Wybrane Zagadnienia Algebry
Sprawozdanie 2

Rafał Włodarczyk 279762

Czerwiec 2026

Podsumowanie

Udało mi się wykonać wszystkie zadania oraz zaproponować dwa sensowne przybliżenia zadanych figur geometrycznych - Kamon Cesarza Japonii oraz figurę z japońskiego paszportu.

Zadanie 4

a) Struktura przechowująca wielomiany wielu zmiennych rzeczywistych

Proponowany kontener to mapa *klucz* $\to$ *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:

$$\{(3,0,0) \to 1\} \quad \text{(odpowiada } 1 \cdot x^3\text{)},$$
$$\{(2,1,0) \to -1\} \quad \text{(odpowiada } -1 \cdot x^2 \cdot y^1\text{)},$$
$$\{(2,0,1) \to -1\} \quad \text{(odpowiada } -1 \cdot x^2 \cdot z^1\text{)}.$$

Zalety struktury: W przypadku wielomianów zadanych w tym sprawozdaniu (dla których większość współczynników jest równa zero) implementacja z mapą pozwala uniknąć przechowywania wielu zer, jednocześnie zachowując szybkie odwołania do każdego ze współczynników oraz całej listy *klucz* oraz całej listy *wartość*. Realizacja w C++ wygląda następująco:

```
using monomian_t = std::vector<int64_t>;
using poly_t = std::unordered_map<monomian_t, int64_t, container_hash<monomian_t>>;
```

Wykorzystujemy kontenery `vector` oraz `unordered_map` z STL oraz `container_hash` do haszowania wektorów wykładników z Boost.

b) Porządki na jednomianach

Mając jednomian $aM = a \cdot x_1^{e_1} \cdot x_2^{e_2} \cdots x_n^{e_n}$, gdzie a jest współczynnikiem, a e_i są wykładnikami zmiennych x_i, porządki na jednomianach definiują sposób porównywania dwóch jednomianów w celu ustalenia, który z nich jest "większy"lub "mniejszy zostanie wpierw wykorzystany do porównywania jednomianów w `PolynomialReduce`.

ba) Standard Lex

Standard Lex porównuje jednomiany zakładając ustaloną kolejność zmiennych $(x_1 > x_2 > \ldots > x_n)$ i porównując wykładniki zmiennych w tej kolejności.

$$(M = a \cdot x_1^{e_1} x_2^{e_2} \ldots) <_1 (N = b \cdot x_1^{f_1} x_2^{f_2} \ldots) \iff ((\exists i \in \{1,\ldots,n\})\,(e_i < f_i) \wedge (\forall j < i : e_j = f_j))$$

bb) Permutowany Lex

Permutowany Lex jest modyfikacją Standard Lex, która dodatkowo przyjmuje permutację π. Porównuje jednomiany zakładając ustaloną kolejność zmiennych $(x_{\pi(1)} > x_{\pi(2)} > \ldots > x_{\pi(n)})$ i porównując wykładniki zmiennych w tej permutowanej kolejności.

$$M <_2 N \iff \left((\exists i \in \{1,\ldots,n\})\left(e_{\pi(i)} < f_{\pi(i)}\right) \wedge \left(\forall j < i : e_{\pi(j)} = f_{\pi(j)}\right)\right)$$

bc) Graded Lex

Graded Lex jest hybrydą Standard Lex i porządku stopniowego. Najpierw porównuje jednomiany według sumy wykładników (stopnia), a jeśli są równe to porównuje je według Standard Lex.

$$M <_3 N \iff \left(\sum_{i=1}^{n} e_i < \sum_{i=1}^{n} f_i\right) \quad else \quad M <_1 N$$

c) `PolynomialReduce`

`PolynomialReduce` 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:

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

Idea algorytmu

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 Coefficient 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.

