Eine sehr kurze Einführung in die Partial Differential Equation Toolbox von MATLAB *

Heinrich Voss†

Zusammenfassung

Dieser Kurs ist eine sehr kurze Einführung in die Partial Differential Equation Toolbox für MATLAB. An fünf Beispielen zeigen wir einige Möglichkeiten der PDE Toolbox zur Beschreibung, Lösung und Visualisierung von partiellen Differentialgleichungen auf. Der Kurs ist als Einstieg gedacht, und wir beschränken uns daher auf die Graphische Benutzer Oberfläche der Toolbox.

0 Einleitung

Die PDE Toolbox für MATLAB ist eine Umgebung zur Lösung von partiellen Differentialgleichungen auf ebenen Gebieten mit Hilfe der Methode der finiten Elemente. Sie gibt Hilfestellungen bei der Definition des Problems (Beschreibung des Gebietes, in dem die Gleichung zu lösen ist, der Randbedingungen und der Koeffizientenfunktionen der Differentialgleichung), erzeugt eine unstrukturierte Dreieckszerlegung des Gebietes, diskretisiert die Differentialgleichung mit linearen finiten Elementen zu dieser Zerlegung und löst das diskrete Problem. Schließlich stellt sie verschiedene Möglichkeiten zur Visualisierung der Lösung bereit. Die minimalen Voraussetzungen für die Benutzung der Toolbox werden in dem Handbuch [1] so beschrieben: "The minimal requirement is that you can formulate a PDE problem on paper (draw the domain, write the boundary conditions, and the PDE)".

Die grundlegende Gleichung, die in der Toolbox behandelt wird, ist die partielle Differentialgleichung

$$-\nabla \cdot (c\nabla u) + au = f \quad \text{in } \Omega, \tag{1}$$

wobei $\Omega \subset \mathbb{R}^2$ ein beschränktes Gebiet ist und c, a, f und die gesuchte Funktion u skalare, komplexwertige Funktionen auf Ω sind, mit Dirichletschen Randbedingungen

$$hu = r \quad \text{auf } \partial\Omega \tag{2}$$

*MATLAB ist ein eingetragenes Warenzeichen der Firma The Math Works, Inc.

†Technische Universität Hamburg–Harburg, Arbeitsbereich Mathematik, Kasernenstrasse 12, D–20173 Hamburg, voss@tu-harburg.d400.de

oder Randbedingungen dritter Art

$$n \cdot (c\nabla u) + qu = g \quad \text{auf } \partial\Omega. \tag{3}$$

Dabei ist n der äußere Normalenvektor auf $\partial\Omega$, und h, r, q und g sind komplexwertige Funktionen, die auf $\partial\Omega$ definiert sind. Die Gleichung (1) wird in der Toolbox elliptisch genannt, auch wenn die Funktion c auf Ω nicht von einem Vorzeichen ist, und die Randbedingung (3) wird abweichend von der mathematischen Literatur Neumannsche Randbedingung genannt.

Es können ferner "parabolische" Anfangsrandwertaufgaben

$$d\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) + au = f \tag{4}$$

und "hyperbolische" Anfangsrandwertaufgaben

$$d\frac{\partial^2 u}{\partial t^2} - \nabla \cdot (c\nabla u) + au = f \tag{5}$$

behandelt werden. In diesen Fällen können die Funktionen c, a, f, d, g, q, h und r auch von t abhängen.

Weiter kann die Eigenwertaufgabe

$$-\nabla \cdot (c\nabla u) + au = \lambda du \quad \text{in } \Omega \tag{6}$$

mit homogenen Randbedingungen ($r = 0$ in (2) bzw. $g = 0$ in (3)) numerisch gelöst werden, und es steht im "elliptischen" Fall ein Löser für die nichtlineare Randwertaufgabe

$$-\nabla \cdot (c(u, \nabla u)\nabla u) + a(u, \nabla u)u = f(u, \nabla u) \tag{7}$$

mit den Randbedingungen (2) bzw. (3) bereit, wobei nun auch die Funktionen h und r bzw. q und g von u abhängen dürfen. Schließlich können auch gewisse ebene Systeme zweiter Ordnung behandelt werden.

Der vorliegende Report war Grundlage des "Technologieseminars" über "Numerische Simulation – Partielle Differentialgleichungen" der Daimler Benz Aerospace, Airbus GmbH, in dem vom Autor nur die numerische Lösung stationärer Probleme behandelt wurde. Wir beschränken daher die folgenden Beispiele auf "elliptische" Randwertaufgaben. Besonderer Dank gilt Vera Lochmann, die alle Abbildungen dieses Reports erstellte.

1 Beispiel 1

Wir betrachten die Differentialgleichung

$$-\Delta u = -4 \quad \text{in } \Omega := \{(x, y) \in \mathbb{R}^2 \;:\; x^2 + y^2 < 1\} \tag{8}$$

mit den Dirichletschen Randbedingungen

$$u(x, y) = 1 \quad \text{für } x^2 + y^2 = 1. \tag{9}$$

Offensichtlich ist

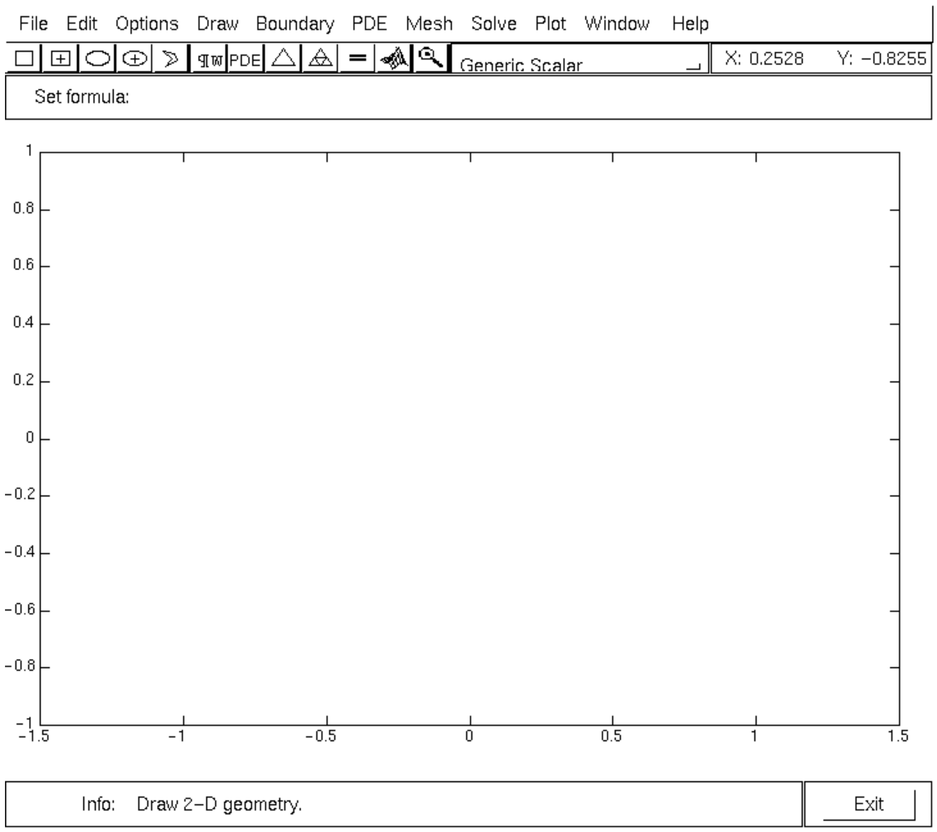
$$u(x, y) = x^2 + y^2 \tag{10}$$

die Lösung der Randwertaufgabe (8), (9), und wir können die Konvergenzeigenschaft der finiten Elementmethode mit stückweise linearen Ansatzfunktionen auf Dreieckszerlegungen von Ω verifizieren.

