschaften über CSS-Klassen festgelegt werden oder ob alle Objekteigenschaften über ein style-Attribut definiert werden.

Schriften

- **gs svg-font directory** = *string:directory*
konfiguriert den Treiber so, daß SVG-Fonts im angegebenen Verzeichnis verwendet werden.
Das statische Font-Mapping ist veraltet. Empfohlen wird, Font-Konfigurationsdateien zu verwenden (siehe nächster Punkt).
- **font configuration file** = *string:Dateiname*
gibt eine Font-Konfigurationsdatei an, z.B. „winfont.cfg“. Für die Fontkonfigurationsdateien wird der dklibs-Suchmechanismus für Konfigurationsdateien verwendet. Es wird empfohlen, die Font-Konfigurationsdateien in $\{\text{prefix}\}/\text{etc}/\text{fig2vect}$ bzw. $\%WINDIR\%\backslash\text{fig2vect}$ abzulegen. Der Aufbau der Font-Konfigurationsdateien wird in dkfont(5) bzw. im dklibs-Handbuch beschrieben.

ECMA-Script

- **js library** = *string:filename*
verwendet die angegebene Datei als Funktionsbibliothek.

Wird ECMA-Script (JavaScript) verwendet, müssen einige Objekte eine ID erhalten. Dies kann mit einem Steuerkommentar, z.B.

```
1 svg: id = obj001
```

bewerkstelligt werden. Der Steuerkommentar muß dabei direkt dem entsprechendem Objekt der Fig-Datei zugeordnet sein (also unmittelbar vor dem Objekt stehen).

Event-Handler werden ebenfalls als Steuerkommentar einem Objekt zugeordnet, z.B.

```
1 svg: onmouseover = changebgcolor(evt)
```

Es werden die Events „onfocusin“, „onfocusout“, „onactivate“, „onclick“, „onmousedown“, „onmouseup“, „onmouseover“, „onmousemove“, „onmouseout“, „onload“, „onunload“, „onabort“, „onerror“, „onresize“, „onscroll“, „onzoom“, „onbegin“, „onend“ und „onrepeat“ unterstützt.

Handler für Dokument-Events (z.B. onload) werden als Steuerkommentare in den Dokument-Kommentarbereich geschrieben – also zwischen Zeile 8 (transparente Farbe) und 9 (Auflösung/Koordinatensystem). Ein Beispiel für eine Fig-Datei, in die nachträglich Steuerkommentare für ECMA-Script eingefügt wurden, finden Sie in Datei doc2.fig⁷.

⁷<http://fig2vect.sourceforge.net/doc2.fig>

1.6 Praktische Tipps

1.6.1 Hintergrund-Rechteck

Fig2vect ermittelt die Bildabmessungen aus den Abmessungen der Graphikelemente mit Ausnahme der Texte.

Das Auslassen der Texte hat folgende Gründe:

- Die benutzten Font-Dateien sind abhängig vom verwendeten Ausgabetreiber, die Angaben zu den Abmessungen werden jedoch schon benötigt, bevor der Treiber gestartet wird.
- Je nach verwendeter Konfiguration wird der Textsatz \LaTeX überlassen, fig2vect kann nicht voraussagen, wie der Textsatz durch \LaTeX erfolgt.

Steht ein Text an einem Bildrand, benötigt fig2vect etwas „Nachhilfe“ und muß mit Informationen zu den Bildabmessungen versorgt werden. Diese Informationen können einem speziellen Hintergrund-Rechteck entnommen werden, das in der Ausgabe nicht mit gezeichnet wird, sondern nur die Bildabmessungen bereitstellt. Ein solches Hintergrund-Rechteck kann auch verwendet werden, um noch etwas Rand zwischen den äußeren Graphikelementen und den Bildgrenzen zu erzeugen.

Ein Hintergrund-Rechteck ist ein Rechteck (Typ 2: Polygon, Subtyp 2: Box) in Layer 999 (der tiefstmögliche Layer) mit folgenden Eigenschaften: Liniendicke 0, Linienfarbe -1 (Default), Füll-Stil -1 (keine Füllung) Füllfarbe egal. Das Hintergrund-Rechteck schließt alle anderen Graphikelemente ein.

Unter XFig können Sie die Linienfarbe -1 über die Edit-Funktion einstellen, indem Sie die Farbe „Default“ verwenden. Unter jFig besteht diese Möglichkeit nicht. Hier verwenden Sie als Linienfarbe weiß, mit dem Konfigurationseintrag

```
1 background rectangle color = white
```

teilen Sie fig2vect mit, daß Hintergrundrechtecke weiß als Linienfarbe haben.

1.6.2 Umgang mit Füllmustern

Der Speicherplatz und der Rechenaufwand für Füllmuster hängt stark vom verwendeten Ausgabetreiber ab. Während bei einigen Treibern nur die Befehle zum Zeichnen einer Füllmuster-Kachel angegeben werden müssen bzw. eine Prozedur zum Zeichnen des Füllmusters, erfordern andere Treiber alle Zeichenbefehle zum kompletten Erstellen des Füllmusters.

In MetaPost-Dateien muß die Füllung von Objekten komplett beschrieben werden. Der Speicherplatz-Bedarf für gefüllte Objekte ist für die meisten Füllmuster sowohl proportional zur Objektbreite als auch proportional zur Objekthöhe. Eine Ausnahme bilden einfache Füllmuster (gerade und schräge Linien und daraus erzeugte Gitternetze), bei denen der Speicherbedarf ungefähr proportional zur Summe aus Höhe und Breite ist. Sie sollten bevorzugt solche einfachen Füllmuster verwenden.

In EPS-Dateien werden Füllmuster durch PostScript-Prozeduren realisiert, der Speicherbedarf ist weitgehend unabhängig von den Objektmessungen.

In PDF-Dateien gibt es prinzipiell die Möglichkeit, ein Füllmuster durch Aneinanderfügen von Kacheln zu erzeugen. Es müssen dann nur die Befehle zum Zeichnen einer Kachel in die Ausgabedatei geschrieben werden. Nachteilig ist dabei, daß es durch mathematische Rundungen zu Irritationen an den Fugen kommen kann, insbesondere wenn schräge Linien über Kachelgrenzen verlaufen.

Daher bietet der PDF-Treiber auch die Möglichkeit, jedes Füllmuster in Form einer großen Kachel darzustellen, die so groß ist wie das gesamte Bild. Damit entfallen die Fügstellen, die Bildschirmdarstellung wird wesentlich besser, allerdings steigt der Speicherbedarf für die meisten Füllmuster sowohl proportional mit der Bildbreite als auch proportional mit der Bildhöhe (mit Ausnahme der o.g. einfachen Füllmuster).