Kod w C++

```
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};
}
```

d) Rozwiązanie ćwiczenia 37 z listy zadań

Rozważamy porządek *GradedLex* oraz wielomiany $f = x^3 - x^2y - x^2z$, $g_1 = x^2y - z$, $g_2 = xy - 1$. Redukujemy przy użyciu funkcji `PolynomialReduce`. Wprowadzam do programu następujące kodowanie parametrów:

```
poly_t poly{
    {{3,0,0},{1}},
    {{2,1,0},{-1}},
    {{2,0,1},{-1}}
};
```

```
poly_t g1{
    {{2,1,0},{1}},
    {{0,0,1},{-1}}
};
```

```
poly_t g2{
    {{1,1,0},{1}},
    {{0,0,0},{-1}}
};
```

Wynikiem działania programu są następujące współczynniki oraz reszty:

```
r1=(x^3) - (x^2)(z) - (z)
a(g0) = -1
a(g1) = 0
r2=(x^3) - (x^2)(z) - (x)
a(g0) = -(x)
a(g1) = 0
```

Wynik interpretowany (w pierwszym przypadku $g_1 \leftarrow$ `g0`, $g_2 \leftarrow$ `g1`, a w drugim $g_2 \leftarrow$ `g0`, $g_1 \leftarrow$ `g1`):

$$f = -1 \cdot g_1 + 0 \cdot g_2 + (x^3 - x^2z - z)$$
$$f = -x \cdot g_2 + 0 \cdot g_1 + (x^3 - x^2z - x)$$

Reszty są różne, ponieważ `PolynomialReduce` redukuje po pierwszym spełniającym warunek dzielniku. Wynik zależy więc od kolejności wielomianów w ciągu $\mathcal{G}$.

Czy $r_1 - r_2 \in \langle g_1, g_2 \rangle$? **Tak**.
Z definicji ideał $\langle g_1, g_2 \rangle$ to zbiór wszystkich kombinacji liniowych wielomianów g_1 i g_2, czyli:

$$a \cdot g_1 + b \cdot g_2 \quad \text{dla dowolnych } a, b \in \mathbb{R}[x, y, z]$$

Zauważmy teraz, że różnica reszt:

$$r_1 - r_2 = (a \cdot g_1 + b \cdot g_2) - (a' \cdot g_1 + b' \cdot g_2) = (a - a') \cdot g_1 + (b - b') \cdot g_2$$

Skoro $a, a', b, b' \in \mathbb{R}[x, y, z]$, to $a - a'$ oraz $b - b'$ również należą do $\mathbb{R}[x, y, z]$. Oznacza to, że $r_1 - r_2$ jest kombinacją liniową g_1 i g_2, a więc należy do ideału $\langle g_1, g_2 \rangle$.

e) Wpływ porządków na wynik `PolynomialReduce`

Weźmy h takie że dla danych $(a, b, c, d, e, f) = (2, 7, 9, 7, 6, 2)$:

$$h(x, y, z) = x^a y^b - y^c z^d + x^e z^f \in \mathbb{R}[x, y, z]$$
$$h(x, y, z) = x^2 y^7 - y^9 z^7 + x^6 z^2$$

Pokażmy, że istnieje taki ciąg wielomianów $\mathcal{G}$, że wynik redukcji h względem $\mathcal{G}$ będzie różny dla porządków Standard Lex, Permutowany Lex z $\pi_1 = (2, 1, 0)$ oraz Permutowany Lex z $\pi_2 = (1, 2, 0)$ Najprościej wymyślić wielomian g_1, taki że każdy z porządków wybierze inny jednomian jako największy.
Weźmy najprościej jednoelementowy ciąg $\mathcal{G} = (g_1)$, gdzie $g_1 = x^5 + y^6 + z^5$. Wtedy:

1. Porządek Standard Lex ($x > y > z$) wybierze najpierw x^5 jako największy jednomian w g_1, więc redukcja nastąpi po x^5.
2. Porządek Permutowany Lex z permutacją $\pi_1 = (2, 1, 0)$ ($z > y > x$) wybierze najpierw z^5 jako największy jednomian w g_1, więc redukcja nastąpi po z^5.
3. Porządek Permutowany Lex z permutacją $\pi_2 = (1, 2, 0)$ wybierze najpierw y^6 jako największy jednomian w g_1 (stopień 6 jest większy niż stopień 5), więc redukcja nastąpi po y^6.

