Wybrane Zagadnienia Algebry
Sprawozdanie 3

Rafał Włodarczyk 279762

Czerwiec 2026

1 (a) Pierścień Wielomianów Wielu Zmiennych

Strukturę danych dla pierścieni wielomianów wielu zmiennych $\mathbb{R}[x_1, \ldots, x_n]$ implementuję jako mapę *klucz* $\rightarrow$ *wartość*, w której kluczem jest wektor wykładników (jednomian bez współczynnika), a wartością jest współczynnik zadanego jednomianu. Nieobecne w strukturze klucze (jednomiany) są interpretowane jako mające współczynnik równy zero. Przykład. Funkcja f (Ćw. 37.) dana wzorem $f = x^3 - x^2 \cdot y - x^2 \cdot z$ będzie reprezentowana jako trzy elementy w mapie:

$$\begin{aligned} \{(3,0,0) \rightarrow 1\} \quad &\text{(odpowiada } 1 \cdot x^3\text{)}, \\ \{(2,1,0) \rightarrow -1\} \quad &\text{(odpowiada } -1 \cdot x^2 \cdot y^1\text{)}, \\ \{(2,0,1) \rightarrow -1\} \quad &\text{(odpowiada } -1 \cdot x^2 \cdot z^1\text{)}. \end{aligned}$$

Wielomiany jednej zmiennej można przechowywać w prostszej postaci zwykłej tablicy. Operacje pierścieniowe ulegają zmianie - część wymaga porządku na jednomianach, który nie jest potrzebny w przypadku wielomianów jednej zmiennej.

2 (b) PolynomialReduce, Syzygium, algorytm Buchbergera

2.1 PolynomialReduce

Algorytm `PolynomialReduce` jest tożsamy z poprzednim sprawozdaniem. Jest to algorytm pobierający na wejściu wielomian f oraz ciąg wielomianów $\mathcal{G} = (g_1, \ldots, g_n)$. Algorytm zwraca ciąg współczynników $(a_1, \ldots, a_n)$ oraz wielomian r taki, że:

$$f = \sum_{i=1}^{n} \alpha_i \cdot g_i + r$$

Wykorzystując jeden ze wcześniej zdefiniowanych porządków na jednomianach dokonujemy redukcji wielomianu f względem ciągu $\mathcal{G}$

