

10. Das MTP-Programmpaket

Proseminar Mathematik am Computer

Volker Ziesing

17. Juli 2006

- 1 Einleitung
- 2 Textfunktionen
- 3 Einbinden von Grafiken
- 4 Grafikfunktionen
- 5 Anhang

Inhalt

- 1 **Einleitung**
 - Das Akronym MTP
 - MTP initialisieren
- 2 Textfunktionen
- 3 Einbinden von Grafiken
- 4 Grafikfunktionen
- 5 Anhang

Das Akronym MTP

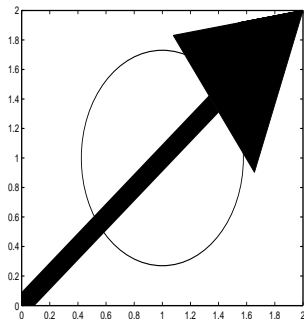
- **MTP** steht für **Matlab**, **TEX** und **Postscript**
- Verbindet *Grafikfähigkeit* von `MATLAB` mit Satzsystem `LATEX`
- Erzeugt typographisch hochwertige Dokumente im Postscript-Format
- *Beschriftung* von Grafiken formatiert anschliessend `LATEX`,
- Komplette Funktionalität von `LATEX` steht zur Verfügung
- Neue **Grafikobjekte** wie zum Beispiel Pfeile (`mtparrow`)
- *Aliaskonzept* vereinfacht den Umgang mit graphischen Objekten

Initialisierung

- Der Befehl `mtpinit` startet das MTP-Paket
- Der Status wird auf `hold on` gesetzt
- Folgende **Parameter** stehen zur Auswahl
(in den Klammern sind die Defaultwerte):
 - ▶ `mtpextdistanceunit`
= Abstand vom Text zum Referenzpunkt (0,3cm)
 - ▶ `LineWidth` = Standardbreite für Linien (0.5)
 - ▶ `MarkerSize` = Standardgröße für MTP-Marker (6)
 - ▶ `ArrowShape=[ArrowLength, ArrowHeadAngle]`
= Länge und Öffnungswinkel der Pfeilspitze ([16 40])

Beispiel

```
>> mtpinit (NaN, 20, 200, [180, 60])  
>> mtpmarker(1, 1)  
>> mtparrow([0, 2], [0, 2])
```



Inhalt

1 Einleitung

2 Textfunktionen

- Einführung
- mptext
- mptexaxes
- mptitle
- mtpshow
- mtpshow

3 Einbinden von Grafiken

4 Grafikfunktionen

5 Anhang

Einführung

Wir behandeln jetzt die MTP-Textfunktionen zum Erzeugen von \LaTeX -Text und deren Nachbearbeitung.

- Den Kern der Textfunktionen übernimmt `mptext`
- Beschriften von Koordinatenachsen und dem Titel der Grafik übernehmen die Befehle `mptexaxes` bzw. `mptitle`
- `mptshow` ermöglicht eine Vorschau der resultierenden Grafik
- MTP-Grafiken die Text-Objekten enthalten sind mit `mptsave tex` oder `mptsave eps` abzuspeichern

mptext

Aufgabe: Setzt \LaTeX -Text in die Grafik

Folgende **Parameter** stehen zur Auswahl:

`mptext(x, y, z, TextString, Alignment, TextD.F., Rotation)`

- x, y, z = horizontale Vektoren, welche die x -, y - und z -Koordinaten der Referenzpunkte enthalten. z ist optional.
- `TextString` = schreibt \LaTeX -Strings an den Referenzpunkt
- `Alignment` = verändert die Textausrichtung zum Referenzpunkt
- `TextDistanceFactor` = orientierter Abstand zwischen Text und Referenzpunkt
- `Rotation` = dreht Text um einen bestimmten Winkel