In SVG-Dateien werden Füllmuster durch Aneinanderfügen von Kacheln dargestellt, es müssen nur die Befehle zum Zeichnen einer Kachel in die Ausgabedatei geschrieben werden.

1.6.3 Eingebundene Bilder

Bei der Einbindung von Bildern unterscheiden sich jFig und XFig. Dem Programm jFig ist es egal, welche der vier Polygon-Ecken zuerst angegeben wird, das eingebundene Bild wird immer aufrecht stehen gelassen.

XFig erzeugt aufrecht stehende Bilder nur, wenn bei der Angabe des Polygons mit der linken oberen Ecke begonnen wird. Wird mit einer anderen Ecke begonnen, dreht XFig das Bild um 90° bzw. um Vielfache von 90° .

Fig2vect ermittelt aus den Polygon-Punkten lediglich den linken und rechten x -Wert und den oberen und unteren y -Wert, es werden keine Rotationen vorgenommen.

Verwenden Sie in XFig daher nur nicht-rotierte Bilder, achten Sie darauf, bei der Angabe des Polygons mit der linken oberen Ecke zu beginnen. Wird eine Bildrotation gewünscht, nehmen Sie sie am besten mit einem Graphikprogramm vor.

1.6.4 SVG-Bilder

SVG-Bilder können zu zwei verschiedenen Zwecken erstellt werden:

- Publikation auf Webseiten und
- eingebettete Bilder in Textverarbeitungen.

Fig-Dateien verwenden die 35 PostScript-Fonts, diese sind in SVG nicht vordefiniert. Der SVG-Viewer muß entweder mit den entsprechenden Fonts versorgt werden, oder die Fonts werden durch andere Fonts ersetzt.

Für Webseiten bietet es sich an, SVG-Varianten der GhostScript-Fonts zu referenzieren.

Für die Einbettung von Bildern z.B. in eine Windows-Textverarbeitung empfiehlt sich die Font-Ersetzung, dabei sollten die Fonts benutzt werden, die auch im Dokumententext verwendet werden.

Für beide Zwecke kann eine Font-Konfigurationsdatei verwendet werden, deren Name mit der Option „-o font.configuration.file=*Dateiname*“ bzw. mit der Konfigurationsoption „font configuration file = *Dateiname*“ angegeben wird.

Es werden zwei Konfigurationsdateien mitgeliefert:

- webfont.cfg
zeigt die Referenzierung von SVG-Fonts und
- winfont.cfg
zeigt die Verwendung von System-Fonts.

Beide Dateien müssen vor Verwendung angepaßt werden.

In „webfont.cfg“ muß hauptsächlich der „directory“-Eintrag für jeden Font angepaßt werden, z.B. auf „<http://www.my-corp.com/fonts/urw-svg>“.

In „winfont.cfg“ müssen Sie kontrollieren, dass nur auf solche Fonts bzw. Fontfamilien verwiesen wird, die auf Ihrem System tatsächlich installiert sind.

1.7 Bitmap Images

1.7.1 Übersicht

Mit fig2vect können Sie Bitmap-Images nicht direkt erstellen, stattdessen erstellen Sie zunächst mit fig2vect eine Vektorgraphik, die dann mit einem Konvertierungsprogramm zu einer Bitmap-Datei umgewandelt wird.

Folgende Dateiformate kommen für die Vektorgraphik in Frage:

- EPS/PS

Die EPS/PS-Datei wird mit GhostScript zu einer Bitmap-Datei umgewandelt. Der klassische Weg, allerdings gibt es Probleme wenn die Fig-Datei teilweise transparente PNG-Bilder einbindet. Das Verfahren ist nur für Fig-Dateien anwendbar, die keinen special text enthalten.

Bei der Konvertierung mit GhostScript wird u.U. die BoundingBox-Angabe der EPS-Datei durch gs ignoriert und ein großes Bitmap erzeugt, das einer Druckseite entspricht. In diesem Fall muß das Image mit einem Graphikprogramm nachbearbeitet werden, um die Ränder abzuschneiden. Mit dem Konfigurationseintrag „ps setpagedevice = yes“ bzw. der Kommandozeilenoption „-o ps.setpagedevice=yes“ kann dies vermieden werden, wenn eine PS-Datei erzeugt wird (Option „-lps“).

Alternativ dazu kann GhostScript auch mit der Option „-dEPSCrop“ aufgerufen werden, die Bildgröße in der Ausgabedatei wird dann an die EPS BoundingBox angepaßt.

- PDF

Der Weg über den Zwischenschritt PDF-Datei ist der bevorzugte Weg für die Erstellung von Bitmap-Graphiken.

Auch die PDF-Datei wird mit GhostScript zu einer Bitmap-Datei umgewandelt. Hier gibt es keine Probleme, wenn teilweise transparente PNG-Bilder in der Fig-Datei eingebunden sind. In der Fig-Datei kann special text genutzt werden.

Bei Tests kam es stellenweise zu Problemen, so brach gs 7.07 unter Fedora Core 3 mit einem Speicherzugriffsfehler ab. Unter Windows XP Prof. mit gs 8.15 gab es keine Probleme.

- SVG

SVG-Dateien können mit dem Batik Rasterizer nach PNG konvertiert werden. Dieser Weg ist allerdings nur dann gangbar, wenn die Fig-Datei keinen special text enthält.

Vorteil dieses Verfahrens ist, daß die Bildbereiche ohne Graphikelemente in der PNG-Ausgabe dann transparent sind. Daher ist das Verfahren besonders für die Erstellung von Web-Logos u.ä. geeignet.

Das Batik-Projekt finden Sie auf <http://xml.apache.org>, zusätzlich zur

1 Nutzer-Handbuch

Batik-Software sollten Sie die SVG-Ports der GhostScript-Fonts installiert haben, siehe 1.2.3 auf Seite 12.

1.7.2 Details für die verschiedenen Konvertierungsmöglichkeiten

Bitmaps über EPS

Zunächst wird mit

```
fig2vect -leps.standalone demofig.fig demofig.eps
```

eine EPS-Datei erzeugt.

Die EPS-Datei demofig.ps wird mit

```
gs Optionen -sOutputFile=demoeps.png demofig.eps
```

zu einer PNG-Datei umgewandelt. Dabei werden Optionen entsprechend Tabelle 1.1 eingesetzt. Unter Windows muß „gswin32c“ anstelle von „gs“ verwendet werden.

