SAGE-Tutorium 09 im SoSe 2009

Lars Fischer*

24.06.2009

Inhaltsverzeichnis

1 Wiederholung

- Interaktive Worksheets und die verschiedenen Kontrollelemente sowie deren Dokumentation (`interact?`)
- Matrizen, Erzeuger von $\mathrm{GL}_2(\mathbb{Z})$ und $\mathrm{SL}_2(\mathbb{Z})$
- Diophantische Gleichungen, die explizite Bestimmung der Lösungsmenge.

*WWW: `http://w3.countnumber.de/fischer`, EMail: vorname.nachname (bei der) uni-siegen.de

2 Projekt: Kegelschnitt nach Diophant

Es folgt eine beispielhafte Bearbeitung des verbliebenen Projektes, das keinen Bearbeiter gefunden hat.

In der Vorlesung habt Ihr gesehen, wie sich die rationalen Pythagoräischen Zahlentripel parametrisieren lassen. Dazu habt Ihr durch einen Punkt $P_0 = (x_0, y_0)$ mit $x_0, y_0 \in \mathbb{Q}$ eine Gerade mit der Steigung $m \in \mathbb{Q}$ gelegt. Die Gerade schneidet den Kreis in einem weiteren Punkt (falls $m \neq 0$). Diesen bestimmt man durch Einsetzen der Geradengleichung und anschließendem Auflösen nach x. Das führt auf die Parametrisierung, die Ihr in der Vorlesung kennengelernt habt.

Für allgemeinere Kegelschnitte (Ellipsen, Parabeln, Hyperbeln, siehe auch `http://de.wikipedia.org/wiki/Kegelschnitt`, dort gibt es Bilder zu den verschiedenen Fällen) kann man analog vorgehen.

Wir wollen eine Funktion entwickeln, die zu einem Kegelschnitt und einem Punkt die Parametrisierung ermittelt. Dazu formulieren wir zuerst das Problem und entwickeln dann eine elegantere Darstellung mittels homogener Koordinaten.

Wir suchen eine Lösung $x, y \in \mathbb{Q}$ der Gleichung

$$Q(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0, \qquad a, b, c, d, e, f \in \mathbb{Q}. \tag{1}$$

2.1 Homogenisierung

Durch Homogenisierung können wir zu einer Lösung $x, y, z \in \mathbb{Z}$ der Gleichung

$$Q_h(x, y, z) = ax^2 + bxy + cy^2 + dxz + eyz + fz^2 = 0 \tag{2}$$

übergehen. Desweiteren sei $x'_0 = x_0/z$ und $y'_0 = y_0/z$ eine rationale Lösung von Q mit dem Hauptnenner z . Wir multiplizieren die Gleichung (1) mit $(z)^2$. Dann erhalten wir die Gleichung

$$(z)^2 Q(x'_0, y'_0) = ax_0^2 + bx_0y_0 + cy_0^2 + zdx_0 + zey_0 + z^2 f = 0,$$

was unsere homogene Gleichung (2) ist.

Der Zusammenhang zwischen Lösungen der beiden Gleichungen (einmal rational, einmal ganz) ist $Q(x/z, y/z) = Q_h(x, y, z)$, insbesondere ist $Q(x, y) = Q_h(x, y, 1)$.

Die Gleichung (2) lässt sich als Matrix schreiben:

```
reset()
# Ein wenig Kreativität und sage erlaubt es uns a,b,c,d,e,f als Symbole zu verwenden

P_abc=PolynomialRing(QQ,6,"abcdef")
#print P
a,b,c,d,e,f=P_abc.gens()

P_xyz=PolynomialRing( P_abc, 3, "xyz")
x,y,z= P_xyz.gens()



Q = a*x^2+b*x*y+c*y^2+d*x+e*y+f
Qh = a*x^2+b*x*y+c*y^2+d*x*z+e*y*z+f*z^2



M= matrix([[a,b,d],[b,c,e],[d,e,f]])
c=vector([x,y,z])


tmp=(c*M*c.transpose())
tmp
```

An dem letzten Ausdruck erkennen wir unsere Ausgangsgleichung wieder, bis auf den Faktor 2 vor b, d und e. Da aber die Koeffizienten alle rational sind, können wir $b' := b/2$ usw. setzen.

Es hat sich allerdings eingebürgert die Gleichung Q so zu schreiben, dass der Faktor 2 hier, wie in dem SAGE-Ausdruck *tmp*, auftaucht.

Gehen wir von nun an von:

$$\begin{aligned} K_h(x,y,z) &= (x,y,z)M(x,y,z)^t \\ &= ax^2 + 2bxy + cy^2 + 2dxz + 2eyz + fz^2 = 0 \end{aligned}$$

aus.