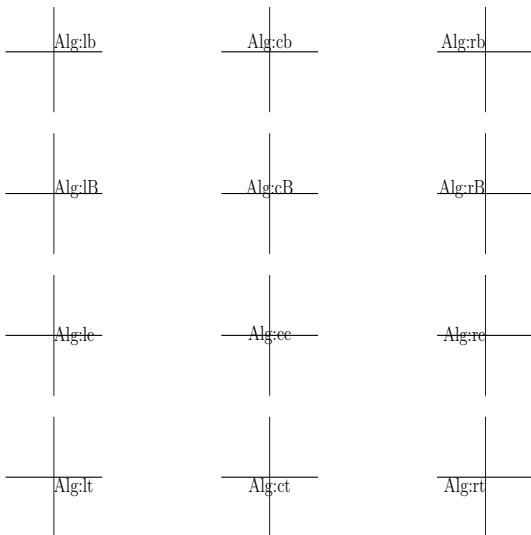
Beispiel zu mtp_text

Beispiel

```
>> mtpinit
>> valign={'t','c','B','b'};
>> halign={'l','c','r'};
>> axis off

>> for hind=1:length(halign)
>>     for vind=1:length(valign)
>>         mtpmarker(hind,vind,'plus',{'markersize',100})
>>         alignment=[halign{hind} valign{vind}];
>>         mtp_text(hind,vind,['Alg:' alignment],alignment)
>>     end
>> end
>> mtpshow(12)
```

Grafik zu mptext



mptexaxes

Aufgabe: Beschriftet die Koordinatenachsen mit \LaTeX -Text

Folgende **Parameter** stehen zur Auswahl:

```
mptextaxes(xTick, xTickLabel, ..., zTick, zTickLabel)
```

Für diese Parameter wird der Defaultwert NaN angenommen

Die **Achsenmarkierungen** (Ticks) werden wie folgt angegeben:

- [...] = Vektor mit streng monoton wachsenden Werten
- NaN = aktuellen Ticks werden beibehalten
- {} = betreffende Achse wird beibehalten
- `mptextaxes(..., DistanceFactor)` = skaliert den Abstand zwischen Achse und Beschriftung. `DistanceFactor` kann eine Zahl, ein zweielementiger- oder ein dreielementiger Vektor sein.
`mptextdistanceunit` ist hierbei die Maßeinheit für die Skalierung im TEX-Dokument

mptexaxes

Die **Schriftzüge** (Labels) werden wie folgt übergeben:

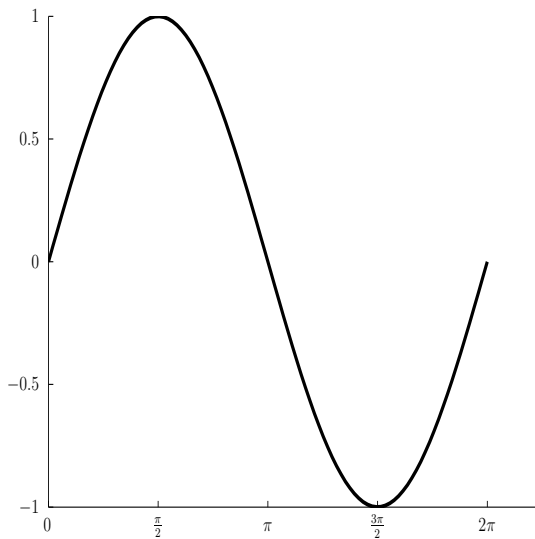
- `[...]` = setzt `Label(i)` an `Tick(i)`. Beide Vektoren müssen gleiche Länge haben
- `{...}` = wie oben, nur das `Label(i)` eine Zahl, NaN oder ein String sein kann
- `'...'` = beschriftet alle Ticks mit einem String
- `NaN` = verwendet den Tick-Vektor zur Beschriftung

Beispiel zu mptexaxes

Beispiel

```
>> mtpinit
>> t=linspace(0,2*pi);
>> mtpline(t,sin(t),'Thick')
>> mptexaxes([0:.5*pi:2*pi],{0 '$\frac{\pi}{2}$' ...
    '$\pi$' '$\frac{3\pi}{2}$' '$2\pi$'}, ...
    [-1:.5:1],NaN,[0 .5])
>> mtpshow(12)
```

Grafik zu mptexaxes



mptitle

Aufgabe: Setzt Titel und Bezeichnung der Achsen mit \LaTeX -Text

Folgende **Parameter** stehen zur Auswahl:

```
mptitle(Titel, xLabel [, xDistanceFactor], ...  
        yLabel [, yDistanceFactor], ...  
        zLabel [, zDistanceFactor])
```

- `Title` = schreibt \LaTeX -String `Title` an die Titelposition. Übergabe von `{}` modifiziert den aktuellen Titel nicht
- `Title, ...` = setzt die zu übergebenden `Label`. Übergabe von `{}` modifiziert die betreffenden Achsen nicht
- `DistanceFactor` = skaliert den Abstand zwischen Referenzpunkt und dem Textobjekt.

