Modele matematyczne w planowaniu i analizie eksperymentu – Projekt

Symulator balistyczny

Adam Basinski 246081

1. Cel.

Celem projektu było przygotowanie kalkulatora artyleryjskiego zdolnego do obliczania trajektorii pocisku z uwzględnieniem rodzaju pocisku (jego oporu) i aktualnych warunków meteorologicznych, takich jak temperatura, ciśnienie, kierunek wiatru.

2. Wstęp i rozwiązania matematyczne.

Balistyka jest nauką o miotaniu i ruchu pocisków (współcześnie również rakiet), której głównym celem jest opracowanie tablic balistycznych, opierających się niejednokrotnie na danych eksperymentalnych. W projekcie sprawdzany jest model znacznie uproszczony, opierający się na równaniach kinetycznych dla cząstki z uwzględnieniem oporu Newtonowskiego.

Równania kinetyczne dla cząstki:

$$\vec{a} = \frac{d^2\vec{r}}{dt^2}$$

$$\vec{a} = \frac{d\vec{v}}{dt}$$

Gdzie r – współrzędna przestrzenna, v – prędkość chwilowa, a – przyśpieszenie chwilowe, t – czas.

Do obliczenia pozycji będzie wykorzystywany wielomian rozłożony dla każdej z 3 współrzędnych:

$$\vec{r} = \vec{r}_o + \vec{v}t + \vec{a}\frac{t^2}{2}$$

Opór Newtonowski:

$$\overrightarrow{C_d} = \frac{\overrightarrow{F_d}}{\frac{1}{2}\rho \cdot \vec{v}^2 \cdot A}$$

Gdzie C_d – współczynnik oporu, charakterystyczny dla kształtu, wyznaczany eksperymentalnie, atrybut obiektu *Projectile.dragCoefficient,* F_d – siła przyłożona na pocisk, ρ – gęstość płynu (tutaj powietrza), v – prędkość chwilowa pocisku, A – powierzchnia czołowa pocisku, atrybut obiektu *Projectile.frontalSurface.*

Gęstość powietrza jest wyliczana na podstawie danych meteorologicznych uwzględniających ciśnienie, temperaturę, wilgotność i aktualnej wysokości pocisku. Wykorzystywane są do tego wzory zaczerpnięte ze strony internetowej OmniCalculator:

https://www.omnicalculator.com/physics/air-density oraz

https://www.omnicalculator.com/physics/air-pressure-at-altitude (odwiedzone ostatni raz 02.01.2023).

Obliczenia są wykonywane co pewien stały krok czasu, atrybut obiektu *Projectile.timeResolution*.

Kierunki odpowiadające wektorom – obliczenia prowadzone są w układzie współrzędnych kartezjańskich, w których:

- oś X określa kierunek północ – południe o zwrocie na północ;
- oś Y określa kierunek wschód – zachód o zwrocie na wschód;
- oś Z jest prostopadła do powierzchni ziemi o zwrocie przeciwnym do powierzchni ziemi.

Kierunek rotacji oraz kierunek wiatru są określane od 0 – 360 stopni, gdzie 0 stopni określa północ i rośnie wraz z obrotem zgodnie z kierunkiem ruchu wskazówek zegara.

Kąt elewacji 0 – 90 stopni, gdzie 0 oznacza ruch prostopadły do powierzchni, a 90 stopni strzał pionowo w górę.

3. Dokumentacja.

Używane biblioteki: math, matplotlib, pandas, json, requests, datetime, time.

Zawartość:

Plik zawierający bibliotekę obiektów wykonujących obliczenia: ThrowingObject.py

Plik zawierający klucz API wymagany do pobrania danych meteorologicznych (wymagany): Api_key.py

Plik zawierający dane meteorologiczne (nie wymagany, zostanie utworzony w razie potrzeby): TodaysWeather.py

Pliki zawierające przykłady, Example(1-4).py

ThrowingObject.Projectile

Obiekt służący za rdzeń obliczeń, pozostałe obiekty dotyczą doprecyzowanych sytuacji i dziedziczą atrybuty i funkcje po tym obiekcie.