Tabelle 1.1: GhostScript-Optionen für die PNG-Erzeugung

Option	Bedeutung
-dSAFER	versetzt GhostScript in den sicheren Modus und unterbindet Aktivitäten, die das System beschädigen können.
-dBATCH	stellt Batch-Betrieb ein, d.h. nach Abarbeitung der Quelldatei beendet sich GhostScript, anstatt weitere Eingaben von der Kommandozeile zu erwarten.
-dNOPAUSE	kein Prompt um jede einzelne Seite zu bestätigen.
-dEPSCrop	passt die Bildgröße in der Ausgabedatei an die BoundingBox der EPS-Datei an.
-dNOCACHE	unterbindet das Caching von Glyph-Bitmaps. Laut Handbuch sollte diese Option nur zum Debugging verwendet werden, in Newsgroups gab es jedoch Hinweise, daß die Verwendung dieser Option zu einer besseren Ausgabequalität führt.
-sDEVICE=png16m	Ausgabe als PNG-Datei
-r200	legt die Auflösung fest (hier 200dpi, Sie können natürlich andere Zahlen verwenden).
-dGraphicsAlphaBits=4 -dTextAlphaBits=4	stellt die Qualität für die Kantenglättung in Graphik- und Textelementen ein. Sie können hier 1, 2, 4... und weitere Zweierpotenzen verwenden oder die Optionen ganz entfallen lassen.

Bitmap über PDF

Die PDF-Datei wird mit

```
fig2vect -l pdf demofig.fig demofig.pdf
```

erstellt.

Enthält die Fig-Datei Text, wird

```
fig2vect -l pdf.tex demofig.fig demoi.pdf
fig2vect -l tex.full -i demoi.pdf demofig.fig
demofig.tex
pdflatex demofig && pdflatex demofig
gs Optionen -sOutputFile=demopdf.png demofig.pdf
```

verwendet. Es werden dieselben Optionen wie bei der Konvertierung von EPS-Dateien verwendet (siehe Tabelle 1.1 auf der vorherigen Seite).

Bitmap über SVG

Die Fig-Datei wird mit

```
fig2vect -lsvg demofig.fig demofig.svg
```

in eine SVG-Datei umgewandelt.

Die zusätzliche Option „-o wh.specification=pixels“ sorgt dafür, daß die Bildabmessungen in Pixeln in die SVG-Datei geschrieben werden. Diese Bildabmessungen können Sie mit einem Texteditor ändern, bevor Sie mit dem nächsten Schritt fortfahren.

Die SVG-Datei wird dann mit

```
java -jar ../../batik-rasterizer.jar demofig.svg
```

nach PNG (demofig.png) konvertiert.

2 Technische Details

2.1 Mathematische Aspekte

2.1.1 X-Splines und Bezier-Splines

Kurven

Kurven in \mathbb{R}^2 können durch die Gleichungen

$$x = x(t) \quad y = y(t) \quad t \in [t_S; t_E]$$

beschrieben werden.

Im Bereich der Computergraphik wird dabei häufig $t \in [0; 1]$ für die gesamte Kurve oder für ein Kurvensegment verwendet.

Ein Spline ist eine Beschreibung für eine Kurve, dabei wird eine kontinuierliche Kurve durch einen Satz diskreter Werte bestimmt.

Kubische Bezier-Splines

Für kubische Bezier-Splines wird jedes Segment durch seine zwei Endpunkte $P_i(x_i, y_i)$ und $P_j(x_j, y_j)$ sowie zwei Kontrollpunkte $P_{ir}(x_{ir}, y_{ir})$ und $P_{jl}(x_{jl}, y_{jl})$ mit $j = i + 1$ beschrieben. Dabei bezeichnet P_{ir} den Kontrollpunkt „rechts“ von der Stützstelle P_i , d.h. für den Kontrollpunkt ist t größer als für die Stützstelle. P_{jl} bezeichnet den Kontrollpunkt „links“ von der Stützstelle, d.h. am Kontrollpunkt ist t kleiner als an der entsprechenden Stützstelle.

In der Literatur wird häufig auch die Schreibweise P_{i+} bzw. P_{j-} oder auch P_i^+ bzw. P_j^- für die Kontrollpunkte verwendet.

$$\begin{aligned}x(t) &= x_i(1-t)^3 + 3x_{ir}(t-1)^2t + 3x_{jl}(1-t)t^2 + x_jt^3 \\y(t) &= y_i(1-t)^3 + 3y_{ir}(t-1)^2t + 3y_{jl}(1-t)t^2 + y_jt^3 \\t &\in [0; 1] \\x(t) &= x_i(1-2t+t^2)(1-t) + 3x_{ir}(1-2t+t^2)t + 3x_{jl}(1-t)t^2 + x_jt^3 \\&= x_i(1-3t+3t^2-t^3) + 3x_{ir}(t-2t^2+t^3) + 3x_{jl}(t^2-t^3) + x_jt^3 \\x'(t) &= x_i(-3+6t-3t^2) + 3x_{ir}(1-4t+3t^2) + 3x_{jl}(2t-3t^2) + x_j \cdot 3t^2\end{aligned}$$

2 Technische Details

Wenn Vorgaben für $x'(0)$ und $x'(1)$ existieren, können x_{ir} und x_{jl} berechnet werden:

$$x_{ir} = x_i + \frac{1}{3}x'(0)$$

$$x_{jl} = x_j - \frac{1}{3}x'(1)$$

Analog gilt für y :

$$y_{ir} = y_i + \frac{1}{3}y'(0)$$

$$y_{jl} = y_j - \frac{1}{3}y'(1)$$

Für jede innere Stützstelle existieren somit zwei zusätzliche Kontrollpunkte. Ein Nachteil der Bezier-Splines ist, daß es für unerfahrene Nutzer schwierig ist, abzuschätzen, wie sich eine Manipulation der Kontrollpunkte auf den Kurvenverlauf auswirkt.

X-Splines

Cross-splines (X-Splines) wurden 1995 durch Schlick und Blanc vorgestellt. X-Splines werden hier nicht im Detail erklärt, hierzu sei auf die entsprechenden SIGGRAPH-Veröffentlichungen verwiesen. Die Darstellung an dieser Stelle beschränkt sich auf die Verwendung von X-Splines in FIG-Dateien und die Behandlung im Programm fig2vect.