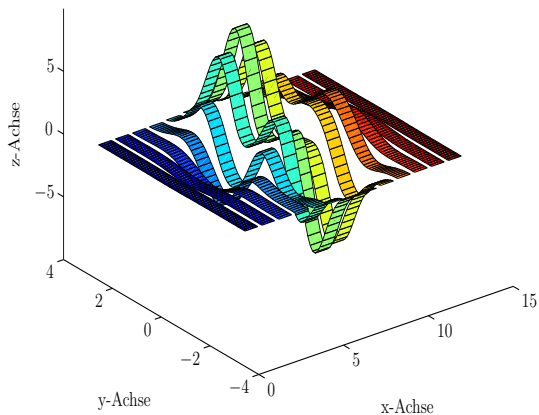
Beispiel zu mptexaxes

Beispiel

```
>> mtpinit
>> view(3)
>> [x,y]=meshgrid(-3:.5:3,-3:.1:3);
>> z=peaks(x,y);
>> ribbon(y,z)
>> mptexaxes(NaN,NaN,NaN,NaN,[-5 0 5],NaN)
>> mptitle('\large Peaks and Stripes', ...
           'x-Achse','y-Achse','z-Achse',1)
>> mtpshow(12)
```

Grafik zu mptexaxes

Peaks and Stripes



mtpshow

Aufgabe: Zeigt die vollständige Grafik inklusive \LaTeX -Text

Mit `MTP` erzeugte \LaTeX -Elemente können im Grafikenster nicht angezeigt werden. Erst nach dem Aufruf von `mtpsavetex` oder `mtpsaveeps` erhält man das gewünschte Resultat, daher stellt die Routine `mtpshow` eine Vorschau bereit.

Folgende **Parameter** stehen zur Auswahl:

`mtpshow (FontSize, Options, mtpdistanceunit)`

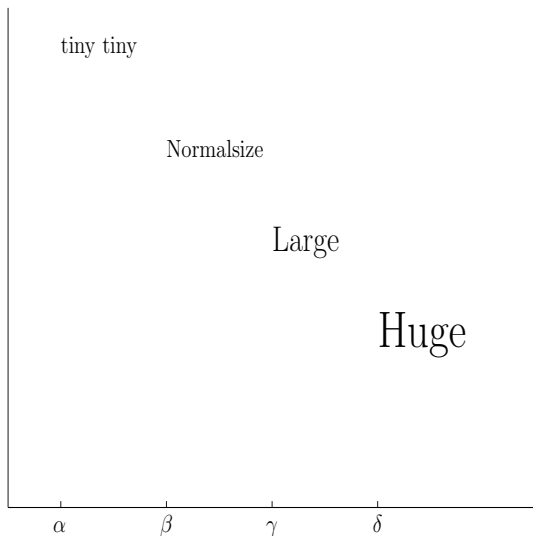
- `FontSize` = gibt die \LaTeX -Fontgröße an (10pt, 11pt oder 12pt)
- `Options, mtpdistanceunit` = einstellen von Skalierung der Grafik

Beispiel zu mtpshow

Beispiel

```
>> mtpinit
>> K=[0.1 0.3 0.5 0.7
      0.9 0.7 0.5 0.3]
>> mptexaxes(K(1,:),...
             {'$\alpha$', '$\beta$', '$\gamma$',...
             '$\delta$'}, [], {})
>> mptext(K, {'{tiny tiny}', '{\normalsize ...
             Normalsize}', '{\Large Large}', ...
            '{\Huge Huge}'}, 'lb')
>> mtpshow(12, 'width=0.8\linewidth')
```

Grafik zu mtpshow



Inhalt

- 1 Einleitung
- 2 Textfunktionen
- 3 Einbinden von Grafiken**
 - mtpsaveeps
 - mtpsavetex
 - mtpincludegraphics
- 4 Grafikfunktionen
- 5 Anhang

mtpsaveeps

Aufgabe: Erzeugt eine vollständige EPS-Grafik

Vorteile dieser Routine:

- Grafik ist unabhängig von `MTP` und `LATEX`
- Grafik kann mit `LATEX`-Paketten `epsfig` und `graphicx` eingebunden werden
- Die Fontgröße wird proportional mit der Grafikgröße skaliert.

Nachteile dieser Routine:

- `LATEX`-Beschriftung verliert unter Skalierungen die Qualität
- `LATEX`-Beschriftung wird bei ungleicher x - und y -Skalierung verzerrt.
- Die Fontgröße der Beschriftung passt nach Skalierung nicht mehr zur Dokumentschriftgröße
- Der Fonttyp ist vom Dokument unabhängig

mtpsavetex

Aufgabe: Trennt die Beschriftung von der EPS-Grafik

Es werden nur Grafikobjekte und Positionen der Textobjekte in der EPS-Grafik abgelegt, während die eigentlichen Textobjekte in einer TEX-Datei landen.

