SAGE-Tutorium 11 im SoSe 2009

Lars Fischer*

08.07.2009

Inhaltsverzeichnis

1 Wiederholung

- die Projektive Ebene in Bildern und Worten
- Kettenbrüche und Algorithmen für Kettenbrüche

2 Aufgaben vom Übungsblatt 9

Ich bin gebeten worden, die SAGE-Aufgaben des letzten Blattes zu besprechen.

Aufgabe 1: Schreibe eine Funktion `comp_line(x, y)`, die für zwei Punkte x, y der Projektiven Ebene die Gleichung der Geraden durch x, y berechnet.

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

Die theoretische Aufgabe 1 des Zettels, sagt hier schon alles. Wir können die Gleichung der Geraden in Normalenform aus dem Kreuzprodukt von x und y berechnen. In der letzten Sitzung habe ich gezeigt, wie sich diese Formel aus der Determinante gewinnen lässt.

```
#Aufgabe 9.2

def comp_line(P1,P2):
    """Berechnet die Gleichung der Geraden durch die Punkte P1,P2
       der projektiven Ebene.

    INPUT:
        P1,P2 -- Punkte der Projektiven Ebene in
                 homogenen Koordinaten (d.h. als Zahlen-Tripel)
    OUTPUT:
        Ein Zahlentripel, das den Normalenvektor angibt.

    EXAMPLES:
        sage: comp_line([1,2,1],[1,2,3])
        [4, -2, 0]
        sage: comp_line([1,2,1],[2,4,2])
        Traceback (most recent call last):
        ...
        ValueError: Die Punkte sind (als homogene Koordinaten) nicht verschieden.
    """

    # so bekommen wir x,y,z als benutzbare Namen
    R.<x,y,z> = PolynomialRing(QQ,3,"x,y,z")

    M= matrix([[x,y,z],list(P1),list(P2) ])
    L=M.det()
    if L==0:
        raise ValueError(
         "Die Punkte sind (als homogene Koordinaten) nicht verschieden."
        )

    return [ L.coefficient(x), L.coefficient(y), L.coefficient(z) ]
```

Aufgabe 2: Schreibe eine SAGE Funktion `conic_has_solutions(a,b,c)`, die für positive, quadratfreie, paarweise verschiedene, teilerfremde Zahlen a, b, c `True` oder `False` ausgibt, falls die Gleichung $ax^2 + by^2 = cz^2$ eine nichttriviale rationale Lösung besitzt oder nicht.

Zähle wieviele Tripel (a, b, c) von positiven, quadratfreien, teilerfremden ganzen Zahlen $a, b, c \leq 100$ es gibt. Bestimme weiterhin die Anzahl der Tripel $a, b, c \leq 100$ für die die Funktion `True` zurückgibt.

```
# Aufgabe 9.3

def TripelIstZulaessig(a,b,c):
    erg= (a >=0) and (b>= 0) and (c>= 0)
    if erg:
        erg= a.is_squarefree() and b.is_squarefree() and c.is_squarefree()
    if erg:
        erg= gcd(a,gcd(b,c)) == 1

    return erg


def conic_has_solution( a, b, c):
    if not TripelIstZulaessig(a ,b ,c):
        raise ValueError("Das Zahlentripel ist nicht zulaessig.")

    # nun sind a,b,c positiv, d.h. ax^2+by^2=cz^2 hat eine reelle Lsg
    # wir püfen noch die Bedingungen im Satz von Legendre
    # in der Vorlesung ist dieser für ax^2+by^2+cz^2 = formuliert,
    # also:
    c= -c

    #QRA= quadratic_residues( a )
    #QRB= quadratic_residues( b )
    #QRC= quadratic_residues( c )

    # dauert etwas länger als über die ... is_square Variante
    #erg=(Mod(-b*c, a) in QRA) and (Mod( -a*c, b ) in QRB)\
    #      and (Mod( -a*b, c ) in QRC)             # --> 33903
    erg = Mod(-b*c, a).is_square() \
          and Mod( -a*c,b ).is_square() \
          and Mod( -a*b, c ).is_square()   # --> 33903

    # ----------------------------------
    #  hierbei kommt etwas anderes heraus:
```

```
    #    das zeigt nochmals schön, dass jacobi_symbol ==1 nicht
    #    bedeutet, dass es ein QR ist,
    #    im Gegensatz zu legendre_symbol == 1
    # (kronecker_symbol berechnet das jacobi_symbol)
    #erg= (kronecker_symbol( -b*c,abs(a)) ==1 ) \
    #      and (kronecker_symbol( -a*c,abs(b)) ==1 ) \
    #      and (kronecker_symbol( -a*b,abs(c)) ==1 ) #--> 22237

    return erg



%time
Zulaessig= [ (a,b,c) for a in [1..100]
                     for b in [1..100]
                     for c in [1..100] if TripelIstZulaessig( a, b, c ) ]
print len(Zulaessig)


%time
Lsg= [ (a,b,c) for (a,b,c) in Zulaessig if conic_has_solution(a,b,c) ]
print len(Lsg)
```

Aufgabe 3: Gibt es eine Zahl n, so dass wir n Kanonenkugeln als Quadrat und als Pyramide mit quadratischer Grundfläche anordnen können?