Eine Funktion $f(u)$ wird als Gewichtsfunktion für die Approximation verwendet:

$$f(u, p) = \begin{cases} u^3(10 - p + (2p - 15)u + (6 - p)u^2) & \text{für } u \in [0; 1] \\ 0 & \text{ansonsten} \end{cases}$$

$$= \begin{cases} (6 - p)u^5 + (2p - 15)u^4 + (10 - p)u^3 & \text{für } u \in [0; 1] \\ 0 & \text{ansonsten} \end{cases}$$

Zwei Teilfunktionen werden verwendet, um die Gewichtsfunktion $e(u)$ für die Interpolation zu bilden:

$$e(u, p, q) = \begin{cases} qu + 2qu^2 + (10 - 12q - p)u^3 \\ \quad + (2p + 14q - 15)u^4 + (6 - 5q - p)u^5 & \text{für } u \in [0; 1] \\ qu + 2qu^2 - 2qu^4 - qu^5 & \text{für } u \in [-1; 0) \\ 0 & \text{ansonsten} \end{cases}$$

Die Ableitung der Funktionen ergibt:

$$\frac{\partial f}{\partial u} = \begin{cases} 5(6 - p)u^4 + 4(2p - 15)u^3 + 3(10 - p)u^2 & \text{für } u \in [0; 1] \\ 0 & \text{ansonsten} \end{cases}$$

$$\frac{\partial e}{\partial u} = \begin{cases} 5(6 - 5q - p)u^4 + 4(2p + 14q - 15)u^3 \\ \quad + 3(10 - 12q - p)u^2 + 4qu + q & \text{für } u \in [0; 1] \\ q + 4qu - 8qu^3 - 5qu^4 & \text{für } u \in [-1; 0) \\ 0 & \text{ansonsten} \end{cases}$$

Bei der Verwendung von von X-Splines in FIG-Dateien wird davon ausgegangen, daß äquidistante Kontrollpunkte vorliegen, fig2vect setzt hierfür $\Delta t = 1$.

Ein X-Spline wird durch n Kontrollpunkte mit Indizes im Bereich von 0 bis $n - 1$ beschrieben, dabei ist jedem Kontrollpunkt P_k am Zeitpunkt $t = T_k$ ein zusätzlicher Parameter s_k zugeordnet. Es gilt $-1 \leq s_k \leq 1$. Punkte mit $0 \leq s_k \leq 1$ sind dabei Approximationspunkte, Punkte mit $-1 \leq s_k < 0$ sind Interpolationspunkte.

2 Technische Details

Zu jedem Kontrollpunkt P_k gehört eine linksseitige Gewichtsfunktion F_k^- und eine rechtsseitige Gewichtsfunktion F_k^+ . Diese Gewichtsfunktionen geben an, wie stark der Kontrollpunkt sich auf den Verlauf in den jeweils angrenzenden Segmenten auswirkt. Die Kontrollfunktionen erstrecken sich jeweils über bis zu 2 Segmente, ein Kontrollpunkt kann somit also insgesamt 4 Segmente beeinflussen.

Bei der Auswahl der links- und rechtsseitigen Gewichtsfunktion werden die jeweils benachbarten Punkte mit berücksichtigt, siehe Tabelle 2.1.

Außerhalb des Definitionsbereiches – d.h. für $t > T_i^+$ bzw. $t < T_j^-$ – nehmen die Funktionen F_i^+ und F_j^- jeweils den Wert 0 an.

Tabelle 2.1: Auswahl der Gewichtsfunktionen

Art des Punktes		Gewichtsfunktion	
P_i	P_j	F_i^+	F_j^-
Approximation	Approximation	$T_i^+ = T_j + s_j \Delta t$ $p_i^+ = \frac{2(T_i^+ - T_i)^2}{\Delta t^2}$ $F_i^+ = f\left(\frac{t - T_i^+}{T_i - T_i^+}, p_i^+\right)$	$T_j^- = T_i - s_i \Delta t$ $p_j^- = \frac{2(T_j^- - T_j)^2}{\Delta t^2}$ $F_j^- = f\left(\frac{t - T_j^-}{T_j - T_j^-}, p_j^-\right)$
Approximation	Interpolation	$T_i^+ = T_j$ $p_i^+ = \frac{2(T_i^+ - T_i)^2}{\Delta t^2}$ $F_i^+ = f\left(\frac{t - T_i^+}{T_i - T_i^+}, p_i^+\right)$	$T_j^- = T_i - s_i \Delta t$ $p_j^- = \frac{2(T_j^- - T_j)^2}{\Delta t^2}$ $F_j^- = f\left(\frac{t - T_j^-}{T_j - T_j^-}, p_j^-\right)$
Interpolation	Approximation	$T_i^+ = T_j + s_j \Delta t$ $p_i^+ = \frac{2(T_i^+ - T_i)^2}{\Delta t^2}$ $F_i^+ = f\left(\frac{t - T_i^+}{T_i - T_i^+}, p_i^+\right)$	$T_j^- = T_i$ $p_j^- = \frac{2(T_j^- - T_j)^2}{\Delta t^2}$ $F_j^- = f\left(\frac{t - T_j^-}{T_j - T_j^-}, p_j^-\right)$
<i>wird fortgesetzt</i>			

2.1 Mathematische Aspekte

Fortsetzung			
Interpolation	Interpolation	$T_i^+ = T_i + 2\Delta t$ $p_i^+ = 2$ $q_i^+ = -\frac{1}{2} \cdot s_i$ $F_i^+ = e\left(\frac{T_j-t}{\Delta t}, p_i^+, q_i^+\right)$	$T_j^- = T_j - 2\Delta t$ $p_j^- = 2$ $q_j^- = -\frac{1}{2} \cdot s_j$ $F_j^- = e\left(\frac{t-T_i}{\Delta t}, p_j^-, q_j^-\right)$
Approximation, P_i ist linker Rand eines offenen Spline	Interpolation	$T_i^+ = T_i + 2\Delta t$ $p_i^+ = 2$ $q_i^+ = -\frac{1}{2} \cdot s_j$ $F_i^+ = e\left(\frac{T_j-t}{\Delta t}, p_i^+, q_i^+\right)$	$T_j^- = T_i$ $p_j^- = \frac{2(T_j^- - T_j)^2}{\Delta t^2}$ $F_j^- = f\left(\frac{t-T_j^-}{T_j-T_j}, p_j^-\right)$
Interpolation	Approximation, P_j ist rechter Rand eines offenen Spline	$T_i^+ = T_j$ $p_i^+ = \frac{2(T_i^+ - T_i)^2}{\Delta t^2}$ $F_i^+ = f\left(\frac{t-T_i^+}{T_i-T_i^+}, p_i^+\right)$	$T_j^- = T_j - 2\Delta t$ $p_j^- = 2$ $q_j^- = -\frac{1}{2} \cdot s_i$ $F_j^- = e\left(\frac{t-T_i}{\Delta t}, p_j^-, q_j^-\right)$

Durch den Ansatz $\Delta t = 1$ in fig2vect können Vereinfachungen entsprechend Tabelle 2.2 getroffen werden.