Vorteile dieser Routine:

- Einheitliches Bild von Text und Grafikbeschriftung
- Beschriftung ist von der Skalierung der Grafik unabhängig
- Textobjekte können nachträglich modifiziert werden

Nachteile dieser Routine:

- Benötigt das Paket `mtp.sty`. Folglich ist in der TEX-Datei die Zeile `\usepackage[<options>]{mtp}` einzubinden
- Feste Positionen der Beschriftung können unter extremer Skalierung zu unschönen Grafiken führen

mtpincludegraphics

Aufgabe: \LaTeX -Befehl zum Einbinden von MTP-Grafiken

Folgende **Syntax** ist vorgeschrieben:

```
\mtpincludegraphics [Optionen] {Bilddatei}
```

- **Optionen** = die selben Optionen wie beim \LaTeX -Befehl `\includegraphics` stehen zur Verfügung. Beispielsweise skaliert `[width=.8\linewidth]` das Bild auf 80% der Zeilenbreite
- **Bilddatei** = Im Gegensatz zum \LaTeX -Befehl `\includegraphics` ist darauf zu achten, die Dateierweiterung im Bildnamen nicht mit anzugeben

Inhalt

- 1 Einleitung
- 2 Textfunktionen
- 3 Einbinden von Grafiken
- 4 Grafikfunktionen**
 - mtpline
 - mtpsegments
 - mtploop
 - mtparc
 - mtparea
 - mtpmarker
 - mtparrow
- 5 Anhang

mtpline

Aufgabe: Zeichnet einen Polygonzug

Folgende **Parameter** stehen zur Auswahl:

```
mtpline=(X,Y,Z [Alias1, Alias2,...])
```

- X, Y, Z = Polygonzug zwischen den Ecken $(X(1, i), Y(1, i), Z(1, i))$
- $Alias1, Alias2, \dots$ = Aliase sind z.B. die **Farbe** ('black'), der **Linientyp** ('solid') und die **Größe** der Linie ('thick')

Die Reihenfolge der optionalen Aliase spielt keine Rolle.

Beispiel zu mtpline

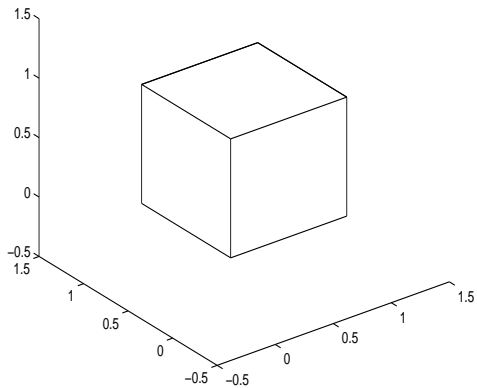
Beispiel

```
>> mtpinit
% Dreidimensionale Blickrichtung
>> view(3)
>> axis([-0.5 1.5 -0.5 1.5 -0.5 1.5])

% Würfeleckpunkte definieren
>> P=[0 1 1 0 0 1 1 0
      0 0 1 1 0 0 1 1
      0 0 0 0 1 1 1 1];

% Kanten zeichnen
>> mtpline(P(:, [1 2 6 7 8 5 1]));
>> mtpline(P(:, [5 6 7 8 4 1]));
```

Grafik zu mtp\line



mtpsegments

Aufgabe: Zeichnet nicht zusammenhängende Geradensegmente

Folgende **Parameter** stehen zur Auswahl:

`mtpsegments (X1, Y1, Z1, X2, Y2, Z2 [, Alias1, Alias2, ...])`

- `X1, Y1, Z1, X2, Y2, Z2` = zeichnet eine gerade Linie zwischen dem i-ten Punkt des ersten Datenblocks und dem i-ten Punkt des zweiten Datenblocks