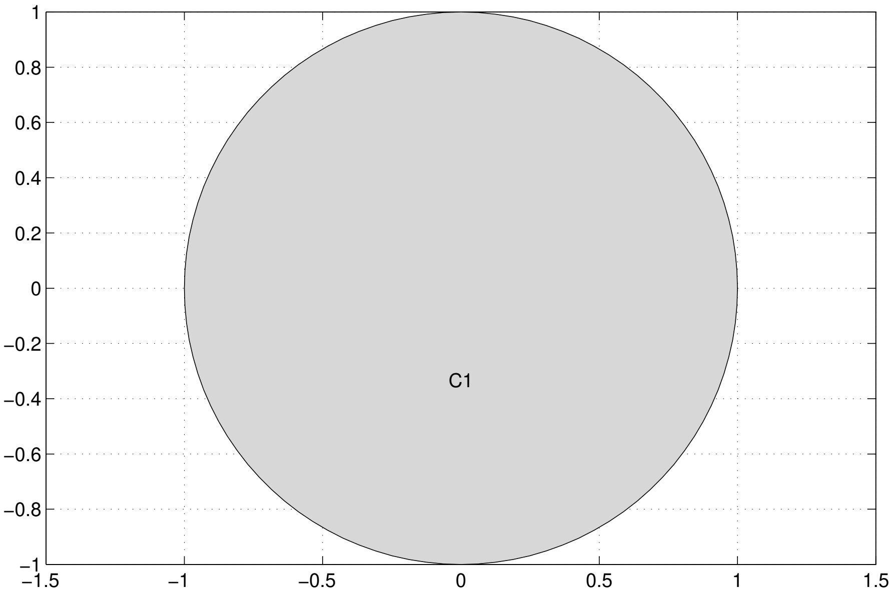
Um die Randwertaufgabe mit der PDE Toolbox zu lösen, verwenden wir die Graphische Benutzeroberfläche (Graphical User Interface: GUI). Unter MATLAB rufen wir diese mit dem Befehl

pdetool

auf und erhalten den folgenden Bildschirm:



Wir erzeugen zunächst das Gebiet Ω, in dem die Differentialgleichung zu lösen ist. Dazu wählen wir in dem **Options** Menü die Punkte *Grid* und *Snap*, die die Beschreibung von Ω häufig erleichtern. Wir klicken das Ikon mit der Ellipse mit dem markierten Mittelpunkt an. Danach klicken wir den Mittelpunkt $(0, 0)$ des Kreises Ω näherungsweise mit der linken Maustaste an (ohne die Option *Snap* hätten wir genau arbeiten müssen) und ziehen bei gedrückter linker Taste die Maus. Dabei werden blaue Ellipsen erzeugt, deren Halbachsen sich nicht kontinuierlich ändern, sondern so springen, dass die Abschnitte auf den Achsen mit Gitterpunkten zusammenfallen (auch dies ist eine Wirkung der Option *Snap*). Es macht keine Mühe, auf diese Weise den Einheitskreis Ω zu erzeugen. Lassen wir die Maustaste los, so erscheint in grauer Schraffur der Einheitskreis mit der Bezeichnung $C1$ (circle 1).



Ferner erscheint in dem Fenster **Set formula** das Symbol $C1$. Auf die Benutzung dieses Fensters bei der Definition komplizierterer Grundgebiete kommen wir später zurück.

Um die Randbedingungen einzugeben, klicken wir das Ikon mit dem Symbol $\partial\Omega$ an (Gleiche Wirkung erreichen wir, indem wir in dem **Boundary** Menü den Punkt *Boundary Mode* wählen). Von dem Gebiet bleibt die Randkurve, die hier in vier rote Viertelkreise unterteilt ist. Voreingestellt sind homogene Dirichletsche Randbedingungen $u = 0$ auf dem ganzen Rand. Um diese in die gewünschten inhomogenen Randbedingungen zu ändern, klicken wir ein Randteil mit der linken Maustaste an. Dieses wird daraufhin schwarz. Hierdurch wird angezeigt, dass die Randbedingung auf ihm (neu) spezifiziert werden kann. Da die Randbedingung auf allen vier Randstücken gleich lautet, $u = 1$, können wir die Änderung auf allen gleichzeitig vornehmen. Dazu klicken wir bei gedrückter Shift-Taste die anderen drei Randteile mit der linken Maustaste an. Um die Änderung durchzuführen wählen wir in dem **Boundary** Menü den Punkt *Specify Boundary Conditions*. In dem erscheinenden Fenster für die Randbedingungen ändern wir den Wert für r in 1 und bestätigen die Randbedingungen durch Anklicken von OK.

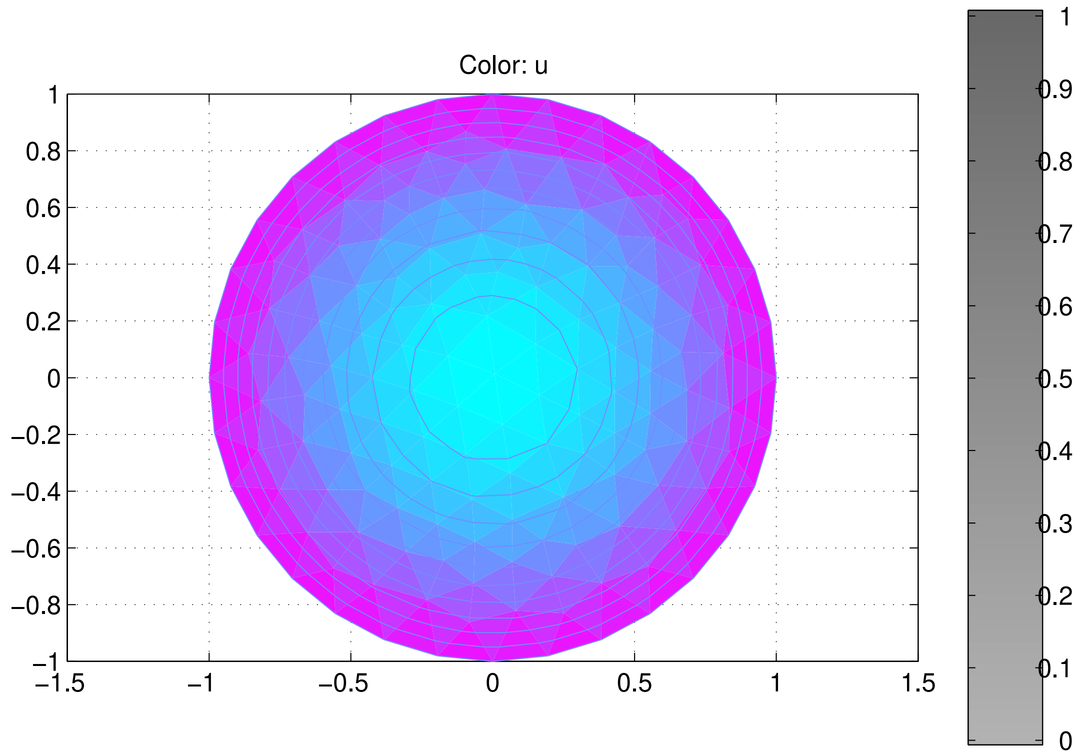
In dem Fenster sind nun wieder alle Randstücke rot gezeichnet. Dies bedeutet, dass überall Dirichletsche Bedingungen vorgeschrieben sind. Hätten wir auf Teilen verallgemeinerte Neumannsche Bedingungen gewählt, so wären diese Randteile nun blau gezeichnet.