Tabelle 2.2: Auswahl der Gewichtsfunktionen in fig2vect

Art des Punktes		Gewichtsfunktion	
P_i	P_j	F_i^+	F_j^-
Approx.	Approx.	$T_i^+ = j + s_j$ $p_i^+ = 2(T_i^+ - i)^2$ $F_i^+ = f\left(\frac{t-T_i^+}{i-T_i^+}, p_i^+\right)$	$T_j^- = i - s_i$ $p_j^- = 2(T_j^- - j)^2$ $F_j^- = f\left(\frac{t-T_j^-}{j-T_j^-}, p_j^-\right)$
wird fortgesetzt			

2 Technische Details

Fortsetzung			
Approx.	Interpolation	$T_i^+ = j$ $p_i^+ = 2$ $F_i^+ = f\left((j-t), p_i^+\right)$	$T_j^- = i - s_i$ $p_j^- = 2(T_j^- - j)^2$ $F_j^- = f\left(\frac{t-T_j^-}{j-T_j^-}, p_j^-\right)$
Interpolation	Approx.	$T_i^+ = j + s_j$ $p_i^+ = 2(T_i^+ - i)^2$ $F_i^+ = f\left(\frac{t-T_i^+}{i-T_i^+}, p_i^+\right)$	$T_j^- = i$ $p_j^- = 2$ $F_j^- = f\left((t-i), p_j^-\right)$
Interpolation	Interpolation	$T_i^+ = i + 2$ $p_i^+ = 2$ $q_i^+ = -\frac{1}{2} \cdot s_i$ $F_i^+ = e\left((j-t), p_i^+, q_i^+\right)$	$T_j^- = j - 2$ $p_j^- = 2$ $q_j^- = -\frac{1}{2} \cdot s_j$ $F_j^- = e\left((t-i), p_j^-, q_j^-\right)$
Approx., P_i ist linker Rand eines offenen Spline	Interpolation	$T_i^+ = i + 2$ $p_i^+ = 2$ $q_i^+ = -\frac{1}{2} \cdot s_j$ $F_i^+ = e\left((j-t), p_i^+, q_i^+\right)$	$T_j^- = i$ $p_j^- = 2$ $F_j^- = f\left((t-i), p_j^-\right)$
Interpolation	Approx., P_j ist rechter Rand eines offenen Spline	$T_i^+ = j$ $p_i^+ = 2$ $F_i^+ = f\left((j-t), p_i^+\right)$	$T_j^- = j - 2$ $p_j^- = 2$ $q_j^- = -\frac{1}{2} \cdot s_i$ $F_j^- = e\left((t-i), p_j^-, q_j^-\right)$

Nachdem für alle Punkte P_k die Funktionen F_k^- und F_k^+ festgelegt wurden, können für jedes Kurvensegment mit den Randpunkten P_i und P_j Kurvenpunkte berechnet werden

mit

$$z(t) = x_{i-1}F_{i-1}^+(t) + x_iF_i^+(t) + x_jF_j^-(t) + x_{j+1}F_{j+1}^-(t)$$

$$v(t) = F_{i-1}^+(t) + F_i^+(t) + F_j^-(t) + F_{j+1}^-(t)$$

$$z'(t) = x_{i-1} \frac{\partial F_{i-1}^+(t)}{\partial t} + x_i \frac{\partial F_i^+(t)}{\partial t} + x_j \frac{\partial F_j^-(t)}{\partial t} + x_{j+1} \frac{\partial F_{j+1}^-(t)}{\partial t}$$

$$v'(t) = \frac{\partial F_{i-1}^+(t)}{\partial t} + \frac{\partial F_i^+(t)}{\partial t} + \frac{\partial F_j^-(t)}{\partial t} + \frac{\partial F_{j+1}^-(t)}{\partial t}$$

$$x(t) = \frac{z(t)}{v(t)}$$

$$x'(t) = \frac{v(t) \cdot z'(t) - z(t) \cdot v'(t)}{v^2(t)}$$

Die Berechnung von $y(t)$ und $y'(t)$ erfolgt analog.

2 Technische Details

Für die Vereinfachung des Berechnungsprogrammes wird jedes X-Spline-Segment so in eine Anordnung mit vier Punkten A , B , C und D gelegt, daß der Kontrollpunkt für den Anfang des Segments auf B liegt und der Kontrollpunkt für den Endpunkt des Segmentes auf C . A entspricht dann dem „linken“ benachbartem Kontrollpunkt, D dem „rechten“.

Es kann nun gesetzt werden:

$$T_A = -1 \quad T_B = 0 \quad T_C = 1 \quad T_D = 2$$

Für die Berechnung von Kurvenpunkten müssen die Funktionen F_A^+ , F_B^+ , F_C^- und F_D^- herangezogen werden, für die Berechnung der Anstiege $\frac{\partial}{\partial t}F_A^+$, $\frac{\partial}{\partial t}F_B^+$, $\frac{\partial}{\partial t}F_C^-$ und $\frac{\partial}{\partial t}F_D^-$.

Zunächst einmal wird davon ausgegangen, daß

$$\begin{aligned} T_A^+ &= s_B & p_A^+ &= 2(-1 - T_A^+)^2 & u_A(t) &= \frac{t - T_A^+}{-1 - T_A^+} & u_A'(t) &= -\frac{1}{T_A^+ + 1} \\ F_A^+(t) &= f(u_A(t), p_A^+) \\ T_B^+ &= 1 + s_C & p_B^+ &= 2T_B^{+2} & u_B(t) &= \frac{t - T_B^+}{-T_B^+} & u_B'(t) &= -\frac{1}{T_B^+} \\ F_B^+(t) &= f(u_B(t), p_B^+) \\ T_C^- &= -s_B & p_C^- &= 2(1 - T_C^-)^2 & u_C(t) &= \frac{t - T_C^-}{1 - T_C^-} & u_C'(t) &= \frac{1}{1 - T_C^-} \\ F_C^-(t) &= f(u_C(t), p_C^-) \\ T_D^- &= 1 - s_C & p_D^- &= 2(2 - T_D^-)^2 & u_D(t) &= \frac{t - T_D^-}{2 - T_D^-} & u_D'(t) &= -\frac{1}{2 - T_D^-} \\ F_D^-(t) &= f(u_D(t), p_D^-) \end{aligned}$$

Falls $s_A < 0$ und $s_B < 0$, so gilt stattdessen:

$$\begin{aligned} u_A(t) &= -t & u_A'(t) &= -1 & q_A &= -\frac{1}{2}s_A & p_A^+ &= 2 \\ F_A^+(t) &= e(u_A(t), p_A^+, q_A) & T_A^+ &= 1 \end{aligned}$$

Falls $s_B < 0$ und $s_C < 0$, so gilt stattdessen:

$$\begin{aligned} u_B(t) &= -1 - t & u_B'(t) &= -1 & q_B &= -\frac{1}{2}s_B & p_B^+ &= 2 \\ F_B^+(t) &= e(u_B(t), p_B^+, q_B) & T_B^+ &= 2 \\ u_C(t) &= t & u_C'(t) &= 1 & q_C &= -\frac{1}{2}s_C & p_C^- &= 2 \\ F_C^-(t) &= e(u_C(t), p_C^-, q_C) & T_C^- &= -1 \end{aligned}$$