#Parametry inicjalizacji, określające cechy miotanego pocisku:

- *projectile_mass* – masa obiektu [kg]
- *drag* – współczynnik oporu powietrza (Newtonowski) [-]
- *frontal_surface* – powierzchnia czołowa [m2]. Dla uproszczenia w trakcie obliczeń, pocisk jest traktowany jak kula o takiej samej powierzchni czołowej z każdej strony.
- *velocity* – prędkość wylotowa [m/s]. Podawana w formie skalarnej, który zostaje przeliczony na wektor początkowy prędkości na podstawie kąta rotacji i elewacji.
- *rotation_angle* –rotacja „działa” określająca kierunek początkowy lotu [°].
- *elevation_angle* – elewacja „działa” [°].
- *height* – wysokość początkowa [m]
- *ifWind* – wartość *True/False* określająca czy uwzględniany jest wiatr w obliczeniach, wyjściowo o wartości *True*.
- *time_resolution* – rozdzielczość czasowa obliczeń [s], wyjściowo o wartości 0.01s.

#Funkcje inicjalizujące:

- *check_angles* – sprawdza, czy zostały podane prawidłowe wartości kątów rotacji i elewacji, przelicza je na radiany rozumiane przez funkcje trygonometryczne biblioteki *math*.
- *set_velocity_vector* – przeliczna skalar prędkości wylotowej na wektor użyteczny w dalszych obliczeniach.
- *Weather_execute* – funkcja wykonująca odpowiednio pozostałe funkcje wiązane z parametrami meteorologicznymi.
- *if_get_todays_weather* – określa czy wymagane jest pobranie pogody na podstawie czy była już dziś pobierana lub z powodu wcześniejszego utworzenia pustego pliku.

Pogoda pobierana jest raz na dzień ze względu na ograniczoną ilość pobrań darmowej wersji API, wymuszenie pobrania możliwe jest poprzez usunięcie pliku *TodaysWeather.py* przed inicjalizacją lub zmianę daty w tym samym pliku.

- *get_weather* – pobiera dane meteorologiczne dla Wrocławia (współrzędne - lat: 51.107883, lon: 17.038538). Współrzędne wpisane są w kod źródłowy.
- *set_weather_params* – informuje, czy dane meteorologiczne zostały pobrane i odświeża je, gdy jest to wymagane. Uwzględnia również siłę wiatru, jeżeli jest to wymagane.
- *save_weather_params* – zapisuje pobrane dane meteorologiczne do pliku pliku *TodaysWeather.py*.

#Funkcje *calc_*:

Wywoływane przez funkcje *trajectory_calculation* mają za zadanie wyliczyć i zapisać w atrybutach obiektu nowy stan pocisku.

#Funkcja pomocnicza *Append_flight_data* używana przez funkcje *trajectory_calculation*. Ma za zadanie rozszerzać tabelę z danymi o locie po wykonaniu kolejnego kroku.

#Główna funkcja obliczeniowa oraz funkcje result_:

Z założenia mają być wywoływane przez użytkownika w celu symulacji lot i wyświetlenia wyników w wybrany sposób.

- *trajectory_calculation* – główna funkcja obliczeniowa, tworząca jako wynik obiekt DataFrame z biblioteki *Pandas* wywoływany poprzez *Projectile.flight_Data*. Funkcja oblicza kolejne kroki lotu, a po uderzeniu w powierzchnie (współrzędna $Z < 0$) znajduje możliwie dokładną pozycje ostateczną o krok czasu mniejszy niż zadany.

Obliczenia prowadzone są w kolejności:

1. Nowa pozycja pocisku na podstawie poprzednich wektorów prędkości i przyśpieszenia;
2. Nowy wektor prędkości na podstawie poprzedniego wektora przyśpieszenia;
3. Nowy wektor przyśpieszenia na podstawie nowego wektora prędkości z uwzględnieniem prędkości wiatru i poprzedniego oporu powietrza;
4. Nowy oporu powietrza na podstawie nowego wektora prędkości i nowej wysokości z uwzględnieniem prędkości wiatru.