Wir definieren nun die Differentialgleichung. Dazu wählen wir in dem **PDE** Menü den Punkt *PDE Mode*). Das Gebiet Ω erscheint wieder grau. Nach Doppelklicken in diesem Gebiet erscheint das Fenster zur Spezifikation des Typs der Differentialgleichung und der Koeffizienten (Dieses Fenster hätten wir auch durch Anklicken des Ikons **PDE** erreichen können). Die Voreinstellung ist “Elliptic”, die wir beibehalten. Die Funktion f der rechten

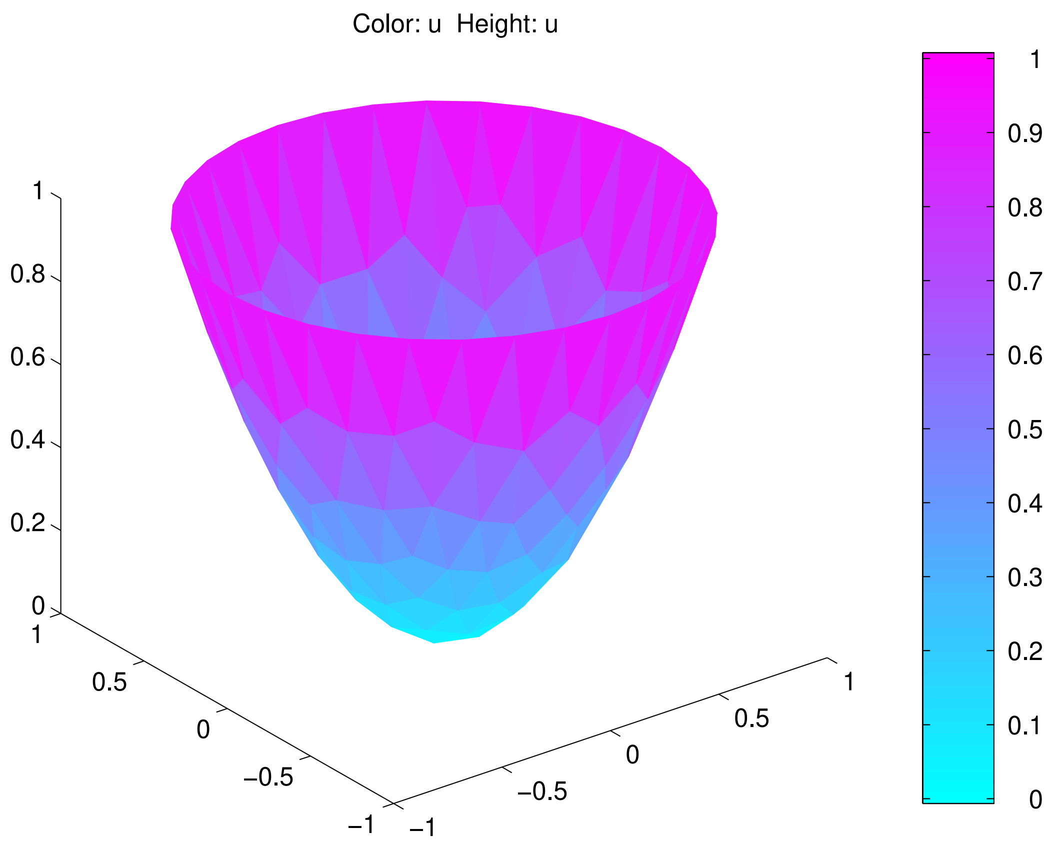
Seite ändern wir in -4 und bestätigen die Wahl durch Anklicken von OK.

Klicken wir nun das Ikon mit dem Symbol = an, so wird eine Triangulierung des Gebiets Ω erzeugt, das lineare Gleichungssystem, das zur finiten Elementmethode mit linearen Elementen zu dieser Triangulierung gehört, aufgestellt und gelöst, und es wird die Lösung $u(x, y)$ dargestellt.

Um den Eindruck von der Lösung zu verbessern, kann man in dem Menü **Plot** mit dem Punkt *Parameters ...* verschiedene Darstellungen wählen. Man kann zusätzlich eine vorgebbare Zahl von Niveaulinien in den Plot eintragen und erhält hier mit 10 Niveaulinien



Man kann die Farbdarstellungen ändern, 3–D Plots erzeugen, und in alle Darstellungen die Triangulierung eintragen. Als 3–D Plot mit Triangulierung erhält man.



Um die Approximationseigenschaften der finite Elementmethode zu verifizieren, exportieren wir die Lösung aus der "pdetool"–Umgebung in die MATLAB Sitzung. Hierzu wählen wir in dem **Solve** Menü den Punkt *Export Solution ...*, ändern in dem erscheinenden Fenster den Namen der Lösung in $u1$ und bestätigen mit OK. Hierdurch wird ein Vektor $u1$ erzeugt, der die errechneten Näherungswerte in den Knoten der verwendeten Triangulierung enthält. Um die Fehler berechnen zu können, benötigen wir die Koordinaten der Knoten. Wir exportieren diese, indem wir in dem Menü **Mesh** den Punkt *Export Mesh ...* wählen, in dem sich öffnenden Fenster die Namen in $p1$, $e1$ und $t1$ ändern und mit OK bestätigen.

Wir wiederholen die Berechnungen nun mit einem verfeinerten Gitter. Wir klicken das Ikon mit den vier Dreiecken an. Dadurch wird jedes Dreieck in vier kongruente Dreiecke zerlegt. Wir bestimmen die zugehörige Näherungslösung durch Anklicken des Ikons $=$ und exportieren die Lösung und die verwendete Triangulierung mit den Namen $u2$, $p2$, $e2$ und $t2$. Diese Verfeinerung wiederholen wir noch viermal und exportieren mit den Namen uj, pj, ej und tj, $j = 3, 4, 5, 6$.

Um die Approximationsgüte zu überprüfen, wechseln wir mit der Tastenkombination *Alt* und *Esc* von dem GUI Fenster in das MATLAB Fenster. Durch die MATLAB Befehle

```
x=p1(1,:);y=p1(2,:);exact=(x.^2+y.^2)';m1=max(abs(exact-u1))
```

erhalten wir maximalen Fehler in den Knoten

m1=5.24e-3,

und mit

x=p2(1,:);y=p2(2,:);exact=(x.^2+y.^2)';m2=max(abs(exact-u2)),
...,
x=p6(1,:);y=p6(2,:);exact=(x.^2+y.^2)';m6=max(abs(exact-u6))

ergibt sich

m2=1.61e-3, m3=4.73e-4, m4=1.35e-4, m5=3.92e-5, m6=1.12e-5.

Es wird also bei Halbierung der Schrittweite der Fehler (wenigstens in den Knoten, dies gilt aber in ganz Ω) fast geviertelt. Die Anzahl der Knoten wird bei jeder Verfeinerung ungefähr vervierfacht.

Tatsächlich kann man zeigen, dass für den Fehler die Abschätzung

$$\|u - u_h\|_\infty \leq Ch^2 |\ln h|$$

gilt, wobei C eine vom Gebiet Ω abhängige Konstante ist und h die maximale Kantenlänge eines Dreiecks in der Triangulierung ist. Die maximale Kantenlänge erhält man durch

h=0;s=size(e); for j=1:s(2) k=e(1,j);l=e(2,j); h=max(h,norm(p(:,k)-p(:,l)));end

Zusammenfassend ist in unserem Beispiel

Dimension	Fehler	h	Fehler/($h^2 \vert \log h \vert$)
144	5.24 e-3	1.96 e-1	8.37 e-2
541	1.61 e-3	9.81 e-2	7.20 e-2
2097	4.73 e-4	4.91 e-2	6.51 e-2
8257	1.35 e-4	2.45 e-2	6.05 e-2
32769	3.92 e-5	1.23 e-2	5.92 e-2
130561	1.12 e-5	6.14 e-3	5.84 e-2

2 Beispiel 2

Wir betrachten nun ein Beispiel, bei dem die Lösung nicht glatt ist. Es sei die Menge Ω' in Polarkoordinaten gegeben durch

$$\Omega' := \{(r, \varphi) \ : \ 0 < r < 1, \ 0 < \varphi < 1.5\pi\}.$$

Dann besitzt die Randwertaufgabe

$$\begin{array}{rcll} -\Delta u & = & 0 & \text{in } \Omega' \\ u & = & 0 & \text{für } 0 \le r \le 1 \text{ und } \varphi = 0 \text{ oder } \varphi = 1.5\pi \\ u & = & \sin\frac{2\varphi}{3} & \text{für } r = 1 \text{ und } 0 \le \varphi \le 1.5\pi \end{array}$$

die Lösung

$$u(r,\varphi) = r^{2/3}\sin\frac{2\varphi}{3}, \tag{11}$$

und es ist offensichtlich, dass für $r \to 0$ nicht einmal die ersten Ableitungen beschränkt sind. Wir wollen sehen, wie sich die Fehler der Näherungslösungen bei Halbierung der Schrittweiten verhalten.