Falls $s_C < 0$ und $s_D < 0$, so gilt stattdessen:

$$\begin{aligned} u_D(t) &= t - 1 & u_D'(t) &= 1 & q_D &= -\frac{1}{2}s_D & p_D^- &= 2 \\ F_D^-(t) &= e(u_D(t), p_D^-, q_D) & T_D^- &= 0 \end{aligned}$$

Bei der Behandlung offener Splines werden weitere Tests vorgenommen:

- Bei der Behandlung des ersten Spline-Segmentes:

Ist $s_B = 0$ und $s_C < 0$ so wird verwendet:

$$\begin{array}{llll} u_B(t) = 1 - t & u'_B(t) = -1 & q_B = -\frac{1}{2}s_C & p_B^+ = 2 \\ F_B^+(t) = e(u_B(t), p_B^+, q_B) & T_B^+ = 2 & & \\ u_C(t) = t & u'_C(t) = 1 & p_C^- = 2 & \\ F_C^-(t) = f(u_C(t), p_C^-, q_C) & T_C^- = 0 & & \end{array}$$

- Bei der Behandlung des zweiten Segmentes:

Ist $s_A = 0$ und $s_B < 0$, so wird verwendet

$$\begin{array}{llll} u_A(t) = -t & u'_A(t) = -1 & q_A = -\frac{1}{2}s_B & p_A^+ = 2 \\ F_A^+(t) = e(u_A(t), p_A^+, q_A) & T_A^+ = 1 & & \end{array}$$

- Bei der Behandlung des vorletzten Segmentes:

Ist $s_C < 0$ und $s_D = 0$, so wird verwendet

$$\begin{array}{llll} u_D(t) = t - 1 & u'_D(t) = 1 & q_D = -\frac{1}{2}s_C & p_D^- = 2 \\ F_D^-(t) = e(u_D(t), p_D^-, q_D) & & & \end{array}$$

- Bei der Behandlung des letzten Segmentes:

Ist $s_B < 0$ und $s_C = 0$, wo wird verwendet

$$\begin{array}{llll} u_B(t) = 1 - t & u'_B(t) = -1 & p_B^+ = 2 & \\ F_B^+(t) = f(u_B(t), p_B^+) & T_B^+ = 1 & & \\ u_C(t) = t & u'_C(t) = 1 & q_C = -\frac{1}{2}s_B & p_C^- = 2 \\ F_C^-(t) = e(u_C(t), p_C^-, q_C) & T_C^- = -1 & & \end{array}$$

Annäherung von X-Splines durch Bezier-Splines

Die für fig2vect vorgesehenen Ausgabeformate unterstützen Bezier-Splines. Es ist erforderlich, X-Splines näherungsweise durch Bezier-Splines nachzubilden.

Zunächst werden Kurvenpunkte berechnet, die den Kontrollpunkten entsprechen (d.h. Kurvenpunkte mit $t = T_1, t = T_2 \dots$). Um die Kontrollpunkte für die Bezier-Splines zu errechnen, wird der Anstieg in den Kurvenpunkten benötigt. Da X-Splines Knicke in den Kurvenpunkten aufweisen können, die den Kontrollpunkten entsprechen, wird sowohl ein „linksseitiger“ als auch ein „rechtsseitiger“ Anstieg berechnet.

X-Splines sind Kurven fünften Grades – und weichen somit von Bezier-Splines dritten Grades ab – es ist deshalb sinnvoll, jedes X-Spline-Segment durch mehrere Bezier-Spline-Segmente darzustellen. Zu diesem Zweck werden Kurvenpunkte und Anstiege auch zwischen den im vorangegangenen Schritt berechneten Kurvenpunkten berechnet. Da hier keine Knicke auftreten, genügt jeweils eine Berechnung des Anstieges.

Erfolgt die Darstellung jedes X-Spline-Segmentes durch mehrere (m) Bezier-Spline-Segmente, ist eine Korrektur der Anstiege erforderlich. Bezier-Splines beginnen und enden jeweils bei $t = 0$ und $t = 1$. Die bisher errechneten Anstiege beziehen sich aber auf Bereiche mit $t = t_s$ und $t = t_s + \frac{1}{m}$. Die Verwendung der berechneten Kurvenpunkte als Anfangs- und Endpunkte von Bezier-Splines entspricht also einer Streckung entlang der t -Achse um den Faktor m . Damit geht eine Verringerung des Anstieges einher, alle bisher berechneten Anstiege müssen also korrigiert werden, indem sie durch m dividiert werden.

2.1.2 Kreisbögen

Kreisbögen sind in Fig-Dateien durch drei Punkte gegeben, die hier mit $A(x_A, y_A)$, $B(x_B, y_B)$ und $C(x_C, y_C)$ bezeichnet werden. Mit \vec{r}_A , \vec{r}_B und \vec{r}_C werden die Vektoren vom Koordinatenursprung zu den drei Punkten bezeichnet. A und C sind die Endpunkte des Bogens, B ist ein innerer Punkt.

Zum Zeichnen der Kreise und zur Berechnung der Bounding-Box werden der Mittelpunkt M und der Radius r benötigt.

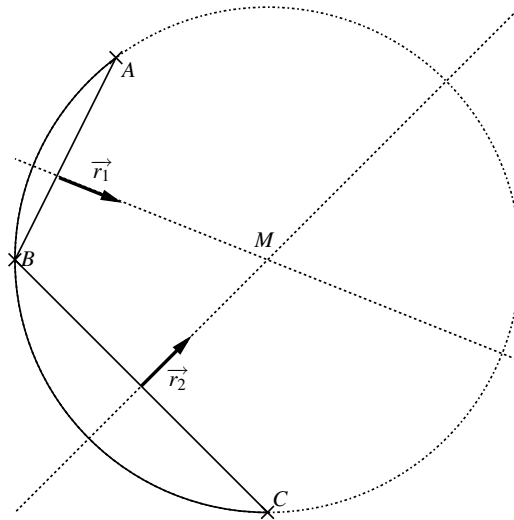


Abbildung 2.1: Berechnungen am Kreisbogen

Der Mittelpunkt $M(x_M, y_M)$ ergibt sich als Schnittpunkt der Mittelsenkrechten auf den Strecken \overline{BA} und \overline{BC}

$$g_1: \quad \vec{r} = \vec{r}_B + \frac{1}{2}\vec{BA} + t\vec{r}_1 \quad t \in \mathbb{R}$$

$$g_2: \quad \vec{r} = \vec{r}_B + \frac{1}{2}\vec{BC} + s\vec{r}_2 \quad s \in \mathbb{R}$$

Da die Mittelsenkrechten senkrecht auf den jeweiligen Strecken stehen, muß gelten:

$$0 = \vec{r}_1 \cdot \vec{AB}$$

$$0 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix}$$

$$0 = \vec{r}_2 \cdot \vec{CB}$$

$$0 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \begin{pmatrix} x_B - x_C \\ y_B - y_C \end{pmatrix}$$