1. Dopóki $f \neq 0$. Znajdź $m =$ `LT(f, ord)` - największy jednomian m w f według porządku *ord*. Ustal $LC(m) = a$ oraz $LM(m) = M$. (Leading cicient oraz Leading Monomial odpowiednio).
2. Dla każdego g_i w $\mathcal{G}$ znajdź $m_i =$ `LT(g_i, ord)`, czyli największy jednomian m_i w g_i według *ord*. Ustal $LC(m_i) = b$ oraz $LM(m_i) = N$.
3. Sprawdź czy $LM(m_i)$ dzieli $LM(m)$. Spełniony musi zostać warunek: $(\forall j : LM(m)[j] \geqslant LM(m_i)[j])$.
4. Jeśli tak:
   (a) Wykonaj redukcję: $f := f - \frac{LC(m)}{LC(m_i)} \cdot \frac{LM(m)}{LM(m_i)} \cdot g_i$
   (b) Zaktualizuj współczynnik $\alpha_i := \alpha_i + \frac{LC(m)}{LC(m_i)} \cdot \frac{LM(m)}{LM(m_i)}$.
   (c) Wróć do kroku 1.
5. Jeśli nie, dodaj $LM(m)$ do r(eszty) ze współczynnikiem $LC(m)$ i usuń m z f. Wróć do kroku 1.

```cpp
inline std::tuple<std::vector<poly_t>, poly_t>
poly_reduce(
    // Wielomian f
    const poly_t& f,
    // Ciąg wielomianów G = (g_1, ..., g_n)
    const std::vector<poly_t>& G,
    // ord na jednomianach
    bool (*compare)(const monomian_t&, const monomian_t&)
)
{
    size_t n = G.size();
    std::vector<poly_t> q(n);
    poly_t r;
    poly_t p = f;

    // (1) Dopóki p != 0
    while (!p.empty()) {
        // LT(f) = aM, LM(f) = M, LC(f) = a
        auto lt_p_it = get_lt(p, compare);
        const monomian_t lm_p = lt_p_it->first;
        int64_t lc_p = lt_p_it->second;

        // Dla każdego g_i w G
        bool divided = false;
        for (size_t i = 0; i < n; ++i) {
            if (G[i].empty()) continue;

            // (2) Znajdź LT(g_i) = bN, LM(g_i) = N, LC(g_i) = b
            auto lt_g_it = get_lt(G[i], compare);
            const auto& lm_g = lt_g_it->first;
            int64_t lc_g = lt_g_it->second;

            // (3) Sprawdz czy LM(g_i) dzieli LM(f) (iteruj po wykładnikach)
            bool divisible = true;
            for (size_t j = 0; j < lm_p.size(); ++j) {
                if (lm_p[j] < lm_g[j]) {
                    divisible = false;
                    break;
                }
            }

            // (4) Jeśli tak, wykonaj redukcję
            if (divisible) {
                // Oblicz q = LM(f) / LM(g_i) (iteruj po wykładnikach)
                monomian_t q_mon;
                for (size_t j = 0; j < lm_p.size(); ++j) {
                    q_mon.push_back(lm_p[j] - lm_g[j]);
                }

                // Oblicz q_coef = LC(f) / LC(g_i)
                int64_t q_coef = lc_p / lc_g;
                q[i][q_mon] += q_coef;

                // Zaktualizuj p := p - q_coef * q_mon * g_i
                for (const auto& [mon_g, coef_g] : G[i]) {
                    monomian_t new_mon;
                    for (size_t j = 0; j < mon_g.size(); ++j) {
                        new_mon.push_back(q_mon[j] + mon_g[j]);
                    }
                    p[new_mon] -= coef_g * q_coef;
                    if (p[new_mon] == 0) p.erase(new_mon);
                }

                divided = true;
                break;
            }
        }

        // (5) Jeśli nie, dodaj LM(f) do r ze współczynnikiem LC(f) i usuń LT(f) z p
        if (!divided) {
            r[lm_p] += lc_p;
            p.erase(lm_p);
        }
    }

    // Zwracamy krotkę ({q_1, ... q_n}, r)
    return {q, r};
}
```

2.2 Syzygium

Służy do eliminacji wiodących wyrazów pary wielomianów f_i oraz f_j. Na wejściu znajduja się dwa wielomiany f_i, f_j, ich $LCM = \text{LCM}(LM(f_i), LM(f_j))$ (najmniejsza wspólna wielokrotność) porządek jednomianów *ord*. Wynikiem działania jest wielomian $S(f_i, f_j)$ zdefiniowany jako:

$$S(f_i, f_j) = \frac{LCM}{LT(f_i)} \cdot f_i - \frac{LCM}{LT(f_j)} \cdot f_j$$

Algorytm działa w następujących krokach:

1. Wyznacz wiodące składniki f_i: $LT(f_i) = (lm_i, lc_i)$ oraz f_j: $LT(f_j) = (lm_j, lc_j)$.
2. $\forall (m, c) \in f_i$ (para jednomian, współczynnik):
   (a) Wyznacz $m\_new := LCM + (m - lm_i)$.
   (b) Dodaj do $S(f_i, f_j)$ wartość $c \cdot lc_j$ pod kluczem m_new.
3. $\forall (m, c) \in f_j$:
   (a) Wyznacz $m\_new := LCM + (m - lm_j)$.
   (b) Odejmij od $S(f_i, f_j)$ wartość $c \cdot lc_i$ pod kluczem m_new.
   (c) Usuń jednomian z $S(f_i, f_j)$ jeśli powyższa operacja ustaliła $c\_new = 0$.

```
poly_t fi = G[i]; poly_t fj = G[j];
auto lt_i = get_lt(fi, order); auto lt_j = get_lt(fj, order);
monomian_t lcm = monomial_lcm(lt_i->first, lt_j->first);
double_t lc_i = lt_i->second; double_t lc_j = lt_j->second;

// 3. Syzygy over all terms
poly_t syzygy;

// (+) (LCM / LT(f_i)) * f_i * lc_j
for (const auto &[m, c] : fi)
{
    monomian_t m_new = lcm;
    for (size_t k = 0; k < m.size(); ++k)
        m_new[k] += (m[k] - lt_i->first[k]);
    syzygy[m_new] += c * lc_j;
}

// (-) (LCM / LT(f_j)) * f_j * lc_i
for (const auto &[m, c] : fj)
{
    monomian_t m_new = lcm;
    for (size_t k = 0; k < m.size(); ++k)
        m_new[k] += (m[k] - lt_j->first[k]);
    syzygy[m_new] -= c * lc_i;
    if (syzygy[m_new] == 0) syzygy.erase(m_new);
}
```