Zobaczmy, że implementacja `PolynomialReduce` potwierdza tę intuicję. Dla każdego z porządków otrzymujemy inną resztę oraz inny współczynnik przy g_1.

Wynik Programu:

```
Formatted polynomial: -(y^9)(z^7) + (x^2)(y^7) + (x^6)(z^2)
g1: (x^5) + (y^6) + (z^5)
Remainder (lex): (x^2)(y^7) - (x)(y^6)(z^2) - (x)(z^7) - (y^9)(z^7)
a(g0) = (x)(z^2)                    // (x>y>z) zredukowaliśmy po x^5,
                                    // nic nie jest większe niż x^5
Remainder (ordered lex): (x^6)(z^2) + (x^5)(y^9)(z^2) + (x^2)(y^7) + (y^15)(z^2)
a(g0) = -(y^9)(z^2)                 // (z>y>x) pi1 : zredukowaliśmy po z^5
                                    // więc nic nie jest większe niż z^5
Remainder (ordered lex): -(x^7)(y) + (x^6)(z^2) + (x^5)(y^3)(z^7) - (x^2)(y)(z^5) + (y^3)(z^12)
a(g0) = (x^2)(y) - (y^3)(z^7)       // (y>z>x) pi2 : zredukowaliśmy po y^6
                                    // więc nic nie jest większe niż y^6
Remainder (graded lex): -(x^7)(y) + (x^6)(z^2) + (x^5)(y^3)(z^7) - (x^2)(y)(z^5) + (y^3)(z^12)
a(g0) = (x^2)(y) - (y^3)(z^7)       // powtórzona sytuacja z ordered lex,
                                    // dla ciekawostki bo mamy już 3
                                    // powyższe lex-y dające różne wyniki
```

Zapis wyników w formie matematycznej. Jak widać wszystkie reszty i wszystkie współczynniki przy g_1 są różne.

$$\alpha_{\text{lex}}(g_1) = xz^2$$
$$r_{\text{lex}} = x^2y^7 - xy^6z^2 - xz^7 - y^9z^7$$
$$\alpha_{\text{ordered\_lex}(\pi_1)}(g_1) = -y^9z^2$$
$$r_{\text{ordered\_lex}(\pi_1)} = x^6z^2 + x^5y^9z^2 + x^2y^7 + y^{15}z^2$$
$$\alpha_{\text{ordered\_lex}(\pi_2)}(g_1) = x^2y - y^3z^7$$
$$r_{\text{ordered\_lex}(\pi_2)} = -x^7y + x^6z^2 + x^5y^3z^7 - x^2yz^5 + y^3z^{12}$$

Żeby uniknąć takiej sytuacji możemy spróbować wkorzystać bazę Grobnera zamiast dowolnego ciągu $\mathcal{G}$, ponieważ ta baza jest definiowana jako taka, która dla każdego wielomianu f daje jednoznaczną resztę r niezależnie od porządku na jednomianach.

Zadanie 5

Figura z Japońskiego Paszportu w oparciu o Quadrifolium

Weźmy Quadrifolium, czyli krzywą zadaną równaniem $r = \sin(2\theta)$ w układzie biegunowym. $\theta \in [0, 2\pi]$.