2 Technische Details

Dies wird erfüllt durch:

$$x_1 = y_B - y_A$$

$$x_2 = y_B - y_C$$

$$y_1 = x_A - x_B$$

$$y_2 = x_C - x_B$$

Somit ergibt sich für die Geradengleichungen:

$$g_1 : \quad \vec{r} = \frac{1}{2} \begin{pmatrix} x_A + x_B \\ y_A + y_B \end{pmatrix} + t \begin{pmatrix} y_B - y_A \\ x_A - x_B \end{pmatrix}$$

$$g_2 : \quad \vec{r} = \frac{1}{2} \begin{pmatrix} x_C + x_B \\ y_C + y_B \end{pmatrix} + s \begin{pmatrix} y_B - y_C \\ x_C - x_B \end{pmatrix}$$

Im Schnittpunkt M beider Geraden gilt:

$$x_B + \frac{1}{2}(x_A - x_B) + t_M(y_B - y_A) = x_B + \frac{1}{2}(x_C - x_B) + s_M(y_B - y_C)$$

$$y_B + \frac{1}{2}(y_A - y_B) + t_M(x_B - x_A) = y_B + \frac{1}{2}(y_C - y_B) + s_M(x_B - x_C)$$

$$\frac{1}{2}(x_A + x_B) + t_M(y_B - y_A) = \frac{1}{2}(x_C + x_B) + s_M(y_B - y_C)$$

$$\frac{1}{2}(y_A + y_B) + t_M(x_B - x_A) = \frac{1}{2}(y_C + y_B) + s_M(x_B - x_C)$$

$$t_M(y_B - y_A) + s_M(y_C - y_B) = \frac{1}{2}(x_C - x_A)$$

$$t_M(x_A - x_B) + s_M(x_B - x_C) = \frac{1}{2}(y_C - y_A)$$

Dieses Gleichungssystem hat entweder

- genau eine Lösung

$$t_M = \frac{\frac{1}{2} \left((x_B - x_C)(x_C - x_A) - (y_C - y_B)(y_C - y_A) \right)}{(y_B - y_A)(x_B - x_C) - (y_C - y_B)(x_A - x_B)}$$

- keine Lösung (die Punkte liegen auf einer Geraden) oder
- unendlich viele Lösungen (zwei oder drei Punkte sind identisch).

2 Technische Details

zwei Schenkeln der Pfeilspitze) mit α , so erhält man:

$$\frac{l}{2} = s \cdot \sin\left(\frac{\alpha}{2}\right)$$
$$s = \frac{\frac{l}{2}}{\sin\left(\frac{\alpha}{2}\right)}$$

Sind Endpunkt P_E , Linienbreite l und Öffnungswinkel α vorgegeben, so muß der Punkt P_0 für das Zeichnen der Pfeilspitze den Abstand

$$s = \frac{\frac{l}{2}}{\sin\left(\frac{\alpha}{2}\right)}$$

zu P_E haben. Die Punkte P_1 und P_2 können dann über die Pfeilspitzenlänge und -breite berechnet werden.

Die eigentliche Linie – die noch vor der Pfeilspitze gezeichnet werden sollte – muß zwischen den Punkten P_A und P_B bzw. auf einem dieser Punkte enden. Die Entfernung von P_A zu P_E beträgt $2s$, die Entfernung von P_B zu P_E ergibt sich als c_{\min} zu:

$$\frac{\frac{l}{2}}{c_{\min}} = \tan\left(\frac{\alpha}{2}\right) \quad c_{\min} = \frac{\frac{l}{2}}{\tan\left(\frac{\alpha}{2}\right)} \quad c_{\min} \leq s$$
$$c_{\max} = 2s$$

$$c = \begin{cases} s & \text{für MetaPost-Treiber mit MetaPost-Pfeilspitzen} \\ \frac{1}{2}(c_{\min} + c_{\max}) & \text{sonst} \end{cases}$$

Längenkorrektur an Linien

Die Längenkorrektur um die Länge c an einer Linie von $P_0(x_0, y_0)$ nach $P_1(x_1, y_1)$ wird mit Hilfe folgender Formeln durchgeführt:

$$l = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \qquad q = \frac{l - c}{l}$$

für nichtnegative q gilt dann:

$$x'_1 = x_0 + q(x_1 - x_0) \qquad y'_1 = y_0 + q(y_1 - y_0)$$

Längenkorrektur an Kreisbögen

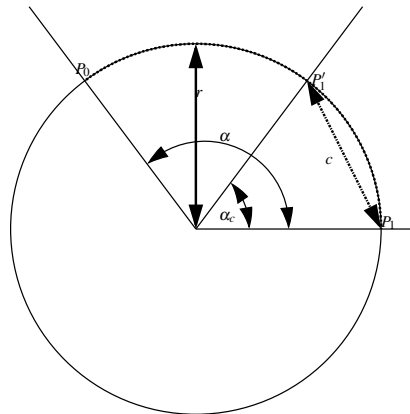


Abbildung 2.3: Korrektur am Kreisbogen

Ist ein Kreisbogen von P_0 nach P_1 durch den Radius r und Winkel α gegeben und soll so verkürzt werden, daß c die Entfernung zwischen dem neuen Endpunkt P_1' und dem alten Endpunkt P_1 ist, so kann die Winkelverkürzung α_c folgendermaßen berechnet werden:

$$\begin{aligned}
 c &= \sqrt{(x_1' - x_1)^2 + (y_1' - y_1)^2} \\
 &= \sqrt{(r \cos \alpha_c - r)^2 + (0 - r \sin \alpha_c)^2} \\
 &= \sqrt{r^2 - 2r^2 \cos \alpha_c + r^2 \cos^2 \alpha_c + r^2 \sin^2 \alpha_c} \\
 &= r \sqrt{1 - 2 \cos \alpha_c + \cos^2 \alpha_c + \sin^2 \alpha_c} \\
 &= r \sqrt{2 - 2 \cos \alpha_c} \\
 \frac{c}{r} &= \sqrt{2 - 2 \cos \alpha_c} \\
 2 - 2 \cos \alpha_c &= \left(\frac{c}{r}\right)^2 \\
 2 \cos \alpha_c &= 2 - \left(\frac{c}{r}\right)^2 \\
 \cos \alpha_c &= 1 - \frac{c^2}{2r^2} \\
 \alpha_c &= \arccos\left(1 - \frac{c^2}{2r^2}\right)
 \end{aligned}$$

Längenkorrektur an Splines

Die Längenkorrektur an Splines ist vergleichsweise schwierig, da hier die Formeln nicht analytisch so umgestellt werden können, daß man ein t'_1 erhält.