`mtpsegments` verwendet die selben Aliase wie `mtpline`

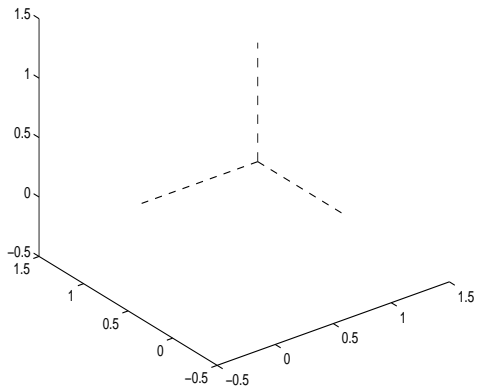
Beispiel zu mtpline

Beispiel

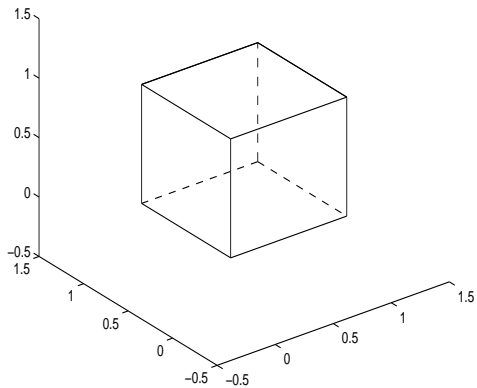
```
>> mtpinit
>> view(3)
>> axis([-0.5 1.5 -0.5 1.5 -0.5 1.5])
>> P=[0 1 1 0 0 1 1 0
      0 0 1 1 0 0 1 1
      0 0 0 0 1 1 1 1];

% Zeichnet vom 3-ten Punkt aus drei Segmente
>> mtpsegments(P(:, [3 3 3]),P(:, [2 7 4]),'dashed');
```

Grafik zu mtpsegments



vollständige Grafik



mtploop

Aufgabe: Zeichnet einen geschlossenen Polygonzug

Folgende **Parameter** stehen zur Auswahl:

```
mtploop(X,Y,Z [,Alias1,Alias2,...])
```

- X, Y, Z = zeichnet wie bei `mtpline` eine gerade Linie zwischen dem i -ten Punkt des ersten Datenblocks und dem i -ten Punkt des zweiten Datenblocks. Der einzige Unterschied zu `mtpline` ist, dass eine schließende Kante miteinbezogen wird.

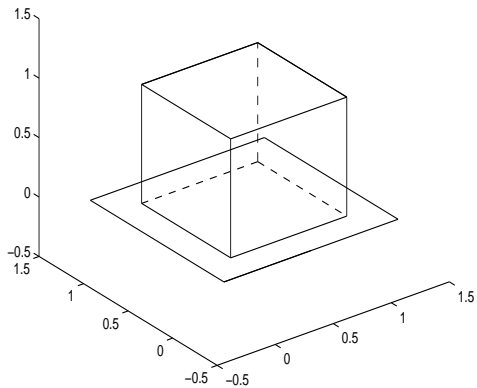
`mtploop` verwendet die selben Aliase wie `mtpline`

Beispiel zu mtploop

Beispiel

```
>> mtpinit
>> view(3)
>> axis([-0.5 1.5 -0.5 1.5 -0.5 1.5])
>> P=[0 1 1 0 0 1 1 0
      0 0 1 1 0 0 1 1
      0 0 0 0 1 1 1 1];
>> mtpline(P(:, [1 2 6 7 8 5 1]));
>> mtpline(P(:, [5 6 7 8 4 1]));
>> mtpsegments(P(:, [3 3 3]), P(:, [2 7 4]), 'dashed');
>> F=[-0.25  1.25  1.25 -0.25
      -0.25 -0.25  1.25  1.25
       0      0      0      0 ];
>> mtploop(F);
```

Grafik zu mtploop



mtparc

Aufgabe: Zeichnet Kreissegmente und beschriftet Winkel

Folgende **Parameter** stehen zur Auswahl:

`mtparc(Centre, rad1, rad2, radius, text_pos, TextString, Aliase)`

- `Centre` = setzt den Kreismittelpunkt auf $[x, y]$
- `rad1, rad2` = Randsegment wird zwischen beiden Winkeln im Bogenmaß gezeichnet
- `radius` = Radius des positiv orientierten Randsegments
- `TextDistanceFactor` = Abstand zwischen Text und Referenzpunkt
- `text_pos` = enthält den auf $[0, 1]$ normierten Vektor mit den Positionierungsdaten für den Referenzpunkt der Beschriftung. Dabei wird die radiale Verschiebung und die Winkelverschiebung definiert. 0 entspricht `rad1` und 1 entspricht `rad2`. Default ist `text_pos=[0.75 0.5]`
- `TextString` = setzt Text an durch `text_pos` definierten Referenzpunkt

Beispiel zu mtparc

Beispiel

```
% Dreieck mit den Seitenlängen 3,4 und 5
>> mtpinit
>> P=[0 5 3.2
>>     0 0 2.4]
>> mtploop(P)

% Gestrichelten Thales-Halbkreis mit Radius 2,5
>> mtparc([2.5 0],0,pi,2.5,'dashed')

% rechten Winkel einzeichnen und beschriften
>> mpos=mtparc(P(:,3),P(:,1)-P(:,3),...
    P(:,2)-P(:,3),.4,[.5 .5]);
```