Wegen der Implementierung der arctan–Funktion in MATLAB ist es einfacher, ein gedrehtes Problem zu betrachten. Es sei

$$\Omega := \{(r,\varphi)\ :\ 0 < r < 1,\ -0.75\pi < \varphi < 0.75\pi\}$$

und

$$u(r,\phi) = r^{2/3}\cos\frac{2\varphi}{3}$$

die Lösung der Potentialgleichung in Ω mit homogenen Dirichlet Bedingungen auf den geraden Randstücken und

$$u(1,\varphi) = \cos\frac{2\varphi}{3}, \quad -0.75\pi \le \varphi \le 0.75\pi.$$

Wir bestimmen Näherungslösungen mit der PDE Toolbox unter Verwendung des GUI. Nach dem Aufruf von

```
pdetool
```

wählen wir wieder die Optionen *Grid* und *Snap* in dem **Options** Menü und erzeugen wie eben den Einheitskreis $C1$.

Da in dem GUI nur achsenparallele Rechtecke vorgesehen sind, erzeugen wir als nächstes ein Quadrat mit der Seitenlänge 1, indem wir das Ikon mit dem Rechteck anklicken, in dem nicht das Zentrum markiert ist, dann mit der linken Maustaste den Punkt $(0,0)$ anklicken und zum Punkt $(1,1)$ ziehen. (Das Ikon mit dem Rechteck, bei dem das Zentrum markiert ist, wird verwendet, um ein Rechteck zu erzeugen, indem die Maus bei festgehaltener linker Taste vom Zentrum des gewünschten Rechtecks zu einem Eckpunkt zieht.)

Um den Bereich Ω zu erzeugen, drehen wir nun das eben erzeugte Quadrat SQ1 (square 1). Dazu wählen wir in dem Menü **Draw** den Punkt *Rotate...*. In das sich öffnende Fenster tragen wir als Rotationswinkel 135° ein, entfernen die Voreinstellung, dass um den Schwerpunkt gedreht wird, tragen als Drehpunkt den Vektor [0 0] ein und bestätigen mit OK. Nachdem wir die Eingabe mit OK bestätigt haben, klicken wir das **Set formula** Fenster an, ändern die Eintragung $C1 + SQ1$ in $C1 - SQ1$ und drücken die Enter–Taste. Es mag zunächst irritieren, dass sich das Fenster, in dem der Bereich Ω erzeugt wird, nicht ändert.

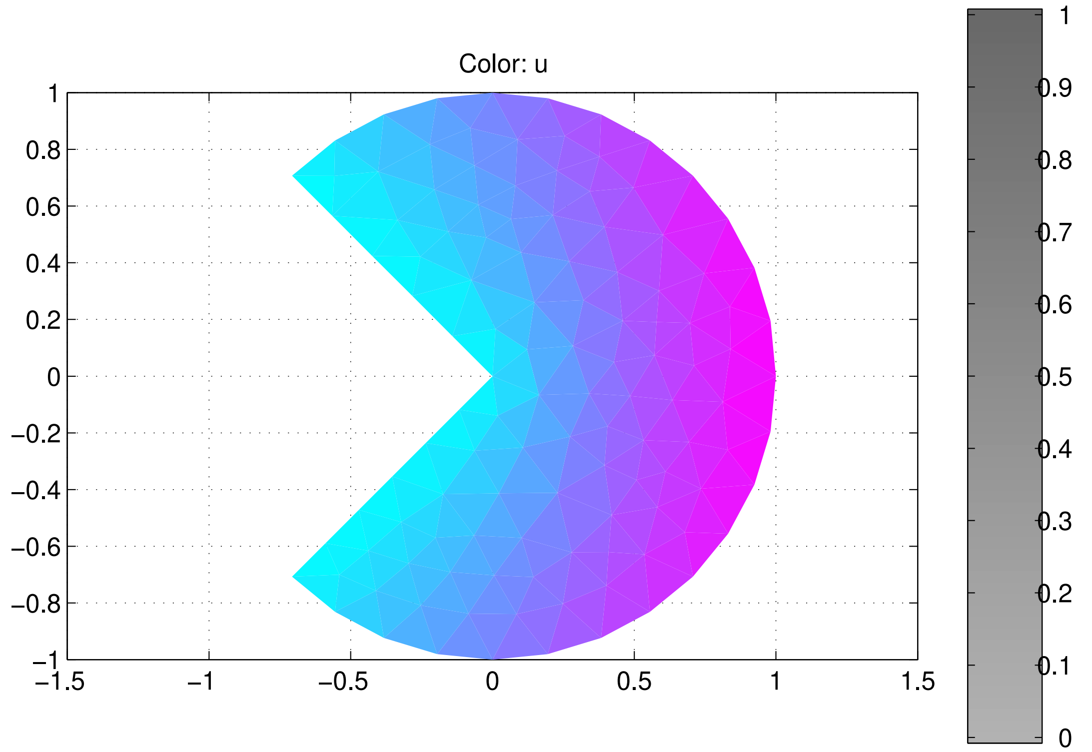
Wählt man nun den *Boundary Mode* in dem **Boundary** Menü, so wird der gewünschte Bereich Ω erzeugt.

Die Randbedingungen auf den geradlinigen Randstücken von Ω müssen nicht geändert werden, da dort homogene Dirichlet Bedingungen gefordert werden. Wir markieren wie in dem letzten Beispiel wieder die drei Kreisbögen und wählen in dem **Boundary** Menü die Einstellung *Specify Boundary Conditions*. In das sich öffnende Fenster tragen wir für die Funktion r

$$\cos(2/3 * atan2(y,x))$$

ein und bestätigen mit OK.

Die Differentialgleichung ändern wir, indem wir im **PDE** Menü den *PDE Mode* wählen, den Bereich Ω doppelt anklicken und in dem sich öffnenden Fenster die Funktion f in 0 ändern. Klicken wir danach das Ikon $=$ an, so wird wieder ein Anfangsgitter erzeugt, die



Da wir an dem Fehler interessiert sind, wählen wir in dem **Plot** Menü den Punkt *Parameters …* an, öffnen unter *Property* das erste Menü und tragen unter *user entry*

u-(x.ˆ2+y.ˆ2).ˆ(1/3).*cos(2/3*atan2(y,x)) ein. Drücken wir nun *Plot*, so erhalten wir einen Plot des Fehlers in dem Bereich Ω. Wir lesen daraus ab, dass der maximale Fehler nahe dem Punkt $(0,0)$ angenommen wird und den Wert $1.7e-2$ hat.