2.3 Algorytm Buchbergera

Służy do wyznaczania bazy Grobnera dla ciągu wielomianów F oraz zadanego ideału $\mathcal{G} = \{g_1, \ldots, g_n\}$. Algorytm pobiera ciąg wielomianow F oraz porządek jednomianów *ord*, a zwraca zredukowaną bazę Grobnera w postaci ciągu wielomianów $G_{reduced}$.

1. Skopiuj wielomiany do bazy $G \leftarrow F$.
2. Dla wszystkich par (i, j) z G oblicz najmniejszą wspólną wielokrotność wiodących jednomianów i dodaj pary do kolejki.
3. Wyznacz Syzygium S_{ij} dla każdej pary (i, j) w kolejce.
4. Zredukuj S_{ij} względem aktualnego zbioru G.
5. Jeśli reszta jest niezerowa, rozszerz zbiór G i dodaj nowe pary do kolejki.
6. Po sprawdzeniu wszystkich elementów w kolejce, zredukuj G - to wynikowa baza Grobnera.

```
// 1. G <- F
std::vector<poly_t> G;
for (const auto &f : F) if (!f.empty()) G.push_back(f);

using index_pair_t = std::pair<size_t, size_t>;
std::queue<index_pair_t> pairs;
// 2. Dla wszystkich par (i,j) stwórz kolejkę Q
for (size_t i = 0; i < G.size(); ++i)
    for (size_t j = i + 1; j < G.size(); ++j)
        pairs.push({i, j});

// Sprawdzaj kolejne pary z kolejki
while (!pairs.empty())
{
    auto [i, j] = pairs.front(); pairs.pop();
    poly_t fi = G[i]; poly_t fj = G[j];
    auto lt_i = get_lt(fi, order); auto lt_j = get_lt(fj, order);
    monomian_t lcm = monomial_lcm(lt_i->first, lt_j->first);
    double_t lc_i = lt_i->second; double_t lc_j = lt_j->second;

    // 3. Wyznacz syzygium f_i, f_j, opisane w poprzednim algorytmie
    poly_t syzygy = Syzygy... ( see previous )
    if (syzygy.empty()) continue;

    // 4. Redukuj S(f_i,f-J) po aktualnym G
    auto [quotients, remainder] = poly_reduce(syzygy, G, order);

    // 5. Jeśli reszta jest niezerowa dodaj ją do G i nowo stworzone pary do Q
    if (!remainder.empty())
    {
        size_t new_index = G.size();
        for (size_t k = 0; k < new_index; ++k)
        {
            pairs.push({k, new_index});
        }
        G.push_back(remainder);
    }
}
```

3 (c) Układy Równań Krzywych Stożkowych

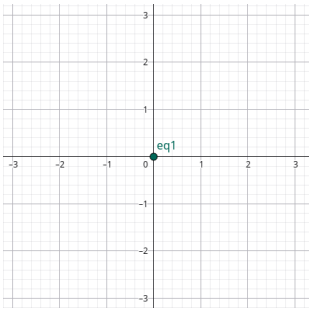
3.1 Układ Pierwszy

Baza Grobnera dla następującego układu równań:

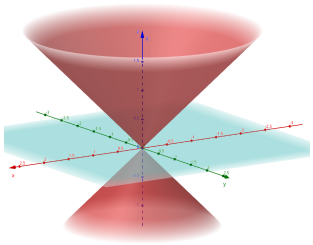
$$\begin{cases} x^2+y^2=z^2 \\ (c+1)z=0 \end{cases} \xrightarrow[z>x>y]{\text{GrobnerBasis}} \begin{cases} x^2+y^2 \\ z \end{cases}$$

$$\implies \mathcal{I}_f = x^2+y^2$$

Drugie równanie $10z = 0$ wymusza $z = 0$. Przekrajamy dwustronnie nieskończony stożek z płaszczyzną przechodzącą idealnie przez wierzchołek stożka. Jedynym rozwiązaniem jest więc jeden punkt - wierzchołek.



$\mathcal{V}(x^2+y^2)$



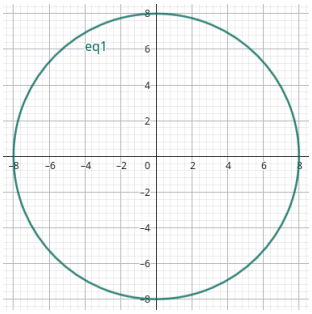
Wykres 3D (1)

3.2 Układ Drugi

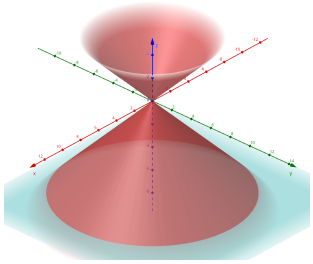
Baza Grobnera dla następującego układu równań:

$$\begin{cases} x^2+y^2=z^2 \\ z+8=0 \end{cases} \xrightarrow[\text{z > x > y}]{\text{GrobnerBasis}} \begin{cases} x^2+y^2-64 \\ z+8 \end{cases}$$
$$\implies \mathcal{I}_f = x^2+y^2-64$$

Drugie równanie $z = -8$. Przekrajamy dwustronnie nieskończony stożek z płaszczyzną przechodzącą powyżej jego wierzchołka równolegle do osi XY. Zbiór rozwiązań jest zatem równaniem okręgu.



$\mathcal{V}(x^2+y^2-64)$



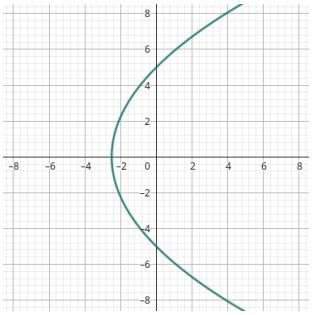
Wykres 3D (2)

3.3 Układ Trzeci

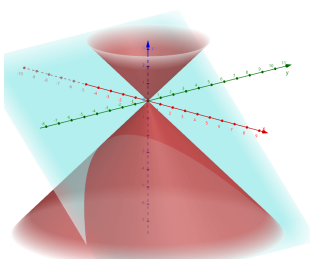
Baza Grobnera dla następującego układu równań:

$$\begin{cases} x^2+y^2=z^2 \\ x+z+5=0 \end{cases} \xrightarrow[\text{z > x > y}]{\text{GrobnerBasis}} \begin{cases} x+z+5 \\ y^2-10x+25 \end{cases}$$
$$\implies \mathcal{I}_f = y^2-10x-25$$

Drugie równanie $x+z+5=0$. Przekrajamy dwustronnie nieskończony stożek z płaszczyzną przechodzącą pod kątem prostopadłym do jego ściany, otrzymując kolejną krzywą stożkową - parabolę.