2.2 Vereinfachung der Gleichung

Diese Gleichung ist immer noch sehr unübersichtlich. Durch quadratische Ergänzung gelangen wir zu einer Form, die nur noch x^2, y^2 und z^2. Die quadratische Ergänzung ist ein Basiswechsel.

```
def qfgaussred( G ):
    """Berechnet die Gauss-Reduktion der symmetrischen Matrix G.  (Auch als Cholesky-

       Das ist die Funktion qfgaussred von Pari/GP.
       Das ist Algorithmus 2.7.6 aus Cohen: "A Course in Computational Algebraic Numb

       Es sei $Q(x) = x^t G x$ eine quadratische Form mit der Gram-Matrix G (hier ist

       Dann werden Koeffizienten $q_{i,j}$ berechnet, so dass
       $$Q(x) = \sum_{i=1}^n q_{ii} (x_i \sum_{j=i+1}^n q_{i,j} x_j ) ^2$$
       gilt.

       Es sei B,S := qfgaussred( G ). B ist dann eine Diagonalmatrix und S hat auf de

       Es ist dann
       $Q(x) = x^t G x = x^tS^t B S x $.
       Vollzieht man aber den Basiswechsel $v:= S x$, dann ist $x^t G x = v^t B v = \
       (Keine gemischten Ausdrücke mehr).

       In nur einer Variable nennt man das "quadratische Ergänzung"!

       INPUT:
           G - eine symmetrische Matrix, die Gram-Matrix der Quadratischen Form
       OUTPUT:
           B,S - das Tupel der quadratischen Matrizen B und S, wie oben beschrieben.

       EXAMPLES:
           sage: x,y = var("x,y");
           sage: G=matrix([[1,2],[2,4]]);
           sage: V=vector([x,y]).transpose();
           sage: V.transpose()*G*V
           [x^2 + 4*x*y + 4*y^2]
           sage: B,S=qfgaussred(G);
           sage: V=S*vector([x,y]).transpose();
           sage: V.transpose()*B*V
           [x^2 + 4*x*y + 4*y^2]
           sage: print B
```

```
            [1 0]

            [0 0]
    """

    if not G.is_symmetric():
        # wir brauchen eine Gram-Matrix
        raise ValueError("Die Matrix ist nicht symmetrisch!")

    n= G.ncols()

    # wegen der Division, die wir unten brauchen, müssen
    # wir die Matrix Q über den Körper der "Brüche von Elementen von G" anlegen
    # sonst scheitern wir unten bei den Divisionen
    MS= MatrixSpace( G[0,0].parent().fraction_field() , n,n )
    Q= MS( G )

    for i in range( n ):
        if G[i,i] == 0:
            for j in range(i+1,n):
                Q[i,i] = -Q[i,j]/2
        else:
            for j in range( i+1, n):
                Q[j,i] = Q[i,j]
                Q[i,j] = Q[i,j] / Q[i,i]
            for k in range(i+1,n):
                for l in range(k,n):
                    Q[k,l] = Q[k,l] - Q[k,i] * Q[i,l]

    B=MS(diagonal_matrix([Q[i,i] for i in range(n)] ))
    S=MS(identity_matrix(n))
    for i in range(n):
        for j in range(i+1,n):
            if G[i,i] == 0:
                S[j,i] = -Q[i,j]
            else:
                S[i,j] = Q[i,j]

    return B,S


B,S = qfgaussred(M)
print B , "\n"
```

```
print S
```

$$\begin{pmatrix} a & 0 & 0 \\ 0 & \frac{-b^2+ac}{a} & 0 \\ 0 & 0 & \frac{-cd^2+2bde-ae^2-b^2f+acf}{-b^2+ac} \end{pmatrix}$$

$$\begin{pmatrix} 1 & \frac{b}{a} & \frac{d}{a} \\ 0 & 1 & \frac{-bd+ae}{-b^2+ac} \\ 0 & 0 & 1 \end{pmatrix}$$

Das ist eine quadratische Form der Bauart $a'x^2 + b'y^2 + c'z^2 = 0$. Mit dem Satz von Legendre werdet Ihr in der Vorlesung ein Verfahren kennenlernen (und ein Projekt wird mit diesem Verfahren einen effizienten Lösungsweg implementieren) welches eine Lösung (sofern sie existiert) liefert.

Aus dieser Lösung lässt sich mit S^{-1} eine Lösung der ursprünglichen Gleichung gewinnen.

2.3 Parametrisierung

Nehmen wir also für die weitere Diskussion an, dass wir neben dem Kegelschnitt $K(x, y, z)$ auch eine ganzzahlige Lösung x'_0, y'_0, z'_0 mit $K(x'_0, y'_0, z'_0) = 0$ haben.