Es muß daher ein Näherungsverfahren eingesetzt werden, fig2vect verwendet das Newton-Iterationsverfahren.

Für ein gegebenes t beträgt der Abstand zum Endpunkt P_1 :

$$d(t) = \sqrt{(x(t) - x_1)^2 + (y(t) - y_1)^2}$$

Gesucht wird die Nullstelle von

$$\begin{aligned} f(t) &= d(t) - c \\ &= \sqrt{(x(t) - x_1)^2 + (y(t) - y_1)^2} - c \\ f'(t) &= \frac{1}{2\sqrt{(x(t) - x_1)^2 + (y(t) - y_1)^2}} \cdot \frac{\partial}{\partial t} \left((x(t) - x_1)^2 + (y(t) - y_1)^2 \right) \\ &= \frac{2(x(t) - x_1) \cdot x'(t) + 2(y(t) - y_1) \cdot y'(t)}{2\sqrt{(x(t) - x_1)^2 + (y(t) - y_1)^2}} \\ &= \frac{(x(t) - x_1) \cdot x'(t) + (y(t) - y_1) \cdot y'(t)}{\sqrt{(x(t) - x_1)^2 + (y(t) - y_1)^2}} \end{aligned}$$

Für ein gegebenes t kann ein $u(t)$ ermittelt werden mit

$$\begin{aligned} 0 &\leq u(t) \leq 1 \\ u(t) &= t - T_l \end{aligned}$$

2 Technische Details

so daß $u(t)$ zu Berechnungen im Bezier-Spline zwischen den Punkten P_l und P_r verwendet werden kann.

$$\begin{aligned}
 x(u) &= (1-u)^3 x_l + 3(1-u)^2 u x_{l+} + 3(1-u)u^2 x_{r-} + u^3 x_r \\
 y(u) &= (1-u)^3 y_l + 3(1-u)^2 u y_{l+} + 3(1-u)u^2 y_{r-} + u^3 y_r \\
 \frac{\partial x}{\partial u} &= 3u^2(x_r - 3x_{r-} + 3x_{l+} - x_l) + 2u(3x_{r-} - 6x_{l+} + 3x_l) + 3x_{l+} - 3x_l \\
 x'(t) &= \frac{\partial x}{\partial t} = \frac{\partial x}{\partial u} \cdot \frac{\partial u}{\partial t} \\
 &= 3u^2(x_r - 3x_{r-} + 3x_{l+} - x_l) + 2u(3x_{r-} - 6x_{l+} + 3x_l) + 3x_{l+} - 3x_l \\
 y'(t) &= \frac{\partial y}{\partial t} = 3u^2(y_r - 3y_{r-} + 3y_{l+} - y_l) + 2u(3y_{r-} - 6y_{l+} + 3y_l) + 3y_{l+} - 3y_l
 \end{aligned}$$

Das Newton-Verfahren beginnt mit einem Startwert t_0 , jeder Folgewert wird dann mittels

$$t_{n+1} = t_n - \frac{f(t_n)}{f'(t_n)}$$

berechnet. Die Berechnung wird mit Fehler abgebrochen, falls t aus dem Definitionsbereich $0 \leq t \leq (B-1)$ hinausläuft oder die maximale Schrittzahl erreicht wurde. B ist die Anzahl der Bezier-Punkte für das Spline.

Die Berechnung wird erfolgreich beendet, wenn die Bedingungen

$$|f(t)| < \varepsilon \qquad |t_n - t_{n-1}| < \varepsilon$$

erfüllt sind und gleichzeitig die minimale Schrittzahl erreicht wurde.

Ein Problem beim Newton-Verfahren ist das Finden des geeigneten Startwertes t_0 . Direkt von den Endpunkten $t = 0$ bzw. $t = B-1$ auszugehen, ist nicht möglich, da hier jeweils

$$\frac{\partial x}{\partial t} = 0 \qquad \frac{\partial y}{\partial t} = 0 \qquad \frac{\partial f}{\partial t} = 0$$

gilt. Auch in der „näheren“ Umgebung der Endpunkte sind die Ableitungen noch immer so klein, daß das Verfahren nicht konvergiert bzw. die t aus dem Definitiosbereich herauslaufen.

Der Startwert t_0 wird daher folgendermaßen gebildet:

- Es wird das t_x gesucht, das dem vorletztem (bzw. dem zweiten) Kontrollpunkt des X-Splines entspricht.

2.1 Mathematische Aspekte

- Für dieses t_x wird der Abstand des Punktes zum Endpunkt berechnet.
- Mit diesem t_x , d_x sowie dem geforderten c wird der Startwert t_0 durch lineare Approximation (Dreisatz) ermittelt.

2.2 Notizen

2.2.1 Font-Handling

Die Komponente *handling* der *dk_fig_fonth_t*-Struktur legt fest, wie der Font für einen Text anzugeben ist.

Folgende Werte sind möglich:

- 0
handling=none, font=tex
Der Text wird nicht durch T_EX/L^AT_EX verarbeitet, es wird aber der L^AT_EX-Fontname verwendet.
Für den MetaPost-Treiber bewirkt dies
... infont "ptmr" scaled ...
- 1
handling=none, font=fig
Der Text wird nicht durch T_EX/L^AT_EX verarbeitet, es wird der PS-Fontname verwendet.
Für den MetaPost-Treiber bewirkt dies
... infont "TimesRoman" scaled ...
- 2
handling=tex, font=tex
Der Text wird durch T_EX/L^AT_EX verarbeitet, es werden keine Angaben zu Font und Schriftgröße gemacht. D. h., es wird mit der Standard-L^AT_EX-Schrift geschrieben.
- 3
handling=tex, font=fig
Der Text wird durch T_EX/L^AT_EX verarbeitet, Font und Schriftgröße werden entsprechend den Angaben der Fig-Datei gewählt.
`\font\fntAA=ptmr at 12pt`
- 4
handling=tex, font=similar, size=fig
Der Text wird durch T_EX/L^AT_EX verarbeitet, es wird eine zum Rest des Dokumentes passende L^AT_EX-Schriftart gewählt, die in ihren Features (roman/sans-serif/monospaced, normal/fett, aufrecht/italic) zu den Angaben in der Fig-Datei passt. Die Größe wird entsprechend der Fig-Datei gewählt.
- 5
handling=tex, font=similar, size=tex

Der Text wird durch $\text{T}_{\text{E}}\text{X}/\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ verarbeitet, es wird eine zum Rest des Dokumentes passende $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ -Schriftart gewählt, die in ihren Features (roman/sans-serif/monospaced, normal/fett, aufrecht/italic) zu den Angaben in der Fig-Datei passt. Die Größeneinstellung wird $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ überlassen, d. h. die Schrift wird genauso groß wie im Rest des Dokumentes gewählt.