// Autor: Daniel Szarek
//====================

Zawartość folderu z zadaniem:
main.cpp – Kod z rozwiązaniem zadania napisanym w języku C++
funkcje.h – Biblioteka z własnymi funkcjami wykorzystywanymi do rozwiązania zadania N10
opracowanie.pdf – Ten dokument z opracowaniem zadania N10
funkcje.cpp – Plik zawierający implementację funkcji podanych w własnej bibliotece funkcje.h
Makefile – Do uruchomienia programu main.cpp (Makefile oferuje możliwość uruchomienia programu za pomocą komendy 'make run', usunięcia plików po uruchomieniu programu komendą 'make clean' oraz możliwość utworzenia paczki .tar.gz z zawartością foldera z zadaniem za pomocą komendy 'make tar'
wyniki.txt – Plik tekstowy z wynikami programu main.cpp

Do zrealizowania zadania wykorzystałem GSL - GNU Scientific Library, w Makefile zawarłem komendy do uruchomienia zadania na swojej maszynie w systemie Ubuntu, w celu uruchomienia zadania na swoim sprzęcie należy w prawidłowy sposób zainstalować bibliotekę GSL oraz zmienić scieżkę INCLUDEPATH1 dla swojej maszyny.

W swoich plikach źródłowych zamieszczam liczne komentarze, które na bieżąco tłumaczą działanie oraz funkcjonalności mojego kodu. W tym dokumencie przedstawię omówienie metod wykorzystanych do rozwiązania zadania.

Treść Zadania:

N10 *Zadanie numeryczne*
Znaleźć wszystkie rozwiązania równania $\det(A-\lambda\mathbb{1})$ trzema metodami (poszukiwania miejsc zerowych) z dokładnością 10^{-8}. Które metody działają najszybciej?

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}.$$

Rozwiązanie zadania:

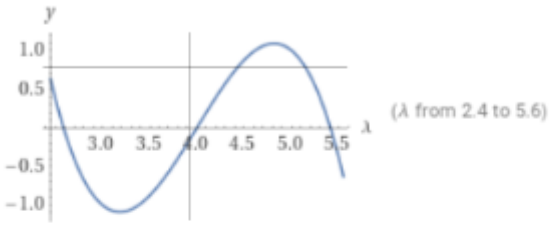
```
vboxuser@Ubuntu:~/Desktop/MetodyNumeryczne/Zadanie10$ make run
ar rsv libMojeFunkcje.a funkcje.o
ar: creating libMojeFunkcje.a
a - funkcje.o
mkdir -p ./lib
mv libMojeFunkcje.a ./lib
g++ -o main.x main.o -Wall -std=c++11 -L./lib -lMojeFunkcje -lgsl -lgslcblas
./main.x
Pierwiastki wielomianu uzyskane dzieki metodzie Newtona:
2.585786
Liczba iteracji: 7
4.000000
Liczba iteracji: 3
5.414214
Liczba iteracji: 4
Czas wykonywania metody: 0.000038133 sekund.
Pierwiastki wielomianu uzyskane dzieki metodzie Bisekcji:
2.585786
Liczba iteracji: 30
4.000000
Liczba iteracji: 29
5.414214
Liczba iteracji: 28
Czas wykonywania metody: 0.000041839 sekund.
Pierwiastki wielomianu uzyskane dzieki metodzie Siecznych:
2.585786
Liczba iteracji: 10
4.000000
Liczba iteracji: 5
5.414214
Liczba iteracji: 9
Czas wykonywania metody: 0.000016251 sekund.
vboxuser@Ubuntu:~/Desktop/MetodyNumeryczne/Zadanie10$
```