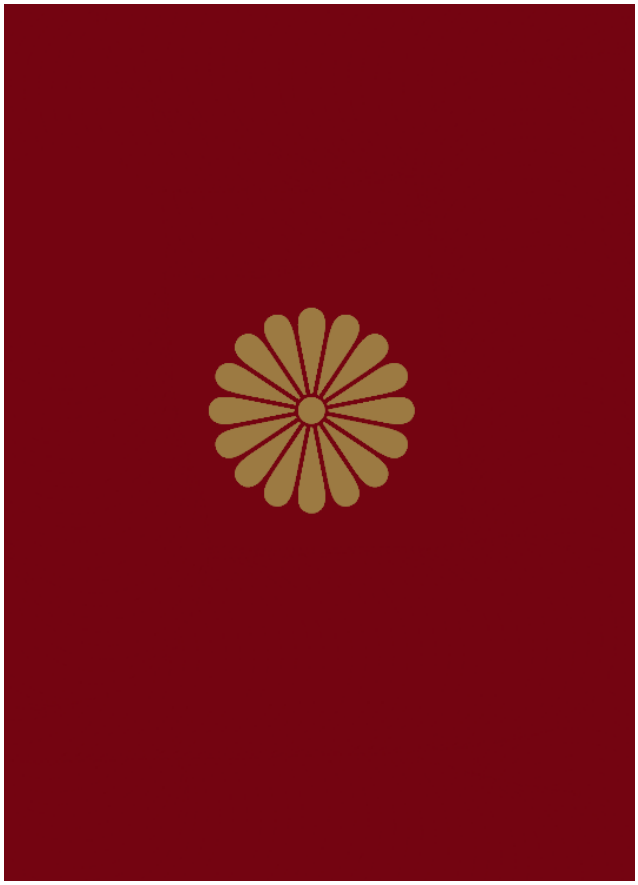
$$
\begin{aligned}
&r(\theta) = \sin(2\theta) \quad \text{stosujemy powinowactwo prostokątne } r(\theta) \xrightarrow{P(k=\frac{1}{4})} r(4\theta) \\
&r(\theta) = \sin(8\theta) \quad \text{nakładamy wartość bezwzględną } r(\theta) \xrightarrow{|\cdot|} |r(\theta)| \\
&r(\theta) = |\sin(8\theta)| \quad \text{wygładzamy poprzez potęgę } (\cdot)^m \text{ dla } m = \frac{1}{10} \quad r(\theta) \xrightarrow{(\cdot)^m} r(\theta)^{\frac{1}{10}} \\
&r(\theta) = |\sin(8\theta)|^{\frac{1}{10}} \quad \text{dodajemy przestrzeń na wewnętrzny okrąg } r(\theta) \xrightarrow{\vec{v}=[0,0.18]} r(\theta) + 0.18 \\
&r(\theta) = 0.18 + |\sin(8\theta)|^{\frac{1}{10}}
\end{aligned}
$$

Dodajemy okrąg o promieniu 0.18 $\left(x^2 + y^2 = (0.18)^2\right)$ i umieszczamy te równania w GeoGebrze.

```
1. f(t)=0.18 + (abs(cos(8t)))^(1/10)
2. Curve(f(t)cos(t),f(t)sin(t),t,0,2pi)
3. x^2+y^2=(0.18)^2
```

Tworzymy za pomocą GeoGebry wykres XY tej funkcji, a następnie dostosowujemy kolory i grubość linii, aby uzyskać efekt zbliżony do emblematu z japońskiego paszportu. (niewielkie różnice w kolorach wynikają z wysokiej ziarnistości zdjęcia referencyjnego). Przyjmując pewne tolerancje dla druku tak stworzony szablon jest bardzo zbliżony do oryginału.

Paszport Japonii



(a) Figura stworzona w GeoGebrze - `geogebra.ggb`

(b) Japoński Paszport - źródło Wikipedia

Środowisko `geogebra.ggb`. Obok sprawozdania zamieszczam również plik `geogebra.ggb` z projektem GeoGebry, który można załadować do programu, aby na żywo zobaczyć figurę widoczną poniżej po lewej stronie. Po prawej stronie znajduje się referencja - zdjęcie japońskiego paszportu.

Rozmaitość algebraiczna

Figurę można przedstawić jako rozmaitość algebraiczną. W tym celu sprowadźmy równanie polarne do postaci kartezjańskiej. Wykorzystajmy narzędzie WolframAlpha do przekształcenia $r(\theta) \to f(x,y)$ (pod spodem wykorzystał podaną w `Curve` parametryzację $x = r\cos\theta$, $y = r\sin\theta$):

```
wolfram>  r(t) = 0.18 + (abs(cos(8t))) pow (1/10) to cartesian
```

$$\sqrt{x^2+y^2} = 0.18 + \left|\cos\left(8\,\mathrm{tg}^{-1}\left(\frac{y}{x}\right)\right)\right|^{\frac{1}{10}}$$