Wiederholt man die Lösung mit verfeinerter Schrittweite, so erhält man den Fehler $1.2e-2$, im nächsten Schritt $7.5e-3$, und dann $4.7e-3$. Den letzten Schritt sollte man nur dann ausführen, wenn man an einer schnellen Workstation arbeitet oder sowieso vorhatte, eine längere Kaffeepause einzulegen. Ferner sieht man, dass in allen Verfeinerungsstufen der maximale Fehler nur in der Nähe des Punktes $(0,0)$ angenommen wird und in dem übrigen Teil von Ω sehr klein ist.

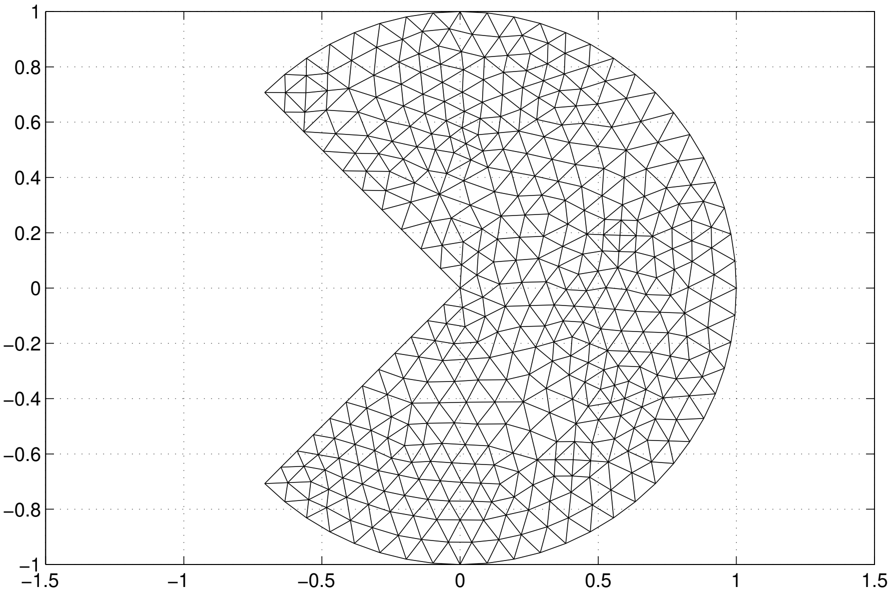
Dies legt die Vermutung nahe, dass man zu sehr viel besseren Näherungen gelangt, wenn man das Gitter nur in der Nähe dieses Punktes verfeinert. Die PDE Toolbox bietet die Option, das Gitter adaptiv zu verfeinern. Es wird dabei der Fehler einer berechneten Näherungslösung in jedem Dreieck geschätzt und nur dort eine Verfeinerung vorgenommen, wo der lokale Fehler oberhalb eines vorgebbaren Teils des maximalen geschätzten lokalen Fehlers liegt.

Bevor wir einstellen, dass das Gitter adaptiv erzeugt werden soll, klicken wir zunächst das Ikon Δ an und erzeugen so das Ausgangsgitter, da sonst das sehr feine Gitter des letzten Schritts adaptiv verfeinert würde.

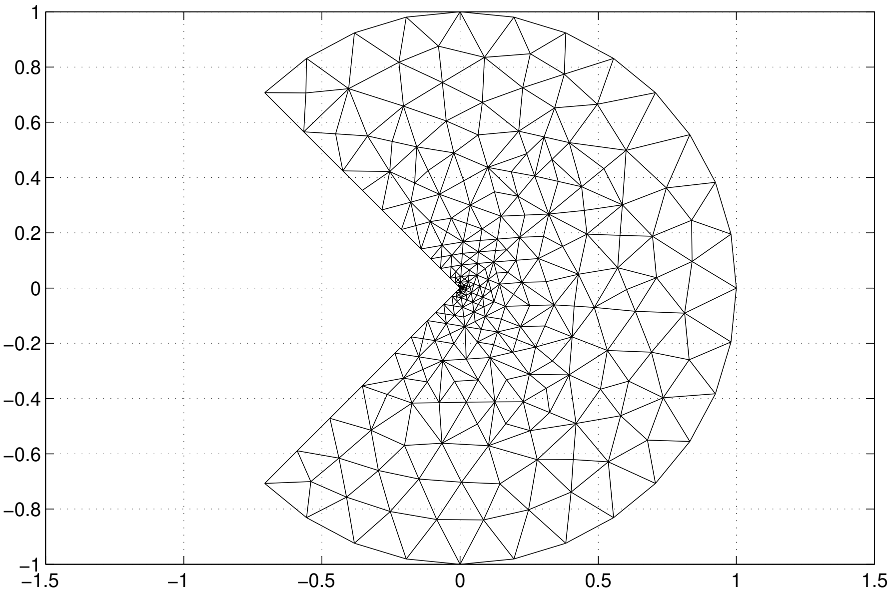
Wir wählen in dem Menü **Solve** den Punkt *Parameters...* an und hierin die Option *Adaptive mode*. Verändern wir die Voreinstellungen nicht, so erhalten wir nach Anklicken von $=$ eine Lösung mit dem maximalen Fehler $2.2e-3$.

Exportiert man die Lösungen wie im letzten Beispiel, so erhält man die tatsächlichen maximalen Fehler $1.72e-2$, $1.16e-2$, $7.46e-3$ und $4.74e-3$ bei den bestimmten Verfeinerungsstufen mit den Problemdimensionen 118, 434, 1663 und 6509 und für das adaptive Verfahren den maximalen Fehler $2.18e-3$ mit der Dimension 286 des diskretisierten Problems.

Die folgenden Abbildungen enthalten die Triangulierung, die man durch eine Verfeinerung ohne Adaption erhält,



und diejenige, die mit dem adaptiven Verfahren erzeugt wird.



3 Beispiel 3

Als drittes Beispiel betrachten wir eine ebene Potentialströmung in einem L–förmigen Kanal. Es sei

$$\Omega := ((-1,1) \times (0,1)) \cup ((0,1) \times (-1,0]).$$

Wir bestimmen das Geschwindigkeitspotential ϕ als Lösung der Laplace Gleichung

$$\Delta\phi = 0 \quad \text{in } \Omega. \tag{12}$$

Hinzu treten die Randbedingungen, dass längs der Wände des Kanals aus dem Kanal keine Flüssigkeit austritt, für die Normalkomponente der Geschwindigkeit also

$$\frac{\partial\phi}{\partial n} = 0 \tag{13}$$

gilt. Ferner ist auf dem Randstück $\{(x,-1) \; : \; 0 \le x \le 1\}$ die Eintrittsgeschwindigkeit

$$\frac{\partial}{\partial y}\phi(x,-1) = -x(1-x) \tag{14}$$

und auf dem Randstück $\{(-1,y) \; : \; 0 \le y \le 1\}$ die Austrittsgeschwindigkeit

$$\frac{\partial}{\partial x}\phi(-1,y) = y(1-y) \tag{15}$$

gegeben.

Unter

pdetool

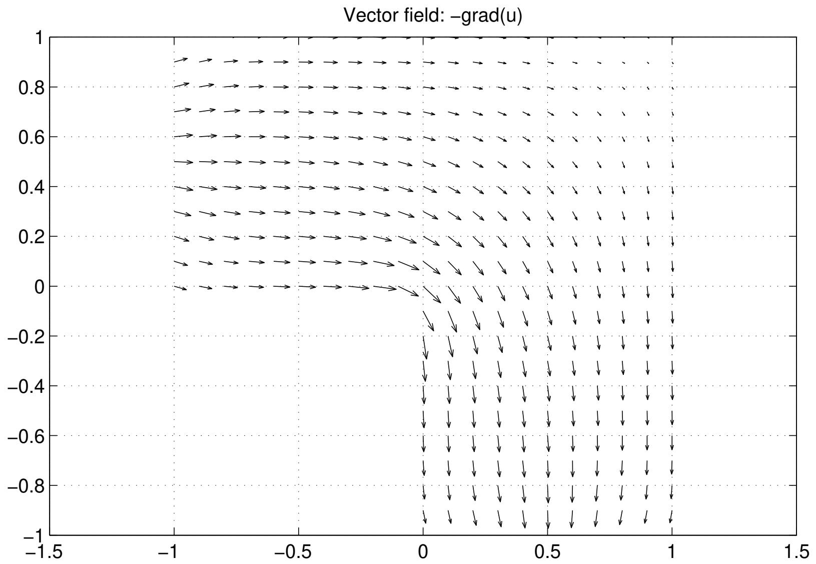
zeichnen wir zunächst das achsenparallele Rechteck $R1$, das durch die Eckpunkte $(-1,1)$ und $(1,0)$ bestimmt ist, und das achsenparallele Quadrat $SQ1$, das durch die Eckpunkte $(0,0)$ und $(1,-1)$ festgelegt ist.