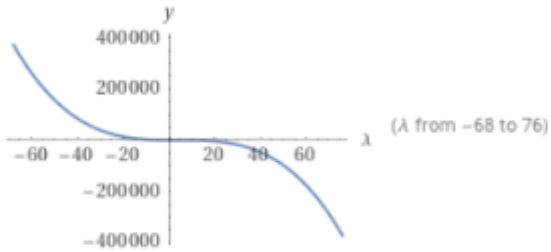
Powyżej znajdują się rozwiązania równania det(A – λI) trzema metodami: Metodą Newtona, Metodą Bisekcji oraz Metodą Siecznych. Równanie będzie sprowadzało się rozwiązania wielomianu trzeciego stopnia co pokażę poniżej. Miejsca zerowe tego wielomianu zostały w sposób prawidłowy obliczone i wskazane przez mój program co można zaobserwować porównując wyniki z poniższym zrzutem ekranu wyniku ukazanego przez stronę https://www.wolframalpha.com/ .

Input

$$-\lambda^3 + 12\lambda^2 - 46\lambda + 56$$

Plots





Alternate forms

$$-(\lambda - 4)\left(\lambda^2 - 8\lambda + 14\right)$$

$$-(\lambda - 4)((\lambda - 8)\lambda + 14)$$

$$\lambda((12 - \lambda)\lambda - 46) + 56$$

Roots

$$\lambda = 4$$

$$\lambda \approx 2.5858$$

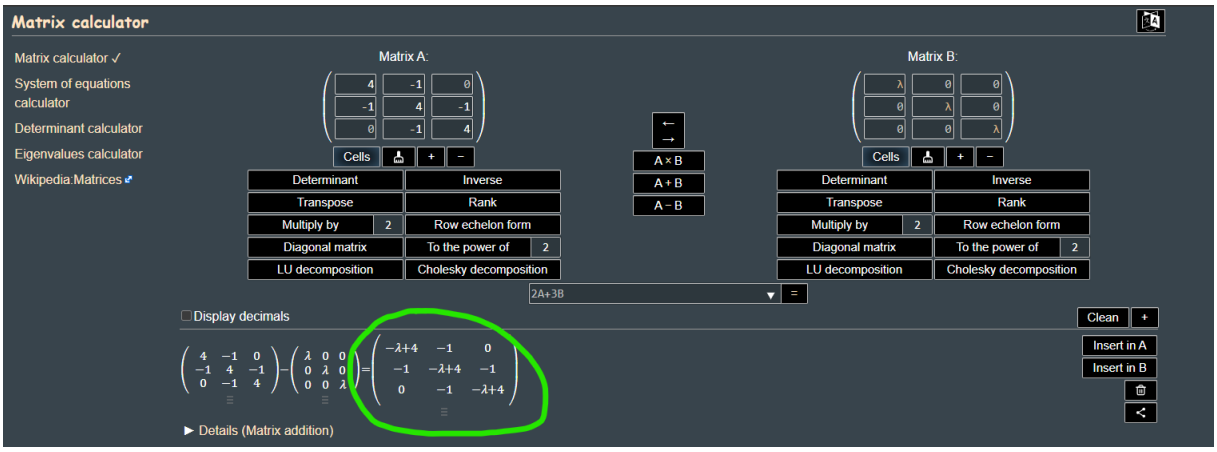
$$\lambda \approx 5.4142$$

Najszybszą metodą okazała się metoda Newtona, potrzebowała ona najmniej iteracji do każdego z pierwiastków, niemniej jednak ogólnie mniej czasu na wykonanie zadania zajęło metodzie siecznych. Wynika to z faktu, że w metodzie Newtona musimy liczyć pochodne funkcji co może być czasochłonne, a w metodzie siecznych nie korzystamy z pochodnych tylko z dwóch początkowych przybliżeń. Ostatecznie wynik szybkości metody będzie sprowadzał się do każdego indywidualnego przypadku czasami liczenie pochodnych będzie szybsze, a czasami wybór i wykorzystanie dwóch początkowych przybliżeń.