Na tej podstawie tworzymy rozmaitość algebraiczną V w $\mathbb{R}^2$:

$$V\left(-\sqrt{x^2+y^2} + 0.18 + \left|\cos\left(8\,\mathrm{tg}^{-1}\left(\frac{y}{x}\right)\right)\right|^{\frac{1}{10}}\right) \text{ w } \mathbb{R}^2$$

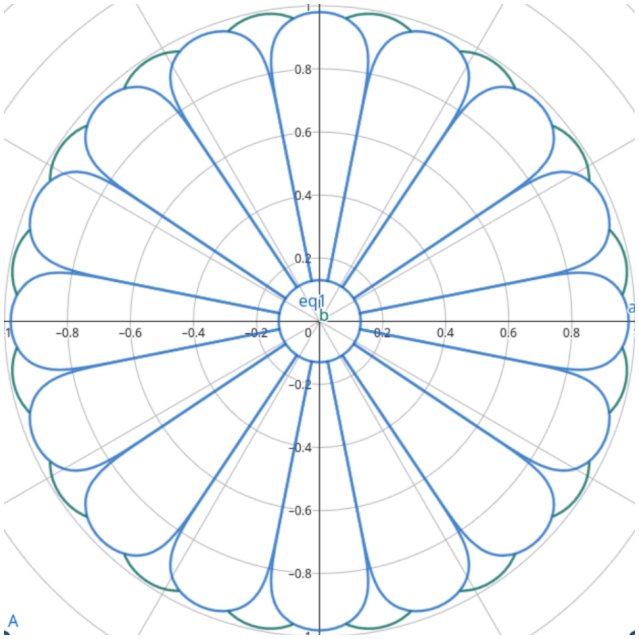
Figura z Japońskiego Godła

Przy pomocy bardzo podobnych operacji można dojść do Kamonu Cesarza Japonii. W tym przypadku zdecydowałem się pokazać rysunek konturowy. Wykorzystane polecenia:

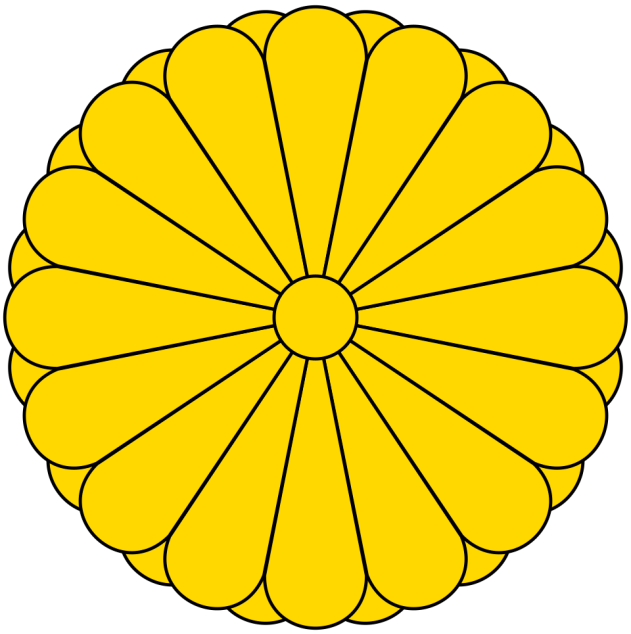
```
1. f(t) = 0.13 + 17/20 (abs(cos(8t)))^(1/13)
2. Curve(f(t)cos(t),f(t)sin(t),t,0,2pi)
3. g(t) = 0.74 + 1/4 (abs(cos(8t + pi/2)))^(1/4)                  // tu dochodzi do rotacji
4. Curve(g(t) cos(t)(g(t)>0.965),g(t) sin(t)(g(t)>0.965),t,0,2pi) // filtrowanie wartości parametrów
5. x^2+y^2=(0.13)^2
```

Po lewej stronie wykonany w GeoGebrze wykres XY z siatką polarną. Tym razem bez dostosowania kolorów, aby ukazać efekt konturowy, który pokrywa się z oryginałem. Po prawej stronie znajduje się referencja, z której odtworzyłem figurę.

Kamon Cesarza Japonii



(a) Figura stworzona w GeoGebrze - `geogebra2.ggb`



(b) Godło Japonii - źródło Wikipedia