$\mathcal{V}(y^2-10x-25)$



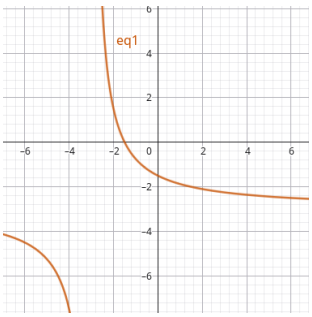
Wykres 3D (3)

3.4 Układ Czwarty

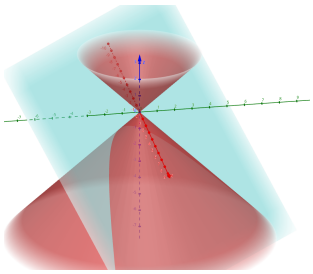
Baza Grobnera dla następującego układu równań:

$$\begin{cases} x^2+y^2=z^2 \\ x+y+z+3=0 \end{cases}$$
$$\xrightarrow[\text{z > x > y}]{\text{GrobnerBasis}} \begin{cases} x+y+z+3 \\ xy+3x+3y+4.5 \end{cases}$$
$$\implies \mathcal{I}_f = xy+3x+3y+4.5$$

Drugie równanie $x+y+z+3=0$. Przekrajamy dwustronnie nieskończony stożek z płaszczyzną przechodzącą pod kątem większym niż 45 stopni do XY, otrzymując kolejną krzywą stożkową - hiperbolę.



$\mathcal{V}(xy+3x+3y+4.5)$



Wykres 3D (4)

3.5 Układ Piąty

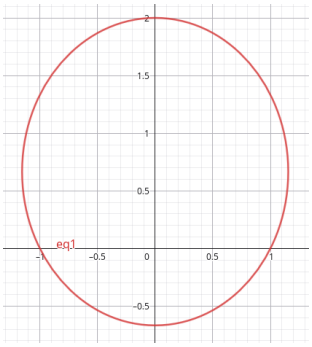
Baza Grobnera dla następującego układu równań:

$$\begin{cases} x^2 + y^2 = z^2 \\ y + 2z + 2 \end{cases}$$

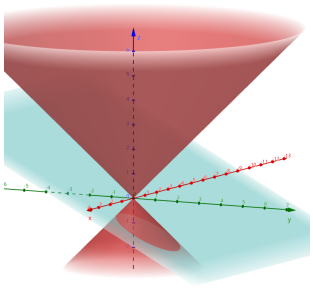
$$\xrightarrow[z > x > y]{\text{GrobnerBasis}} \begin{cases} y + 2z + 2 \\ 4x^2 + 3y^2 - 4y - 4 \end{cases}$$

$$\implies \mathcal{I}_f = 4x^2 + 3y^2 - 4y - 4$$

Przekrajamy dwustronnie nieskończony stożek z płaszczyzną przechodzącą pod kątem mniejszym niż 45 stopni XY, otrzymując kolejną krzywą stożkową - elipsę.



$\mathcal{V}(4x^2 + 3y^2 - 4y - 4)$



Wykres 3D (5)

4 (d) Baza oraz Eliminacja zmiennej

Dla zadanego układu wyznaczam Bazę Grobnera przy użyciu mojej implementacji algorytmu Buchbergera.

$$\begin{cases} (x^2 + y^2 - ax)^2 = z^2(x^2 + y^2) \\ x + 2y + 3z = 0 \end{cases}$$

Koduję wejście - wielomiany wielu zmiennych (rozszerzam $(a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2bc + 2ca$)):

```
const poly_t D1 = {
    {{2, 0, 0}, a * a},
    {{3, 0, 0}, -2 * a},
    {{1, 2, 0}, -2 * a},
    {{4, 0, 0}, 1},
    {{2, 2, 0}, 2},
    {{0, 4, 0}, 1},
    {{2, 0, 2}, -1},
    {{0, 2, 2}, -1}
};
```

```
const poly_t D2 = {
    {{1, 0, 0}, 1},
    {{0, 1, 0}, 2},
    {{0, 0, 1}, 3}
};
```

Wynik działania programu:

```
Grobner basis von
(x^4) - 4(x^3) + 2(x^2)(y^2)
- (x^2)(z^2) + 4(x^2) - 4(x)(y^2)
+ (y^4) - (y^2)(z^2)
und
(x) + 2(y) + 3(z)
Computed Groebner basis:
1. (x) + 2(y) + 3(z)
2. (y^4) + 4.80(y^3)(z) + 1.60(y^3) + 9.160(y^2)(z^2)
   + 6.240(y^2)(z) + 0.640(y^2) + 8.160(y)(z^3)
   + 8.640(y)(z^2) + 1.920(y)(z) + 2.880(z^4) + 4.320(z^3) + 1.440(z^2)
```