- *result_graph_trajectory* – metoda wyświetlająca w formie 3 wykresów zależność wysokości przelotu od dystansu poziomego, trajektorie poziomą oraz zależność wysokości przelotu od czas.
- *result_graph_trajectory_3D* – podobnie do poprzedniej metody, jednak w tym przypadku trajektoria widoczna jest w jako rzut przestrzeni trójwymiarowej.
- *result_graph_drag_force* – wartość oporu powietrza na pocisk w poziomie w zależności od jego prędkości.

- *result_graph_velocity* – wykres pokazujący wartość prędkości pocisku w zależności od czasu lotu.
- *result_table* – wyświetla najważniejsze dane w formie wydruku w konsoli.

Pozostałe obiekty dziedziczące funkcje ThrowingObject.Projectile

Wybrane obiekty ułatwiają symulacje, ponieważ mają zaimplementowane parametry takie jak prędkość wylotowa, waga pocisku, jego powierzchnia i opór powietrza. Wybrane możliwie zróżnicowane dla zobrazowania wpływu parametrów na lot.

- Glock_handgun – prosty popularny ręczny pistolet o skutecznym zasięgu około 50 m.
- Karl_Gerat_mortar – duży (kaliber 60 cm) niemiecki moździerz używany podczas 2 wojny światowej.
- Sniper_rifle_M2010 – karabin precyzyjny dalekiego zasięgu.
- Ancient_cannon – klasyczne działo czarnoprochowe, używane na statkach w XVIII – XIX wieku.

4. Przykłady.

Przykład 1 – Wyliczenia dla trzech pierwszych obiektów dziedziczących.

Wydruk z dnia 03.01.2023

```
TodaysWeather file is missing, creating new one.
Creating new weather file done.
---Calculation for Handgun---
Initializing...
Setting weather params...
Weather will be downloaded.
New weather data downloaded.
Saving todays weather...
Saving done.
Checking angles...
Setting velocity vector...
Setting acceleration vector...
Initializing finished.
---Calculation for Karl Gerat mortar---
Initializing...
Setting weather params...
Weather will be downloaded.
New weather data downloaded.
Saving todays weather...
Saving done.
Checking angles...
Setting velocity vector...
Setting acceleration vector...
Initializing finished.
---Calculation for Sniper rifle M2010---
Initializing...
Setting weather params...
Weather will be downloaded.
New weather data downloaded.
Saving todays weather...
Saving done.
```

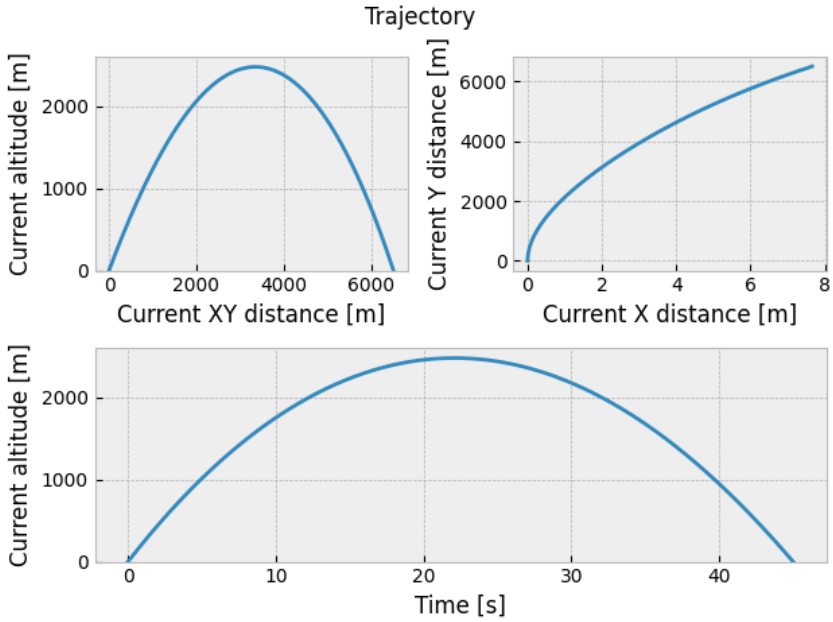
Checking angles...
Setting velocity vector...
Setting acceleration vector...
Initializing finished.
------Handgun------
 ---Final results---
 Impact speed: 333.93 m/s.
 With projectile mass equal to: 0.00804 kg, momentum is equal to: 2.685 kg*m/s.
 Total distance flown: 208.69 m.
 Maximum height: 1.8 m.
 Time of flight: 0.611 s.
------Mortar------
 ---Final results---
 Impact speed: 250.274 m/s.
 With projectile mass equal to: 1700 kg, momentum is equal to: 425465.562 kg*m/s.
 Total distance flown: 6038.95 m.
 Maximum height: 2778.801 m.
 Time of flight: 47.583 s.
------Rife------
 ---Final results---
 Impact speed: 806.901 m/s.
 With projectile mass equal to: 0.01166 kg, momentum is equal to: 9.408 kg*m/s.
 Total distance flown: 513.34 m.
 Maximum height: 1.8 m.
 Time of flight: 0.613 s.
Press any key to exit