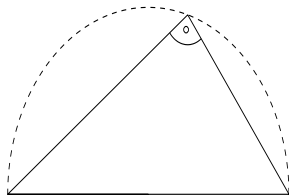
Grafik zu mtparc

Beispiel

```
>> mtpmarker(mpos, 'circle')
```

```
% Koordinatenachsen anpassen
```

```
>> axis equal, axis off
```



mtparea

Aufgabe: Zeichnet einen gefüllten Polygonzug

Folgende **Parameter** stehen zur Auswahl:

```
mtparea(X,Y,Z [,Alias1,Alias2,...])
```

- X, Y, Z = zeichnet einen mit einer beliebigen Farbe gefüllten Polygonzug. Die Eckpunkte werden mit X, Y, Z definiert und es wird eine geschlossene Kante wie bei `mtploop` automatisch eingefügt
- $Alias1, Alias2, \dots$ = Vordefinierte Aliase sind die **Farbe** ('gray'), die **Farbe der Berandung** ('rededge') und die **Dicke der Berandung**('thick')

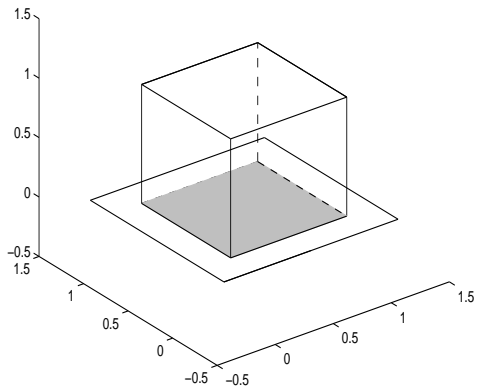
Die Reihenfolge der Aliase ist hier wichtig . Es muss zuerst die Farbe und dann die Farbe der Berandung angegeben werden!

Beispiel zu mtparea

Beispiel

```
>> mtpinit
>> view(3)
>> axis([-0.5 1.5 -0.5 1.5 -0.5 1.5])
>> P=[0 1 1 0 0 1 1 0
      0 0 1 1 0 0 1 1
      0 0 0 0 1 1 1 1];
[... ]
>> F=[-0.25  1.25  1.25 -0.25
      -0.25 -0.25  1.25  1.25
      0      0      0      0 ];
>> mtploop(F);
>> mtparea(P(:, [1 2 3 4]), 'gray75');
```

Grafik zu mtparea



mtpmarker

Aufgabe: Zeichnet Markersymbole

Folgende **Parameter** stehen zur Auswahl:

`mtpmarker(X,Y,Z [,Alias1,Alias2,...])`

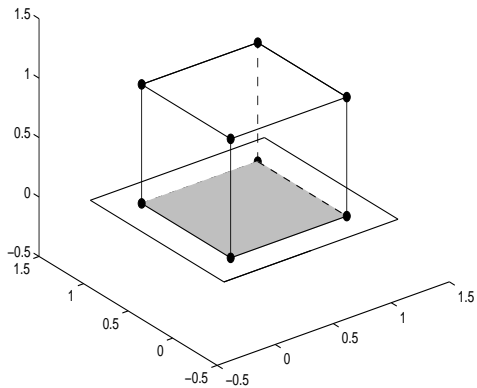
- `X,Y,Z` = zeichnet Marker in die durch `X,Y,Z` gegebenen Punkte
- `Alias1,Alias2,...` = Vordefinierte Aliase sind die **Farbe** (`'solidblack'`), die Art des **Markersymbols** (`'circle'`) und die **Größe** des Symbols (`'normalsize'`)

Beispiel zu mtpmarker

Beispiel

```
>> mtpinit
>> view(3)
>> axis([-0.5 1.5 -0.5 1.5 -0.5 1.5])
>> P=[0 1 1 0 0 1 1 0
      0 0 1 1 0 0 1 1
      0 0 0 0 1 1 1 1];
[... ]
>> mtploop(F);
>> mtparea(P(:, [1 2 3 4]), 'gray75');
>> mtpmarker(P, 'solidblack');
```

Grafik zu mtpmarker



mtparrow

Zeichnet einen Pfeil aus einem Polygonzug und einer Pfeilspitze

Folgende **Parameter** stehen zur Auswahl:

`mtparrow(X,Y,Z [,Alias1,Alias2,...])`

- `X,Y,Z` = zeichnet einen Pfeil in Richtung des vorletzten Punktes, dabei werden die Koordinaten `X,Y,Z` des Polygonzugs wie bei `mtpline` übergeben
- `Alias1, Alias2, ...` = Vordefinierte Aliase sind
 - ▶ die **Farbe** (`'black'`),
 - ▶ die **Dicke** des Polygonzugs (`'thick'`),
 - ▶ die **Art** des Polygonzugs (`'withtail'`),
 - ▶ das **Verkürzen** des Polygonzugs (`'shortentail'`),
 - ▶ die **Gestalt** der Pfeilspitze (`'noslighthead'`),
 - ▶ die **Form** der Pfeilspitze (`'normalanglehead'`),
 - ▶ die **Länge** der Pfeilspitze (`'normallengthhead'`)
- `mtparrow('redraw')` führt zu einer Anpassung an die aktuellen Achsen

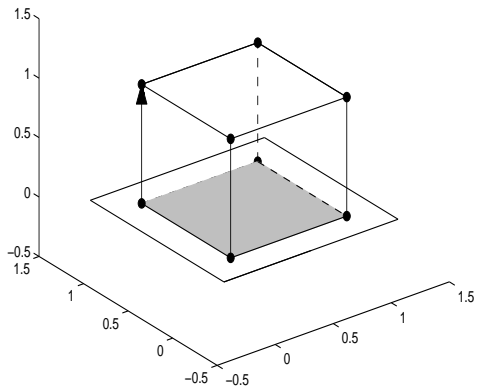
Beispiel zu mtparrow

Beispiel

```
>> mtpinit
>> view(3)
>> axis([-0.5 1.5 -0.5 1.5 -0.5 1.5])
>> P=[0 1 1 0 0 1 1 0
      0 0 1 1 0 0 1 1
      0 0 0 0 1 1 1 1];
[...]
```

```
>> mtpmarker(P,'solidblack');
>> mtparrow([0 0; 1 1; 1 1]);
```

Grafik zu mtparrow



Inhalt

- 1 Einleitung
- 2 Textfunktionen
- 3 Einbinden von Grafiken
- 4 Grafikfunktionen
- 5 Anhang**
 - Anwendung Potentialströmungen
 - Links

Potentialströmungen

Durch die Annahmen der Inkompressibilität und Wirbelfreiheit ergeben sich für ebene stationäre Strömungen die aus der Funktionentheorie bekannten Cauchy-Riemann-Differentialgleichungen. Die resultierenden Potentialströmungen sind mittels `MTP` innerhalb `MATLAB` darstellbar.

Auf den folgenden Seiten wird das Umströmen eines Zylinders (Beispiel 1) und von Ecken (Beispiele 2 und 3) beschrieben.

Beispiel Zylinder (1. Seite)

Beispiel

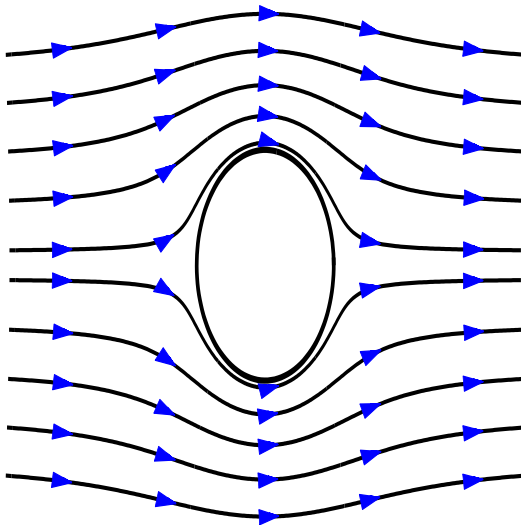
```
>> for k = 0.06:0.2:1
>> z = linspace(-2, 2, 301) + k*i;
>> h = sqrt((z - 1)./(z + 1));
>> w = (1 + h)./(1 - h);
>> plot(w, 'linewidth', 2);
>> xlim([-4 4]);
>> ylim([-2 2]);
>> idx = find(abs(real(w) + 3) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) + 1.5) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) + 0) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) - 1.5) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) - 3) < 2e-1);
```

Beispiel Zylinder (2. Seite)

Beispiel

```
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> w = w - 2*imag(w)*i;
>> plot(w, 'linewidth', 2);
>> idx = find(abs(real(w) + 3) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) + 1.5) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) + 0) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) - 1.5) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> idx = find(abs(real(w) - 3) < 2e-1);
>> mtparrow([real(w(idx)); imag(w(idx))], 'b');
>> end
>> plot(cos(0:0.05:2*pi+0.1), sin(0:0.05:2*pi+0.1), ...
        'k', 'linewidth', 3);
```