Nun gehen wir wieder zu der Gleichung $K_{\text{inhomogen}}(x, y) := ax^2 + by^2 + c = 0$ über. (Wir nehmen $x, y \in \mathbb{Q}$ an). Es sei $x_0 := x'_0/z'_0, y_0 := y'_0/z'_0$. Wir teilen durch a und bringen $\frac{c}{c}$ auf die andere Seite. Durch die Benennung $D := -\frac{b}{a}, N := \frac{-c}{a}$ haben wir die Gleichung auf die Form

$$x^2 - Dy^2 = N \tag{3}$$

gebracht. Da (x_0, y_0) eine Lösung ist, finden wir

$$N = x_0^2 - Dy_0^2 \tag{4}$$

Wir legen nun die Gerade l mit der Steigung $m \in \mathbb{Q}$ durch (x_0, y_0). Sie hat die Punkt-Steigungsform

$$\begin{aligned} m &= \frac{y - y_0}{x - x_0} \Rightarrow \\ x &= \frac{y - y_0}{m} + x_0 \end{aligned} \tag{5}$$

Einsetzen von (5) in Gleichung (3) liefert nach einigen Umformungen die Gleichung:

$$(y - y_0)\left(\frac{y - y_0}{m^2} + \frac{2x_0}{m} - D(y + y_0)\right) = 0 \tag{6}$$

Eine Nullstelle ist $y = y_0$. Nun suchen wir eine Nullstelle der rechten Klammer, dazu sortieren wir y nach links:

$$\begin{aligned}
y(\frac{1}{m^2} - D) - \frac{y_0}{m^2} + \frac{2x_0}{m} - Dy_0 = 0 \Leftrightarrow \\
y(\frac{1}{m^2} - D) = \frac{y_0}{m^2} - \frac{2x_0}{m} + Dy_0 | \text{ Multiplikation mit } m^2 \\
y(1 - Dm^2) = y_0 - 2x_0m + Dy_0m^2 \Leftrightarrow \\
y = \frac{y_0 - 2x_0m + Dy_0m^2}{1 - Dm^2}
\end{aligned}$$

Dieses y in Gleichung (5) eingesetzt, liefert den zweiten Schnittpunkt[1] der Geraden mit den Kegelschnitt.

Nun der Sage-Code, der den allgemeinen Kegelschnitt und eine Lösung entgegennimmt, den Kegelschnitt auf die vereinfachte homogene Form bringt, dann die Parametrisierung des vereinfachten inhomogenen Kegelschnitts bestimmt und dann alles auf die Form des ursprünglichen Kegelschnittes zurückrechnet:

```
def giveParametrization( a, b, c, d, e, f, x_0, y_0, testeLsg=False):
    """Berechnet eine Parametrisierung der impliziten Gleichung
       Q(x,y) = a x^2+b 2 xy + c y^2 + d 2 x + 2 ey+ f = 0
    """
    print "Parametrisierung eines Kegelschnittes:"
    print "  Betrachte die Gleichung: %sx^2 + %s2xy + %sy^2 + %s2x + %s2y + %s = 0"%(

    P.<m> = QQ[]

    Delta = 4*b^2-4*a*c

    if Delta == 0: # Wir folgen der Darstellung in http://www.jpr2718.org/ax2p.pdf
        # erst einmal die Zweien loswerden, da obiges Paper keine Zweien benutzt
        b= 2*b; d= 2*d; e= 2*e
        Q(x,y) = a*x^2+b*x*y + c *y^2 + d*x + e*y+ f

        # liegt (x_0,y_0) auf dem Kegelschnitt?
        tmp=Q(x_0,y_0)
```