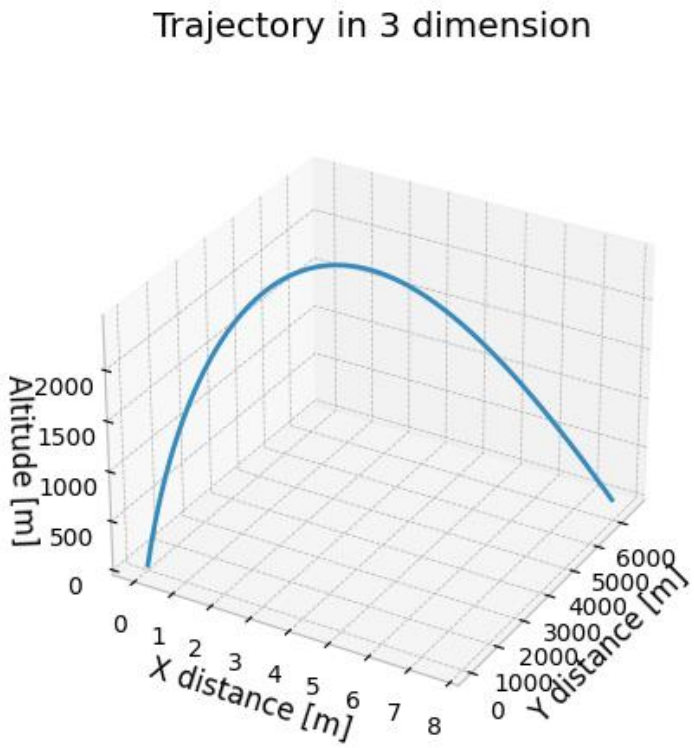
Wyniki pozwalają porównać zasięg broni. Najlepszy wynik uzyskujemy dla moździerza o zasięgu (6039 m) bardzo podobnym do wartości historycznych (6440 m) przy kącie elewacji wynoszącym 60 stopni, czyli nie najefektywniejszym, jeżeli chodzi o zasięg (najefektywniejsze jest 45 stopni elewacji). Pistolet ma zasięg powyżej skutecznego zasięgu, jednak według symulacji, pocisk karabinu uderzy w ziemie w mniej niż połowie rzeczywistego skutecznego zasięgu (1370 m). Źródła parametrów są zawarte w głównym pliku.