In dem **Boundary** Menü entfernen wir den inneren Rand $\{(x,0) \; : \; 0 < x < 1\}$ durch Anklicken von *Remove All Subdomain Borders*. Dies geht manchmal erst, nachdem Sie *Boundary Mode* angeklickt haben. Unter *Specify Boundary Conditions* wählen wir Neumannsche Randbedingungen (mit dem Voreinstellungswert $g = 0$) und bestätigen mit OK. Um die Randbedingung am Eintrittsrand $\{(x,-1) \; : \; 0 < x < 1\}$ zu ändern, klicken wir dieses Randstück doppelt an und ändern in dem sich öffnenden Fenster g in $-x.*(1-x)$. Genauso ändern wir die Randbedingung am Austrittsrand $\{(-1,y) \; : \; 0 < y < 1\}$, indem wir das Randstück doppelt anklicken und in dem sich öffnenden Fenster dir Funktion g in $y.*(1-y)$ ändern.

Wir ändern die Differentialgleichung, indem wir das Ikon mit dem Symbol **PDE** anklicken und in dem sich öffnenden Fenster der rechten Seite f den Wert 0 zuordnen.

Klicken wir nun das Ikon = an, so wird eine Triangulierung erzeugt, das entstehende Gleichungssystem gelöst und das Geschwindigkeitspotential graphisch ausgegeben.

Interessanter ist in diesem Beispiel das Geschwindigkeitsfeld. Dieses kann man ausgeben, indem man in dem **Plot** Menü unter dem Punkt *Parameters ...* die Option "Arrows" anklickt. Man erhält dann die folgende Ausgabe:



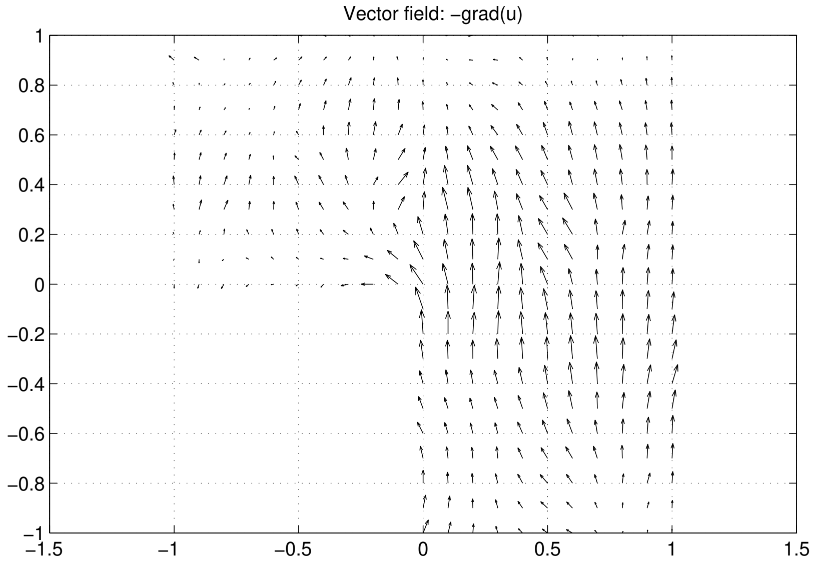
Erstaunlich ist, dass das Programm keinen Hinweis darauf gibt, dass die Steifigkeitsmatrix des Problems singulär ist. Unser Ausgangsproblem ist nicht eindeutig lösbar, denn ist ϕ ein Geschwindigkeitspotential, so auch $\phi + c$ für jede Konstante c, und diese Eigenschaft vererbt sich offenbar auf das diskretisierte Problem.

Immerhin besitzt das Problem überhaupt eine Lösung, und es könnte sein, dass die Toolbox beim Potentialproblem mit Neumannschen Randbedingungen eine Normierungsbedingung für das Potential enthält und so die Regularität der Steifigkeitsmatrix erzwungen wird.

Um dies zu prüfen, ändern wir die Randbedingungen so ab, dass das entstehende Problem nicht lösbar ist. Wir wählen auf dem Randstück $\{(x,-1)\ :\ 0 < x < 1\}$ die Randbedingung

$$\frac{\partial \phi}{\partial n}(x,-1) = 10, \quad 0 \leq x \leq 1,$$

und homogene Neumannsche Bedingungen auf dem Rest des Randes. Hierfür wird durch die Toolbox die Lösung der folgenden Skizze erzeugt, und es wird keine Warnung ausgegeben, dass die Steifigkeitsmatrix singulär ist oder sehr große Kondition besitzt.



4 Beispiel 4

Wir betrachten ein einfaches Modell eines Schraubenschlüssels. Ein ähnliches Beispiel findet man in [2].

Wir wechseln von der **Generic Scalar** Anwendung in die **Structural Mechanics; Plane Stress** Anwendung. Hierzu klicken wir (rechts in der Kopfleiste) **Generic Scalar** an und wählen in dem sich öffnenden Menü **Structural Mech., Plane Stress**.

Um die Beschreibung des Schraubenschlüssels zu erleichtern, wählen wir in dem **Options** Menü den Punkt *Grid Spacing*, schalten für die x–Achse die Voreinstellung "Auto" aus und ersetzen "-1.5:0.5:1.5" durch "-1.4:0.2:1.4". Danach bestätigen wir die Wahl durch *Apply* und *Done*. Ferner wählen wir im **Options** Menü wieder die Punkte *Grid* und *Snap*.

Zur Beschreibung des Gebiets Ω zeichnen wir das Rechteck R1 mit den Eckpunkten $(-0.8, -0.2)$ und $(0.2, 0.2)$, das Rechteck R2 mit den Eckpunkten $(0.2, -0.2)$ und $(0.8, 0.2)$, die Kreise C1 mit dem Mittelpunkt $(0.8, 0)$ und dem Radius 0.2 und C2 mit dem Mittelpunkt $(-1, 0)$ und dem Radius 0.4, sowie das Quadrat SQ1 mit den Eckpunkten $(-1.4, -0.2)$ und $(-1, 0.2)$.