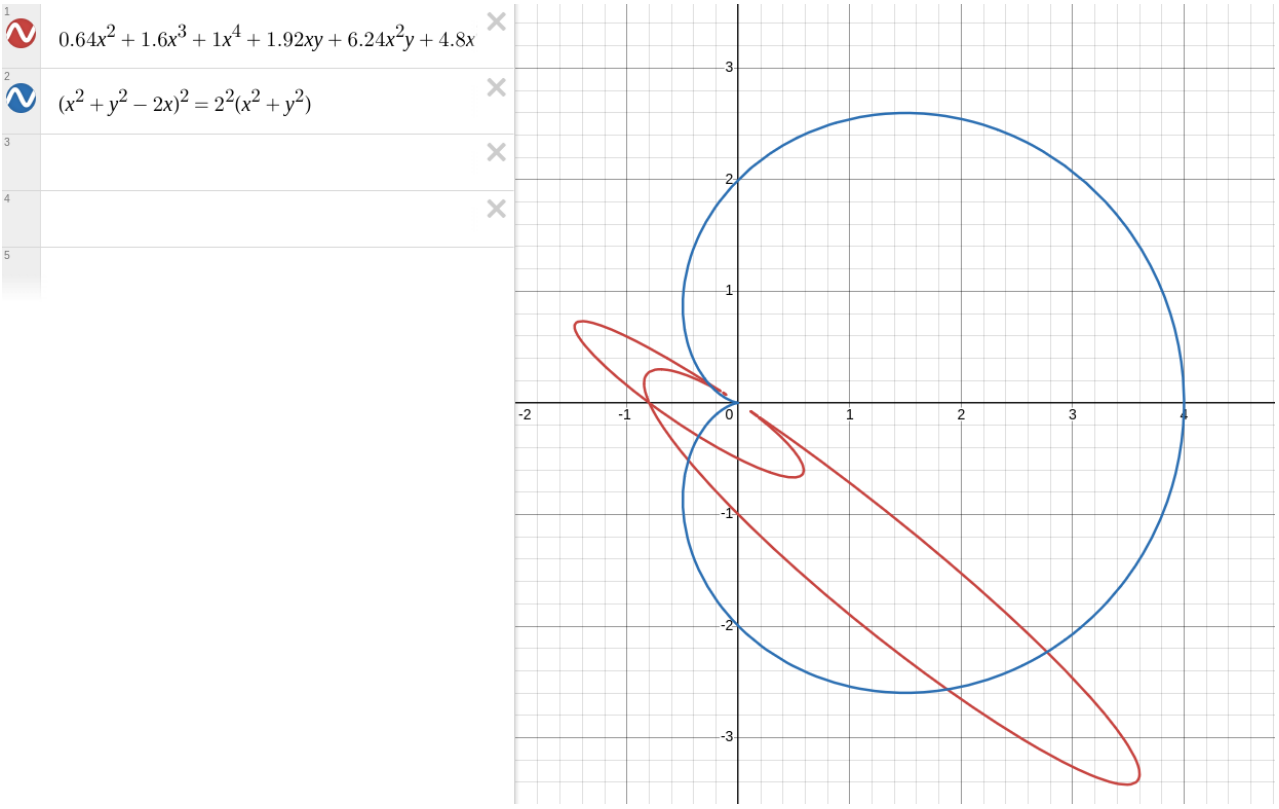
Wybieramy drugie wyrażenie, nieposiadające zmiennej x. W celu wykonania wykresu tego równania $w(y,z) = 0$ wprowadzamy podstawienie $x \leftarrow y, y \leftarrow z$ i rysujemy w układzie kartezjańskim (x, y).

$$0.64x^2 + 1.6x^3 + 1x^4 + 1.92xy + 6.24x^2y + 4.8x^3y + 1.44y^2 + 8.64xy^2 + 9.16x^2y^2 + 4.32y^3 + 8.16xy^3 + 2.88y^4 = 0$$

W tym podpunkcie należy również porównać wynik z podstawieniem $z = a$ do pierwszego z równań, zachodzi:

$$(*) \quad (x^2 + y^2 - ax)^2 = a^2(x^2 + y^2)$$

Kolorem czerwonym oznaczona jest krzywa policzona za pomocą bazy Grobnera. Kolorem niebieskim oznaczone jest równianie $(*)$, dla $a = 2$ - jest to kardioida - często spotykana np. w muzyce - stanowi profil zbierania dźwięku mikrofonów pojemnościowych.