Omówienie:

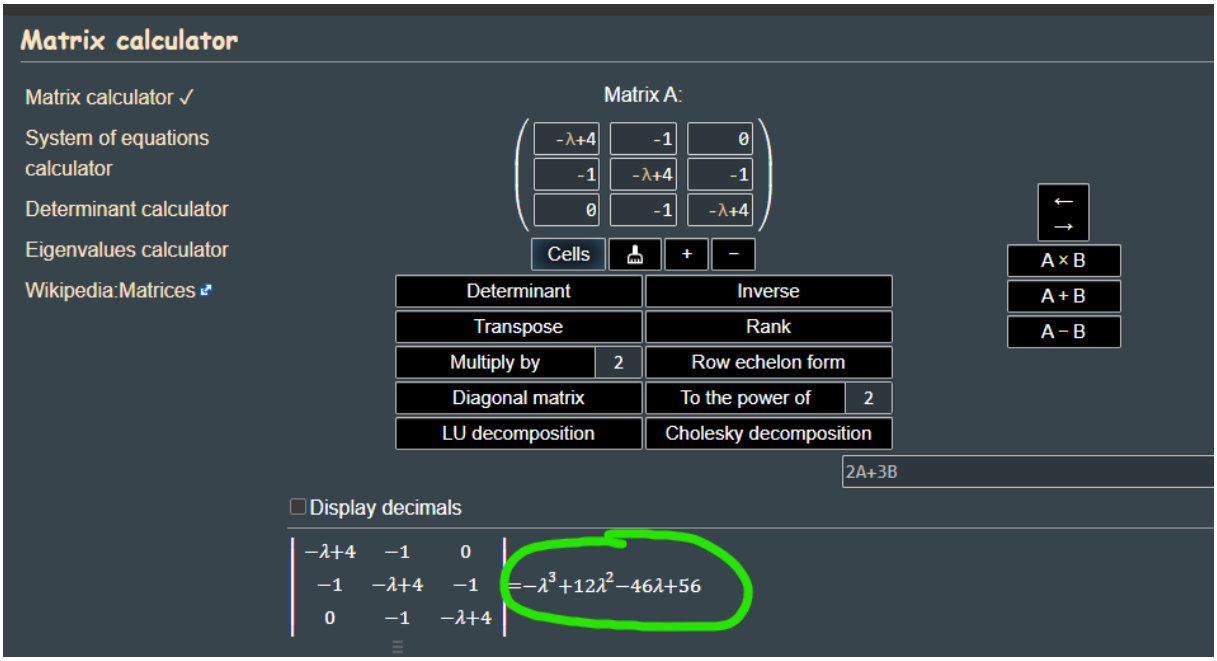
W celu wizualizacji problem skorzystam ze strony https://matrixcalc.org/

Musimy obliczyć równanie det(A – λI), stąd w pierwszej kolejności obliczymy różnicę macierzy A z iloczynem skalarnym z macierzom jednostkową.



Wynik będzie reprezentowany przez powyższą macierz wskazaną na zielono.

Następnie policzmy wyznacznik macierzy wskazanej na zielono powyżej.



Wynikiem jest wielomian: -λ^3 + 12λ^2 - 46λ + 56.

Naszym zadaniem będzie obliczenie miejsc zerowych tego wielomianu.

Tak jak podałem powyżej realizuje to na trzy sposoby, poniżej wkleję zdjęcia algorytmów na których bazowałem swoją implementację:

Metoda Newtona-Raphsona, znana również jako metoda Newtona:

Wzór rekurencyjny [edytuj | edytuj kod]

Dowodzi się, że kolejne przybliżenia szukanego miejsca zerowego dane są wzorem rekurencyjnym:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, k = 1, 2, \ldots.$$

Warunek zakończenia obliczeń [edytuj | edytuj kod]

Stosowanych jest kilka kryteriów zakończenia obliczeń (ϵ to przyjęta dokładność obliczeń):