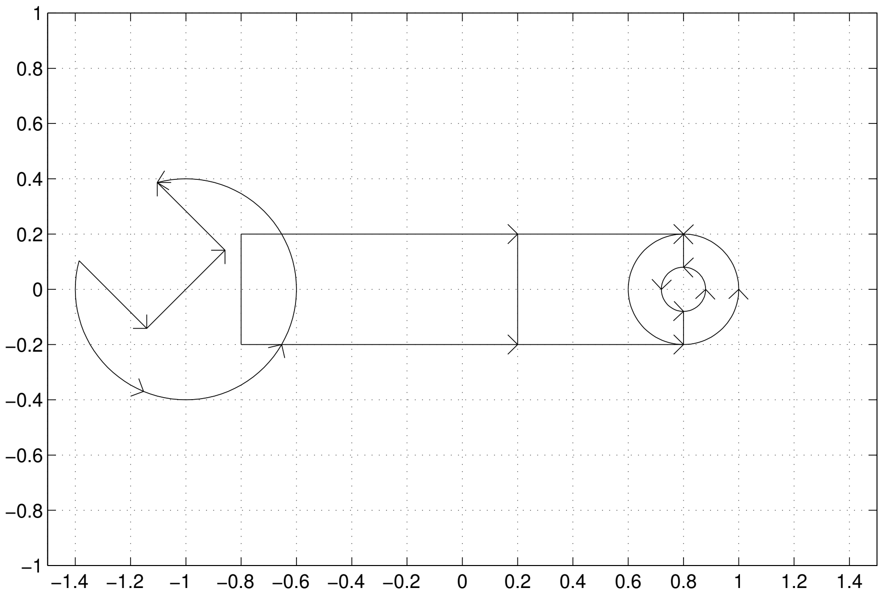
Das Quadrat SQ1 rotieren wir (ähnlich wie in Beispiel 2) um den Winkel $-45°$ und den Mittelpunkt $(-1, 0)$.

Wir schalten die Option *Snap* aus und zeichnen den Kreis C3 (oder die Ellipse E1 bei nicht so genauem Zeichnen) mit dem Mittelpunkt $(0.8, 0)$ und dem Radius 0.08. Diesen Kreis können wir korrigieren, indem wir ihn doppelt anklicken. In das sich öffnende Fenster können wir die Koordinaten des Mittelpunktes, die Halbachsen und die Bezeichnung $C3$ eintragen.

Nach diesen Vorbereitungen editieren wir die Mengenformel, setzen

(R1+R2+C1+C2)-(SQ1+C3)

und drücken die *Enter* Taste. Nachdem wir in das **Boundary** Menü umgeschaltet haben, erhalten wir das folgende Ergebnis.

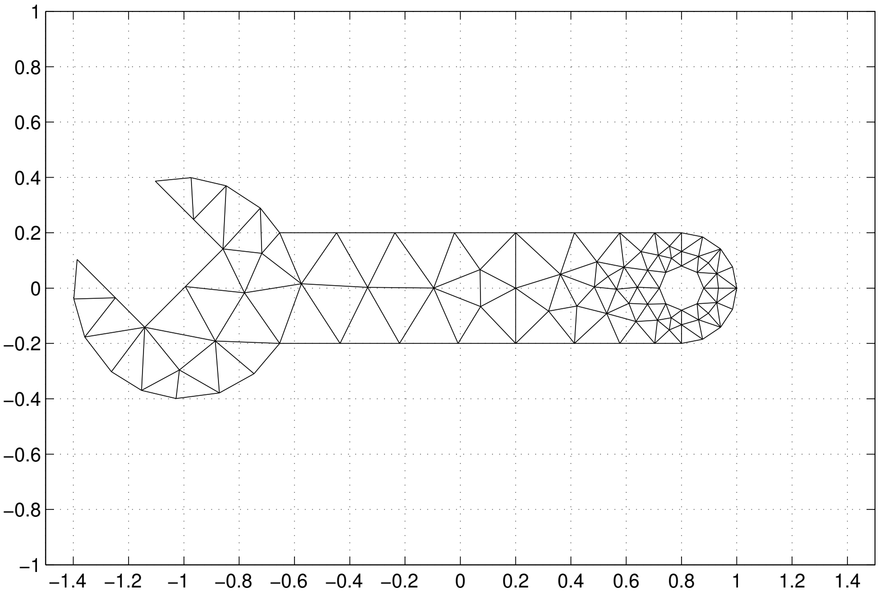


Wir entfernen die inneren Ränder außer demjenigen, der die Rechtecke R1 und R2 trennt (die Unterteilung von Ω, die hierdurch definiert wird, werden wir noch benötigen, da nur auf den rechten Teil des Schlüssels eine Kraft wirken soll). Hierzu klicken wir die inneren Ränder nacheinander an, wobei wir jeweils die Shift–Taste gedrückt halten. Danach wählen wir in dem **Boundary** Menü den Punkt *Remove Subdomain Border*.

Im Punkt *Specify Boundary Conditions ...* des **Boundary** Menüs wählen wir die Neumannschen Randbedingungen und belassen alle Koeffizienten bei ihren Voreinstellungen 0. Ferner klicken wir bei gedrückter Shift–Taste die drei geradlinigen Teile des Randes im Kopf des Schraubenschlüssels an und wählen in dem Punkt *Specify Boundary Conditions ...* des **Boundary** Menüs für diese Randteile homogene Dirichlet Bedingungen, da diese Teile des Randes festgehalten werden.

Wir lassen den Youngschen Modul und die Poisson Zahl bei ihren Voreinstellungen und lassen nur im rechten Teil des Schraubenschlüssels eine Kraft wirken. Dazu wählen wir den PDE Modus und klicken den rechten Teil des Schraubenschlüssels doppelt an. In dem sich öffnenden Fenster verändern wir den Wert von Ky (y–Richtung der Volumenkraft) auf -5 und bestätigen die Einstellungen mit OK.

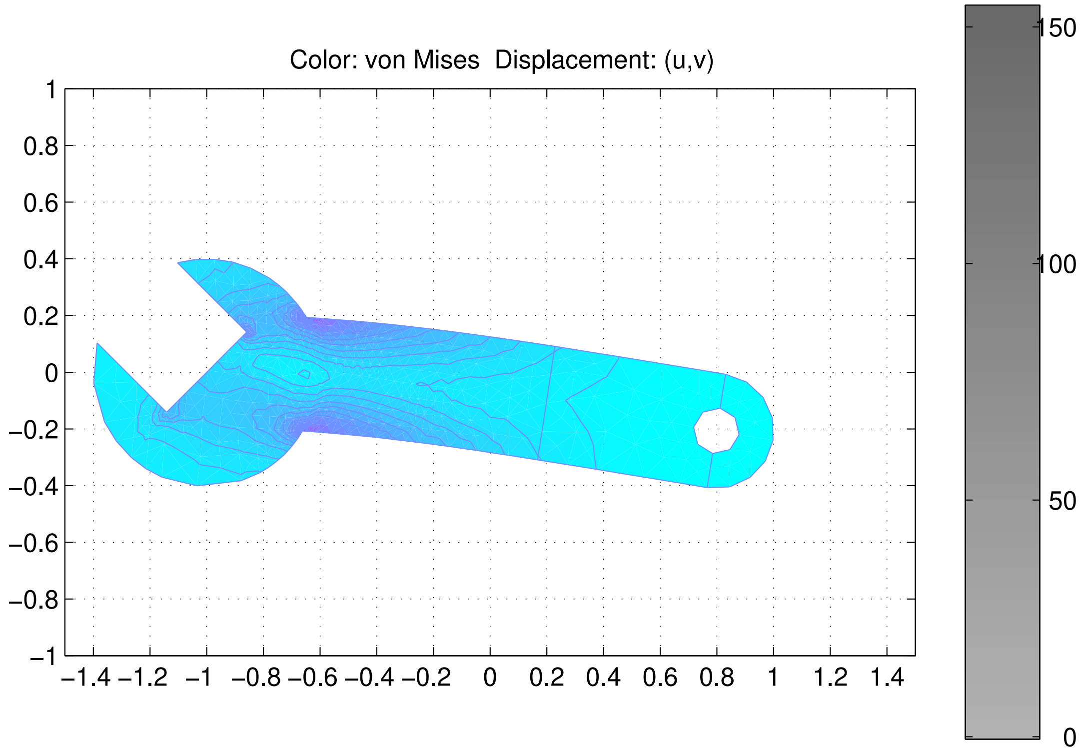
Wir erzeugen durch Anklicken von Δ die folgende Triangulierung.