5 (e) Baza Grobnera oraz Trysektrysa

Dla zadanej funkcji w układzie biegunowym wprowadzamy podstawienie $\cos(\theta) = \frac{x}{r}$, tak jak na poprzedniej liście w celu przejścia do układu kartezjańskiego.

$$r = b\left(4\cos(\theta) + \frac{1}{\cos(\theta)}\right) = b\left(4\frac{x}{r} + \frac{r}{x}\right)$$
$$b\left(\frac{4x^2 + r^2}{rx}\right) - r = 0$$
$$b(4x^2 + r^2) - r^2x = 0$$
$$br^2 + 4bx^2 - r^2x = 0$$

Mamy następujący układ równań:

$$\begin{cases} x^2 + y^2 - r^2 = 0 \\ br^2 + 4bx^2 - r^2x = 0 \end{cases}$$

Wyciągnijmy z pomocą Bazy Grobnera wyrażenia zawierające jedynie x, y. Wynik działania programu:

```
Grobner basis von -(r^2) + (y^2) + (x^2) und -(r^2)(x) + 7(r^2) + 28(x^2)
Computed Groebner basis:
1. (r^2) - (y^2) - (x^2)
2. (y^2)(x) - 7(y^2) + (x^3) - 35(x^2)
```

Zauważmy, że drugi element bazy składa się tylko z x, y. Jest on zatem naszym rozwiązaniem:

$$f(x, y) = y^2 \cdot x - 7y^2 + x^3 - 35x^2 = 0$$

5.1 Wykresy Podanego Równania

Po lewej stronie narysowałem krzywą w Geogebrze za pomocą wyjściowego równania $r(\theta)$ w układzie biegunowym. Po prawej stronie narysowałem te samą krzywą za pomocą równania dwóch zmiennych $f(x, y) = 0$

b = 7

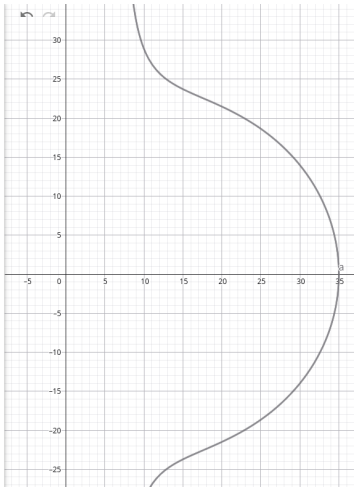
$f(x) = 7\left(4\cos(x) + \frac{1}{\cos(x)}\right)$

$a = \text{Curve}(f(t)\cos(t), f(t)\sin(t), t, 0, 2\pi)$

$$= \left\{ \begin{array}{l} x = 7\left(4\cos(t) + \frac{1}{\cos(t)}\right)\cos(t) \\ y = 7\left(4\cos(t) + \frac{1}{\cos(t)}\right)\sin(t) \end{array} \right\} 0 \le t \le 6$$

eq1: $y^2 x - 7y^2 + x^3 - 35x^2 = 0$

Input...



b = 7

$f(x) = 7\left(4\cos(x) + \frac{1}{\cos(x)}\right)$

$a = \text{Curve}(f(t)\cos(t), f(t)\sin(t), t, 0, 2\pi)$

$$= \left\{ \begin{array}{l} x = 7\left(4\cos(t) + \frac{1}{\cos(t)}\right)\cos(t) \\ y = 7\left(4\cos(t) + \frac{1}{\cos(t)}\right)\sin(t) \end{array} \right\} 0 \le t \le 6$$

eq1: $y^2 x - 7y^2 + x^3 - 35x^2 = 0$

Input...