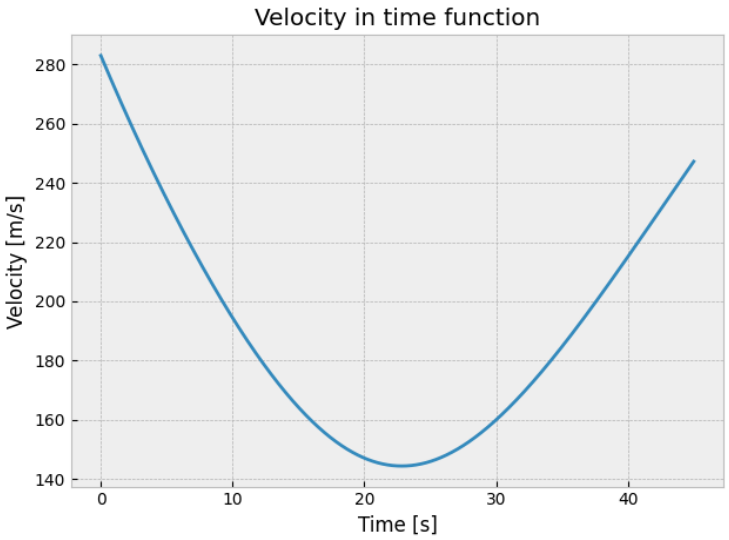
Przykład 2 – prezentacja możliwości funkcji *result_* .

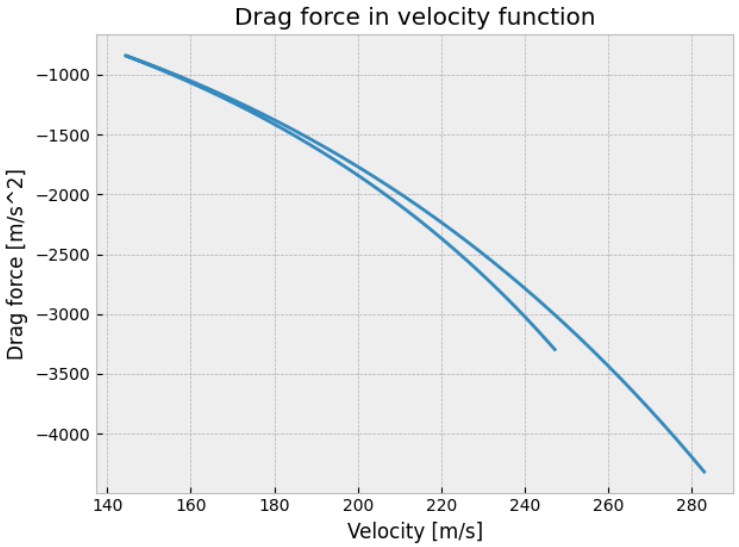
Wydruk z dnia 3.01.2023

---Calculation for Karl Gerat mortar---
Initializing...
Setting weather params...
Using old data from: 3 th day of 1 month.
Checking angles...
Setting velocity vector...
Setting acceleration vector...
Initializing finished.


Trajectory
Current altitude [m]
Current XY distance [m]
Current Y distance [m]
Current X distance [m]
Current altitude [m]
Time [s]


Trajectory in 3 dimension
Altitude [m]
X distance [m]
Y distance [m]





