1. Wprowadzenie teoretyczne

Z algorytmicznego punktu widzenia w klasycznym – uczonym w szkołach – podejściu do bilansowania równań reakcji chemicznych wykorzystywana jest metoda siłowa. Tak naprawdę sprawdzamy każde możliwe rozwiązanie, intuicyjnie odrzucając te niewłaściwe. Rozwiązanie takie zdaje egzamin przy zastosowaniu komputera, jednak tylko w przypadku reakcji, których ustalone współczynniki są odpowiednio małe. W poniższym przykładzie wyraźnie widać, dlaczego podejście to zawodzi:

$$10\,K_4Fe(CN)_6 + 122\,KMnO_4 + 299\,H_2SO_4 \rightarrow$$

$$162\,KHSO_4 + 5\,Fe_2(SO_4)_3 + 122\,MnSO_4 + 60\,HNO_3 + 60\,CO_2 + 188\,H_2O$$

Widzimy, że największy ze współczynników ma wartość 299. *Brute force* musiałby wykonać nieakceptowalnie dużą liczbę obliczeń, aby zbilansować takie równanie. Dlatego właśnie potrzebujemy algorytmicznego podejścia do problemu.

Okazuje się, że istnieją dobrze przebadane alternatywne metody, wykorzystujące algebrę liniową. Najlepszą zdaje się być opisana przez Lawrence R. Thorne w publikacji *"An Innovative Approach to Balancing Chemical-Reaction Equations: A Simplified Matrix-Inversion Technique for Determining The Matrix Null Space"* i to jej implementację stanowi ten program.
(zobacz https://arxiv.org/ftp/arxiv/papers/1110/1110.4321.pdf po więcej informacji)

Operacją determinującą złożoność tego algorytmu jest odwracanie macierzy – jego czas wykonania jest więc sześcienny $(O(n^3))$, jednak w praktyce $n<10$, dlatego program działa bardzo szybko.

2. Opis interfejsu

Dokumentacja wszystkich klas i ich metod znajduje się w folderze *docs*.

Program składa się z następujących modułów:

- *parsing* – moduł rozkładający równanie reakcji chemicznej na poszczególne cząsteczki, następnie te na atomy; w celu przeprowadzenia obliczeń algebraicznych potrzebujemy znać liczbę poszczególnych atomów w każdym z reagentów,
- *computing* – moduł zawierający kod przeprowadzający wszelkie obliczenia rozwiązujące problem, w szczególności tworzenie macierzy, obliczanie jej odwrotności, sprawdzanie poprawności rozwiązania,
- *balancing* – moduł łączący *parsing* oraz *computing*, udostępniajacy ostateczną metodę rozwiązującą problem,
- *chembal_logging* – moduł logujący komunikaty i błędy,
- *testing* – moduł do przeprowadzania manualnych testów,
- *client.py* – program kliencki

3. Wyniki testów

Za pomocą modułu *testing* przeprowadzone zostały testy na 32 równaniach. Program poprawnie zbilansował 28 reakcji – skuteczność wyniosła więc 87.5%. Warto zauważyć, że algorytm bezproblemowo radzi sobie z bardzo skomplikowanymi równaniami z chemicznego punktu widzenia, zaś problemy sprawiają trywialne przypadki. Wynika to ze specyfiki obliczeń algebraicznych – w “łatwych” reakcjach napotykamy problem macierzy osobliwej.

4. Podsumowanie

Program wykazał satysfakcjonującą skuteczność. Zwiększenie jej (być może nawet do 100%) jest możliwe – należałoby dokładniej przestudiować algebraicznie przypadki, z którymi algorytm sobie nie radzi. Alternatywnym rozwiązaniem byłoby identyfikowanie takich sytuacji i wykonywanie w nich metody siłowej, jako, że charakteryzują się one niskimi współczynnikami.