[1] Es sei denn, m entspricht genau der Steigung der Tangente in (x_0, y_0).

```
#print tmp, type(tmp)
if tmp != 0:
    raise ValueError("Der Punkt x_0,y_0 liegt nicht auf dem Kegelschnitt! Q(x

if a== 0:
    # Vertausche x, und y. Es ist Delta == 0. Wären a und c  beider 0, so
    # folgt aus Delta== 0 auch b =0 und wir haben ein lineare Gleichung.
    a,c=c,a
    d,e=e,d

    S= matrix( [[0,1],[1,0]] )
    vx= y; vy= x
else:
    S= matrix( [[1,0],[0,1]] )
    vx=x;  vy=y

# nun ist a != 0, wir multiplizieren Q(x,y)= 0 mit 4a
# und können setzen:
X= 2*a*vx+b*vy+d
A= 2*b*d-4*a*e
K= d^2-4*a*f
# Nun geht es weiter mit der Gleichung
#   X^2-A*Y -K = 0
# in die wir die ursprüngliche überführt haben.
if A== 0:
    #   eine Lsg ist mgl., falls K ein Quadrat K=m^2 ist, dann ist
    #   X= +/- m und Y beliebig eine Lsg.
    if type(K.sqrt() ) ==  sage.rings.rational.Rational:
        r= S* (vector([ K.sqrt(), m ] )).transpose()
    else:
        raise ValueError(" Es ist K=%s kein Quadrat, es gibt keine Lösung"%st
else:
    # X = m, Y= (X^2-K)/A,    ist eine Lsg
    # x= (X-by-d)/(2*a), y=Y  ist eine Lsg der ursprünglichen Gleichung
    tmp_y= (m^2-K)/A
    tmp_x= (m-b*tmp_y-d)/(2*a)
    r= S* (vector([ tmp_x, tmp_y ] )).transpose()
# Die Multiplikation S*r macht die etwaige Vertauschung von x uny y rückgängi

# Einsetzen und prüfen:
sollteNullSein= Q(r[0,0], r[1,0])

if sollteNullSein !=0:
    print tmp_x, tmp_y
```

```
        print sollteNullSein
        raise ValueError("Hilfe! (Delta=0)")
    print "inhomogen:\t[", r[0],", ", r[1], ", 1]"
    return r


# Das ist der Fall, der oberhalb diskutiert wurde:
G= matrix([[a,b,d],[b,c,e],[d,e,f]])

# liegt (x_0,y_0) auf dem Kegelschnitt?
tmp=(vector([x_0,y_0,1]) * G * vector([x_0,y_0,1]).transpose())
#print tmp, type(tmp)
if tmp != 0:
    raise ValueError("Der Punkt x_0,y_0 liegt nicht auf dem Kegelschnitt! Q(x_0,y

print G
B,S= qfgaussred(G)
alpha,beta,gamma=B[0,0],B[1,1],B[2,2]
#print "  Vereinfachte homogene Gleichung: %s x^2+ %s y^2 + %s z^2 =0"%(str(alpha

z_0=QQ(x_0).denom()*QQ(y_0).denom()

tmp = S*vector([x_0 * z_0, y_0*z_0, z_0]).transpose()

X_0= tmp[0][0] ; Y_0= tmp[1][0]; Z_0= tmp[2][0]

D = - beta / alpha
N = - gamma / alpha
#print "  Vereinfachte inhomogene Gleichung: x^2 - %s y^2 = %s "%(str(D), str(N))

v = (Y_0 + D*Y_0*m^2 - 2 * X_0 *m)/(1-D*m^2)
u = (v - Y_0)/m + X_0

r= S^-1*vector( [ u, v, 1] ).transpose()

sollteNullSein=(r.transpose()*G*r)[0][0]
if sollteNullSein !=0:
    raise ValueError("Hilfe!")

tmp=r*r[0].denominator()
print "inhomogen:\t[", r[0]/r[2],", ", r[1]/r[2], "]"

return r
```

```
#   Q(x,y) = a x^2+b 2xy + c y^2 + d 2x + e 2y+ f = 0
# Einheitskreis:
#r=giveParametrization(a=1 , b=0, c=1, d=0, e=0, f=-1 , x_0=0, y_0=-1)
# Parabel:
r=giveParametrization(a=0 , b=0, c=1, d=-1/2, e=0, f=0 , x_0=1/4, y_0=1/2)
# Hyperbel 1:
#r=giveParametrization(a=1 , b=0, c=-1, d=0, e=0, f=-1 , x_0=1, y_0=0)
# Hyperbel 2:
#r=giveParametrization(a=-1 , b=0, c=1, d=0, e=0, f=-1 , x_0=0, y_0=1)
# Ellipse
#r=giveParametrization(a=9 , b=0, c=25, d=0, e=0, f=-225 , x_0=5, y_0=0)
```

3 Fragen zur Vorlesung?

4 Fragen zu den Projekten?

5 Nächstes Mal

- Mehr zur Projektiven Geometrie

6 Quellcode

Das gesamte Worksheet ist als Text-Datei in dem PDF eingebettet.

- Im Acrobat-Reader lässt es sich unter dem Büroklammer-Symbol in der linken Leiste herunterladen.
- Okular zeigt es im File-Menu als Embeded Files an.

- Unter Linux kann man die Text-Datei auch mit
  pdftk Tutorium09.pdf unpack_files
  aus dem PDF herauslösen.

Anschließend lässt sich die Text-Datei mit der Upload-Funktion des SAGE-Notebooks hochladen.