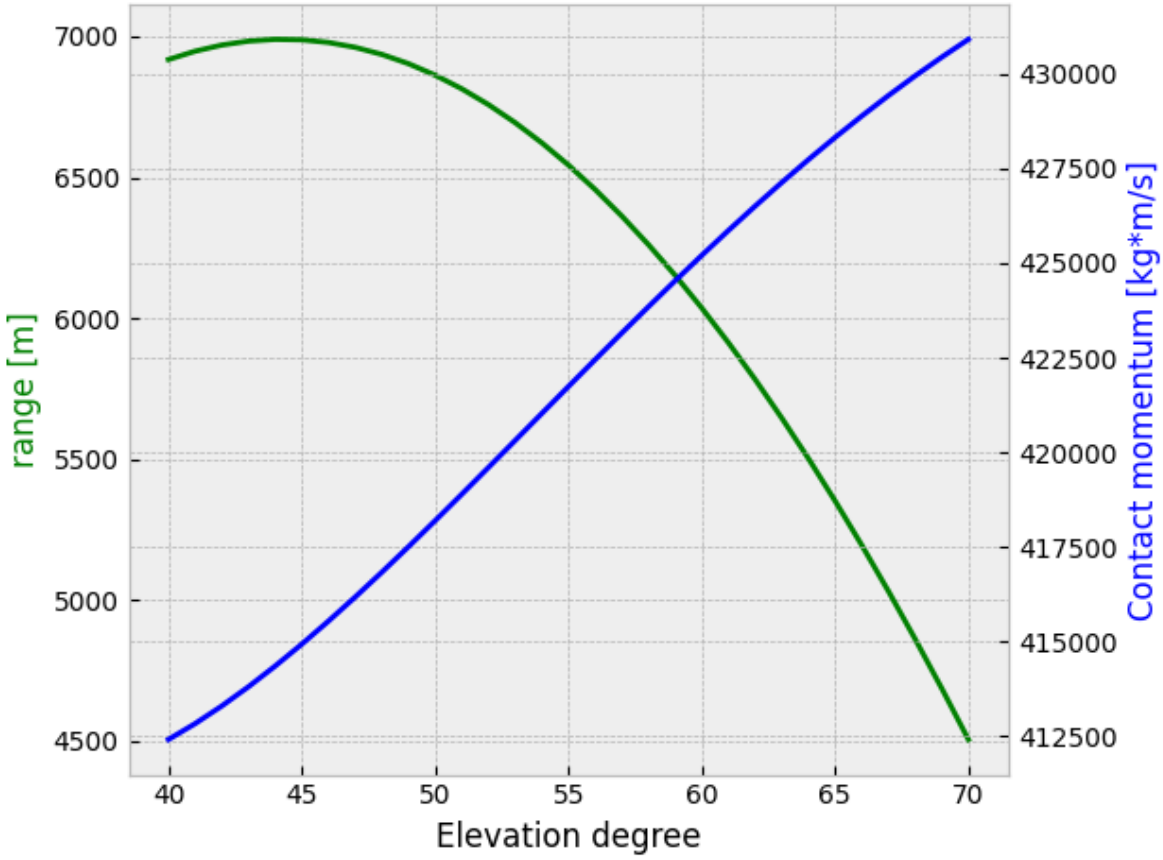
---Final results---
Impact speed: 247.218 m/s.
With projectile mass equal to: 1700 kg, momentum is equal to: 420270.396 kg*m/s.
Total distance flown: 6509.19 m.
Maximum height: 2482.881 m.
Time of flight: 44.993 s.
Press any key to exit

Zastosowanie kolejno 5 funkcji drukujących wyniki (*result_graph_trajectory*, *result_graph_trajectory_3D*, *result_graph_velocity*, *result_graph_drag_force*, *result_table*)

Przykład 3 – obliczenia optymalnego kąta elewacji dla moździerza. (Czas obliczeń wynosi około 5 minut.)