(Tipp: Formuliert das Problem als diophantische Gleichung und entscheidet mit SAGE, ob es eine Lösung gibt oder nicht.)

Offensichtlich ist gefragt, ob es eine Zahl n gibt, die selber das Quadrat einer Zahl N ist. Weiterhin muss sich n als Summe der ersten M Quadrate schreiben lassen.

$$n = N^2 = \sum_{k=1}^{M} k^2 = \frac{M(M+1)(2M+1)}{6}, \quad \text{bzw.}$$
$$y^2 = \frac{x(x+1)(2x+1)}{6}$$

```
for M in [1..100]:
    summe = M*(M+1)*(2*M+1)/6
    if summe.is_square(): # summe ist immer in ZZ!
        N= sqrt(summe)
        print "N=%2d, M=%2d: n=N^2=%4d"%(N,M,summe)
```

3 Fragen zur Vorlesung?

Folgende Aufgaben passen zu den momentanen Themen der Vorlesung:

Aufgabe 4: Gegeben sei eine algebraische Kurve und eine Primzahl p. Zähle, wieviele Punkte die Kurve in der Projektiven Ebene über $\mathbb{F}_p$ hat.

```
def countPointsOnCurveModP( f, p):
    """Zählt die Punkte die f in der Projektiven Ebene über $\mathbb{F}_p$ hat."""
    F_p= GF(p)
    P= PolynomialRing( F_p, 3, 'xyz')

    tmp= f.homogenize('z')
    h= P( tmp )
    print h

    # den Anfang habe ich gemacht, den Rest koennt Ihr ergänzen.
```

Aufgabe 5: Bestimme die Schnittpunkte eine algebraischen Kurve f mit einer Geraden g, $f, g \in \mathbb{Q}[X, Y]$.

Erste Möglichkeit: die Gerade nach y auflösen und den Ausdruck in die Gleichung der Kurve einsetzen, lösen. Im Projektiven Fall noch nach eigentlichen und uneigentlichen Punkten unterscheiden.

Zweite Möglichkeit, allgemeiner:

- Bestimmung der Resultante (siehe auch `http://de.wikipedia.org/wiki/Resultante` bzw. die dortigen Literaturhinweise)
- Bestimmung der Nullstellen der Resultante
- Damit wieder in die Gleichung der Geraden einsetzen, Nullstellen der Geraden suchen

(Zugegebenermassen ist das in diesem Fall Overkill, da obige Schritte ebenfalls zu Ziel führen. Aber nur dass Ihr mal den Begriff der Resultante in Aktion gesehen habt.)

```
def find_intersection_points(f,g):
    print "Suche Schnittpunkte von %s und %s:"%(str(f),str(g) )

    # Polynomring und die Erzeuger zum Arbeiten:
    P = f.parent()
    x,y = P.gens()

    # Resultante von f und g in der Variablen y
    R= f.resultant(g, x)

    # die Linearfaktoren ergeben Kandidaten für mögliche Werten von y
    # an den gemeinsamen Nullstellen von f und g
    YKandidaten = [ -l[0](x=1,y=0) for l in R.factor() if l[0].degree() == 1 ]

    for yk in YKandidaten:
        # mit einem Kandidaten bestimmen wir durch Einsetzen und
        # Faktorisieren Kandidaten für Werte von x
```

```
        XKandidaten = [ -l[0](x=0,y=1) for l in f(x,yk).factor()
                                       if l[0].degree() == 1 ]
        for xk in XKandidaten:
            # xk und yk sind nur Kandidaten, wir erhalten aus f
            # zuviele xk, deswegen
            # prüfen wir durch Einsetzen in g
            if g(xk, yk) == 0 and f(xk, yk) == 0:
                print "  eingesetzt: x=%s , y=%s:\t f=%s, g=%s "%(
                    str(xk), str(yk), str( f(xk, yk) ),str(g(xk,yk))
                  )

P.<x,y> = PolynomialRing(QQ)

find_intersection_points( x^2+y^2 -1 ,  x+1 -y  )
# dieses Resultanten Ungeheuer geht auch für kompliziertere g als Geraden:
find_intersection_points( x^2+y^2 -1 ,  3*x^2+y^2-1  )
find_intersection_points( x^3+y^3 -1 ,  x^2+y^2-1  )
```

4 Aufgaben

Aufgabe 6: Berechne $3^{2009} \pmod{11}$.

Aufgabe 7: Löse $x^2 + 13x - 78 \equiv 0 \pmod{65}$.

Aufgabe 8: Ist die Kongruenz $81x \equiv 1 \pmod{103}$ lösbar, wenn ja, bestimme x.

5 Fragen zu den Projekten?

Welchen Termin habt Ihr mit Prof. Skoruppa für die Präsentation vereinbart?

Nächste Woche (nur nicht Mittwoch) oder Anfang Oktober?

Zur Ausstellung der Scheine benötigen wir die folgenden Informationen:

- Name

- Matrikelnummer
- Studiengang (genaue Bezeichnung)
- Geburtstag
- Anschrift

Sendet mir diese Daten bitte per EMail zu.

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 Tutorium11.pdf unpack_files
  aus dem PDF herauslösen.

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