Grafik zu Zylinder

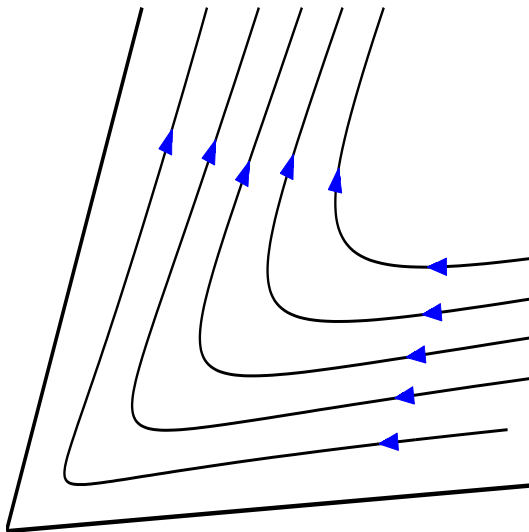


1. Beispiel Ecke

Beispiel

```
>> theta = 0.2:0.01:pi/2-0.2;
>> for k = 0.6:0.7:4
>> rho = k*(sin(pi*theta./(4*pi/9))).^(-9/4);
>> [x, y] = pol2cart(theta, rho);
>> plot(x, y, 'linewidth', 2);
>> xlim([0 4]);
>> ylim([0 4]);
>> idx = find(abs(y - 3) < 3e-1 & x < 3);
>> mtpararrow([x(idx(1:2)); y(idx(1:2))], 'b');
>> idx = find(abs(x - 3) < 3e-1 & y < 3);
>> mtpararrow([x(idx(1:2)); y(idx(1:2))], 'b');
>> end
>> plot([0 4], [0 4*sin(pi/36)], 'k', 'linewidth', 3);
>> plot([0 4*sin(3*pi/36)], [0 4], 'k', 'linewidth', 3);
```

1. Grafik zu Ecke

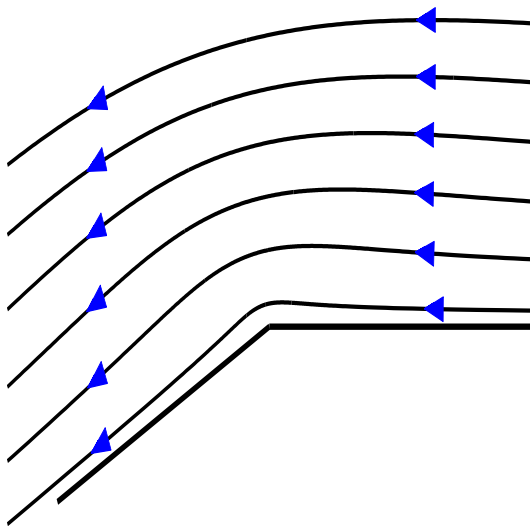


2. Beispiel Ecke

Beispiel

```
>> theta = 0.01:0.01:1.2*pi;
>> for k = 0.3:0.7:4
>>   rho = k*(sin(pi*theta./(1.2*pi))).^(-1/1.2);
>>   [x, y] = pol2cart(theta, rho);
>>   plot(x, y, 'linewidth', 2);
>>   xlim([-4 4]);
>>   ylim([-5 5]);
>>   idx = find(abs(x + 2.5) < 3e-1);
>>   mtparrow([x(idx); y(idx)], 'b');
>>   idx = find(abs(x - 2.5) < 3e-1);
>>   mtparrow([x(idx); y(idx)], 'b');
>> end
>> plot([0 4], [0 0], 'k', 'linewidth', 3);
>> plot([0 4*cos(1.2*pi)], [0 4*sin(1.2*pi)], ...
        'k', 'linewidth', 3);
```

2. Grafik zu Ecke



Links zu MTP

- Die Dokumentation zum MTP-Paket:

<http://www-old.mathematik.uni-stuttgart.de/mathA/1st2/Software/MTPdokumentation.pdf>

- Die Software herunterladen:

<http://www-old.mathematik.uni-stuttgart.de/mathA/1st2/Software/mtp-rell1.1.2.zip>

- MATLAB-Programmpaket zum Zeichnen von Quadriken
(Beispielaufruf: kegel)

<http://www.mathematik.uni-stuttgart.de/studium/infomat/HM-Hollig-SS05/quadrik.zip>

Vielen Dank
für die
Aufmerksamkeit