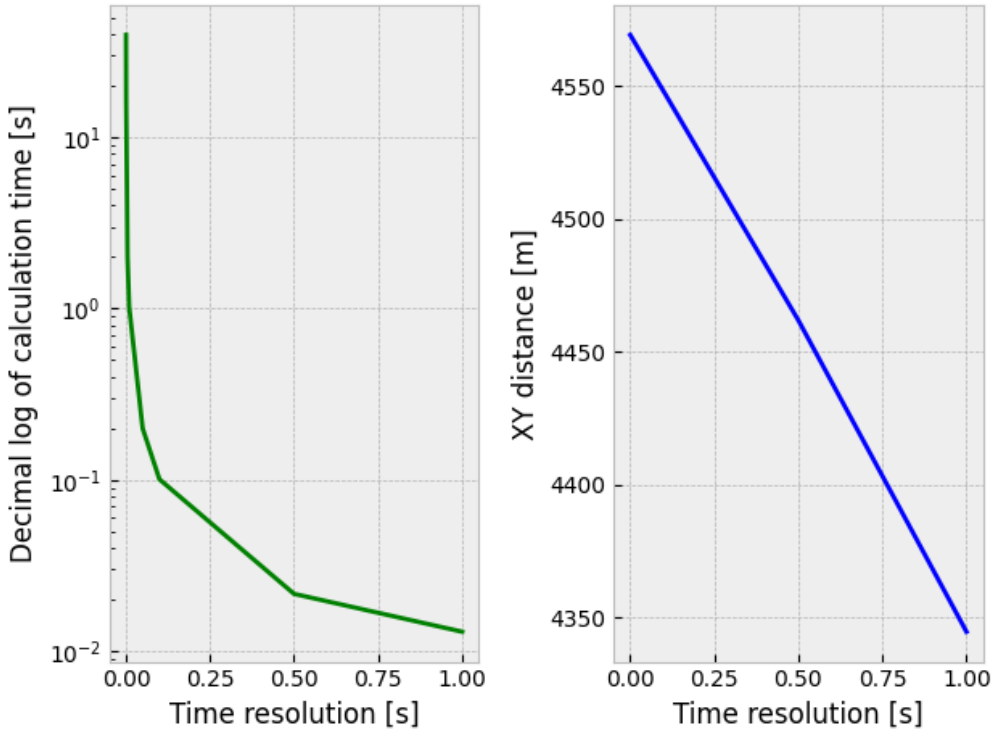
Oblicza całość dla kąta elewacji od 40 do 70 stopni drukuje wykres zależności zasięgu (zielony) i energii uderzenia (niebieski) od kąta elewacji.

Wydruk z dnia 3.01.2023



Wydruk pokazuje, że najefektywniejszym kątem elewacji pod względem zasięgu jest kąt 45 stopni. Siła uderzenia na podstawie pędu pocisku w momencie uderzenia rośnie wraz ze wzrostem kąta elewacji co oznacza, że zależy on głównie od wysokości maksymalnej osiągniętej przez pocisk.

Przykład 4 – Sprawdzenie wpływu rozdzielczości czasu na długość obliczeń i dokładność strzału. (Obliczenia zajmują około 5 minut.)



Resolution 1 s, takes 0.01177 seconds and execute 15 steps.
Resolution 0.5 s, takes 0.01996 seconds and execute 30 steps.
Resolution 0.1 s, takes 0.09577 seconds and execute 145 steps.
Resolution 0.05 s, takes 0.18619 seconds and execute 290 steps.
Resolution 0.01 s, takes 0.95621 seconds and execute 1444 steps.
Resolution 0.005 s, takes 1.94987 seconds and execute 2887 steps.
Resolution 0.001 s, takes 11.99853 seconds and execute 14432 steps.
Resolution 0.0005 s, takes 33.34863 seconds and execute 28864 steps.
Press any key to exit

Wyniki pokazują, że każde zmniejszenie czasu obliczeń zwiększa liniowo dokładność symulacji i tym samym zmniejsza błąd, jednak powoduje również znaczne wydłużenie czasu obliczeń. Wzrost dokładności obliczeń wynika z założenia, że liniowa zależność dąży do nieskończenie małej rozdzielczości obliczeń, w której otrzymujemy idealnie dokładny wynik.

5. Wnioski.

Symulator obarczony jest sporą ilością błędów wynikających ze sposobu liczenia uwzględniającego wiele uproszczeń. Najlepsze wyniki uzyskiwane są dla prostych pocisków i moździerzy. Uproszczona aerodynamika pocisków, sprowadzona do stałej wartości współczynnika oporu powietrza nie pozwala uzyskać wiarygodnych wyników dla obiektów typu pocisk karabinu czy pistoletu, które z założenia mają lecieć po linii prostej. Wynika to z niemożności uwzględnienia większej ilości efektów działających na pocisk, takich jak jego rotacja czy stabilizacja toru lotu wynikająca z kształtu pocisku.

Wzrost precyzji obliczeń związany jest w omawianym przypadku w liniowy sposób z rozdzielczością czasu obliczeń. Jako domyślną rozdzielczość czasową obliczeń wybrano 0.01 sekundy oferujący wystarczającą dokładność i krótki czas obliczeń, dla wyjściowych metod wykorzystania procesora przez kod.