1. wartość funkcji w wyznaczonym punkcie jest bliska 0:
   $|f(x_k)| \leqslant \epsilon$
2. odległość pomiędzy kolejnymi przybliżeniami jest dość mała:
   $|x_{k+1} - x_k| \leqslant \epsilon$

Metoda ta polega na iteracyjnym zbliżaniu się do miejsca zerowego funkcji poprzez aktualizację przybliżeń za pomocą pochodnej i wartości funkcji w bieżącym punkcie. Metoda Newtona jest efektywna, gdy początkowe przybliżenie jest bliskie rzeczywistego miejsca zerowego. Jednak może być wrażliwa na wybór początkowego przybliżenia i może prowadzić do niestabilności lub braku zbieżności w pewnych sytuacjach. W praktyce jest szeroko stosowana do znajdowania miejsc zerowych funkcji różniczkowalnych, zwłaszcza gdy pochodne są łatwo obliczalne.

Metoda Bisekcji:

Przebieg algorytmu [edytuj | edytuj kod]

1. Sprawdzenie, czy pierwiastkiem równania jest punkt $x_1 = \frac{a+b}{2}$, czyli czy $f(x_1) = 0$.
   Jeżeli tak jest, algorytm kończy działanie, a punkt x_1 jest szukanym miejscem zerowym.
2. W przeciwnym razie, dopóki nie osiągniemy żądanej dokładności, czyli dopóki $|a - b| > \epsilon$:
   1. Zgodnie ze wzorem z punktu pierwszego ponownie wyznaczane jest x_1, dzieląc przedział $[a, b]$ na dwa mniejsze przedziały: $[a, x_1]$ i $[x_1, b]$.
   2. Wybierany jest koniec przedziału, którego wartość funkcji posiada znak przeciwny do $f(x_1)$ i odpowiednio górny albo dolny kraniec przedziału (b albo a) przyjmuje wartość x_1, tj.
      1. Jeżeli $f(a)f(x_1) < 0$, to $b = x_1$,
      2. Jeżeli $f(x_1)f(b) < 0$, to $a = x_1$.
3. Po osiągnięciu żądanej dokładności algorytm kończy działanie, a szukany pierwiastek równania wynosi $\frac{a+b}{2}$.

Metoda ta rozpoczyna się od wybrania przedziału, w którym funkcja zmienia znak. Następnie iteracyjnie dzieli się przedział na pół, sprawdzając znak w środkowym punkcie. Proces powtarza się, skracając przedział, aż osiągnie się zadowalającą dokładność lub wartość funkcji zbliży się do zera. Metoda ta jest prostą i stabilną techniką, choć może być wolniejsza dla funkcji o szybko zmieniających się znakach. Jej efektywność zależy od precyzyjnego początkowego wyboru przedziału.

Metoda Siecznych:

Opis procedury [edytuj | edytuj kod]

Wersja podstawowa [edytuj | edytuj kod]

Polega ona na przyjęciu, że funkcja ciągła na dostatecznie małym odcinku w przybliżeniu zmienia się w sposób liniowy. Możemy wtedy na odcinku $\langle a, b \rangle$ krzywą $y = f(x)$ zastąpić sieczną. Za przybliżoną wartość pierwiastka przyjmujemy punkt przecięcia siecznej z osią OX.

Metodę siecznych dla funkcji $f(x)$, mającej pierwiastek w przedziale $\langle a, b \rangle$ można zapisać następującym wzorem rekurencyjnym:

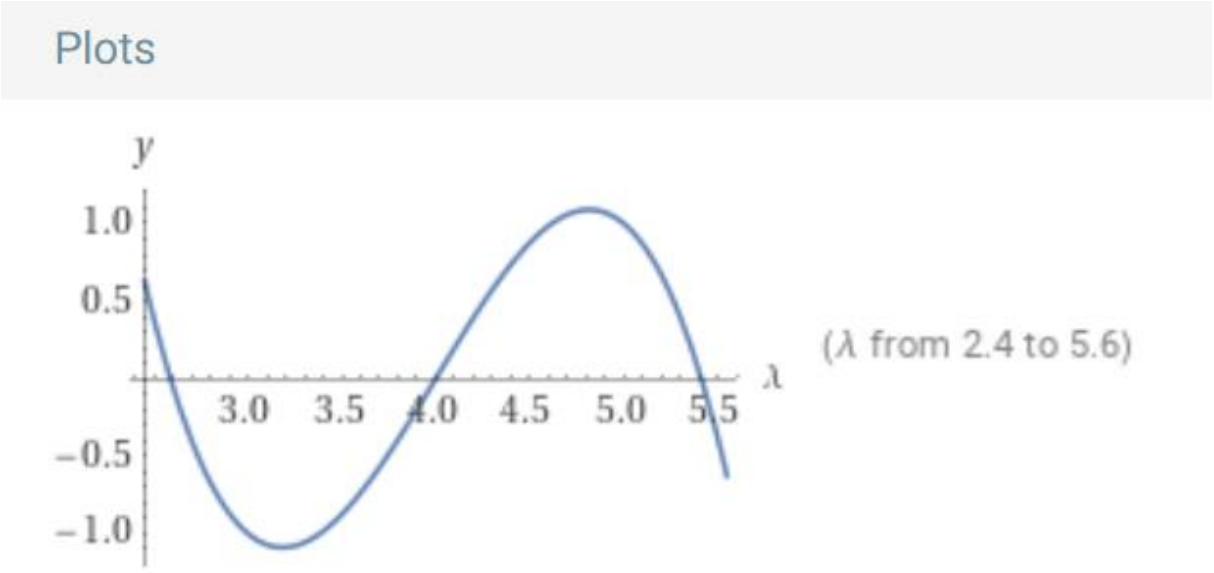
$$\begin{cases} x_0 = a \\ x_1 = b \\ x_{n+1} = \frac{f(x_n)x_{n-1} - f(x_{n-1})x_n}{f(x_n) - f(x_{n-1})} \end{cases}$$

Aby metoda się powiodła, dla każdego n musi zachodzić $f(x_n)f(x_{n-1}) < 0$, gdyż tylko wtedy sieczna przechodząca przez punkty $(x_n, f(x_n))$ i $(x_{n-1}, f(x_{n-1}))$ przecina oś OX. Metoda ta nie zawsze jest zbieżna.

Przykład naiwnego zastosowania metody siecznych. Pierwsza iteracja – zwracająca punkt x_2 – przybliża do miejsca zerowego, jednak następna – zwracająca punkt x_3 – od niego oddala. To dlatego, że dla punktów x_1 i x_2 wartości funkcji mają ten sam znak, co nie gwarantuje miejsca zerowego między nimi.

Metoda ta zaczyna od wyboru dwóch punktów na wykresie funkcji, co wyznacza początkowy przedział poszukiwań. Następnie, tworzona jest linia sieczna, przecinająca wykres i tworząca przybliżenie miejsca zerowego jako punkt przecięcia z osią x. Kolejne przybliżenia są obliczane, korzystając z poprzednich punktów i wartości funkcji w tych miejscach, a proces ten trwa, aż uzyskamy odpowiednią dokładność lub wartość funkcji zbliży się do zera. Tak jak podałem powyżej będzie ona najszybsza czasowo, niemniej jednak nie zawsze będzie ona gwarantować zbieżność.

W ramach implementacji metody bisekcji oraz metody siecznych wykorzystałem przedziały poszukiwań, przedziały te dobrałem na podstawie poniższego wykresu. W tych przedziałach można odczytać gdzie mniej więcej funkcja zmienia znak i znajdują się miejsca zerowe naszego wielomianu.



Na potrzebę realizacji metod w praktyce utworzyłem w pliku funkcje.h klasę Wielomian oraz strukturę Punkt, oba są wykorzystywane do obliczenia miejsca zerowego wyznacznika podanego w poleceniu.