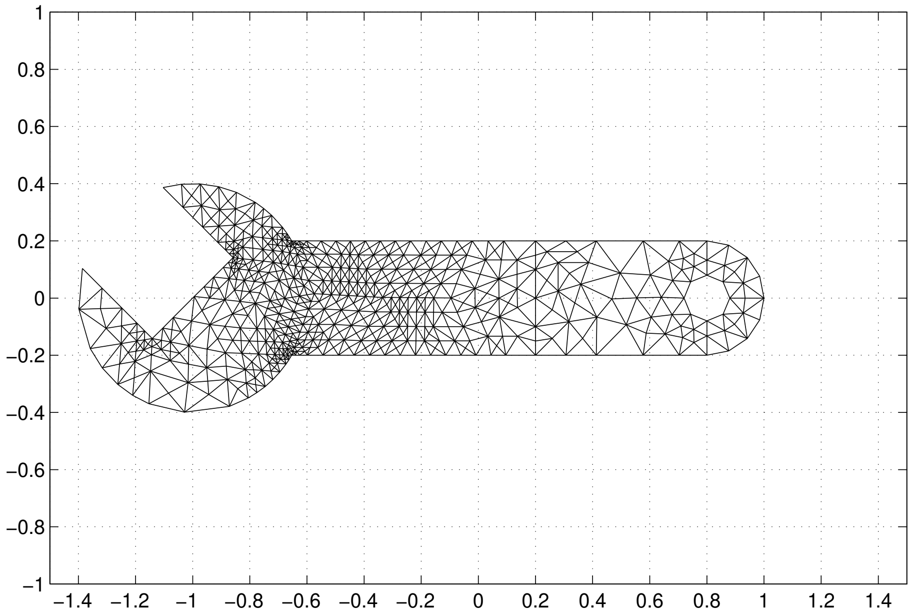
Da wir bei den einspringenden Ecken am Kopf des Schraubenschlüssels besonders große Spannungen erwarten, hier aber die Triangulierung besonders grob ist, wählen wir in dem Punkt *Parameters ...* des **Solve** Menüs die Option "Adaptive mode". Durch Anklicken von $=$ wird der Lösungsprozess begonnen, und man erhält als graphische Darstellung der Lösung eine Farbdarstellung der Verschiebungen in x–Richtung.

In diesem Fall kann man die Verschiebungen in beide Richtungen besser visualisieren durch einen Plot des deformierten Schraubenschlüssels. In der Einstellung *Parameters ...* des **Plot** Menüs wählen wir "Deformed Mesh", sowie "Contour" und "Color", und als Eigenschaft im Color Menü "von Mises stresses". Hierdurch werden die Niveaulinien der von Mises Spannungen $\sqrt{\sigma_1^2 + \sigma_2^2 - \sigma_1\sigma_2}$ ausgegeben.

Man erhält



Diese Lösung wurde mit der folgenden Triangulierung berechnet:



5 Beispiel 5

Als abschließendes Beispiel betrachten wir ein Minimalflächenproblem

$$\nabla \cdot \Big(\frac{1}{\sqrt{1 + u_x^2 + u_y^2}} u \Big) = 0 \quad \text{in } \Omega, \tag{16}$$

eine quasilineare Differentialgleichung, mit Dirichletschen Randbedingungen

$$u = g \quad \text{auf } \partial\Omega. \tag{17}$$

Wir zeichnen die Ellipse E1 mit dem Mittelpunkt $(0, 0)$ und den Halbachsen 1.5 und 0.8 und die konzentrische Ellipse E2 mit den Halbachsen 0.5 und 0.2, und definieren das Grundgebiet Ω als Mengendifferenz E1-E2.

Auf der inneren Ellipse wählen wir homogene Dirichlet Bedingungen und auf der äußeren Ellipse die inhomogene Dirichlet Bedingung

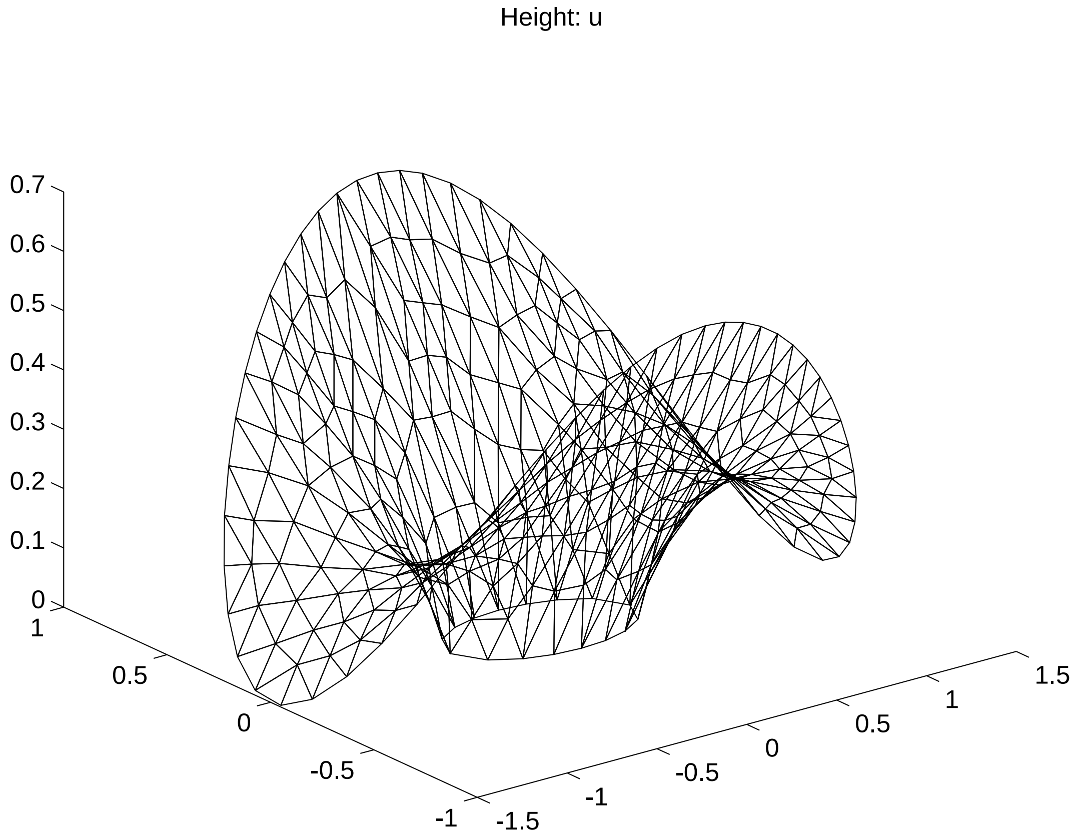
$$u(x, y) = y^2, \quad \frac{x^2}{1.5^2} + \frac{y^2}{0.8^2} = 1.$$

Nach Anklicken des Ikons mit dem Symbol PDE und Doppelklick in dem Gebiet Ω ändern wir in dem sich öffnenden Fenster die Funktion c in

1./sqrt(1+ux.^2+uy.^2)

und die rechte Seite f in 0.

Wir klicken das Ikon mit den vier Dreiecken an (da in der Ausgangstriangulierung die innere Ellipse zu grob dargestellt würde) und wählen in dem Punkt *Parameters ...* des Menüs **Solve** die Option “Use nonlinear solver”. Klickt man danach das Ikon mit dem Symbol = an, so erhält man die Lösung. Dabei wurde in dem Punkt *Parameters ...* des Menüs **Plot** die Option “Height (3-D plot) gewählt.



Literatur

[1] Partial Differential Equation TOOLBOX. For Use with MATLAB. User's Guide. The Math Works, Inc., Natwick 1995

[2] M. Dorobantu and M. Ringh: The PDE Toolbox, Minicourse. Computer Solutions Europe AB, Stockholm 1995