

Michał Gawryluk  
**SPADOCHRONIARZ.C**

Wydział Fizyki, Astronomii i Informatyki Stosowanej  
Uniwersytet Jagielloński

# 1. KOD ŹRÓDŁOWY PROGRAMU

Poniższy kod źródłowy napisano w języku ANSI C.

```
/******  
* *  
* Michal Gawryluk 2008 *  
* *  
*****/  
#include<stdio.h>  
  
/*STALE *****/  
  
#define PLIK_WYJSCIOWY "spadochroniarz.out" /*nazwa pliku wyjsciowego*/  
#define P_FW_ERROR 1 /*kodu bledu przy nieudanej probie otwarcia pliku  
do zapisu*/  
/**/  
  
#define G 9.81 /*stala grawitacyjna*/  
#define H 1000. /*wysokosc poczatkowa skoczka*/  
#define V 0. /*predkosc poczatkowa*/  
  
#define MOMENT_POCZATKOWY 0. /*moment "zerowy" ruchu*/  
#define CZAS_OTWARCIA 3. /*czas [s] rozkladania spadochronu*/  
#define CZAS_DO_OTWARCIA 5.8 /*czas [s] po którym skoczek otworzy spadochron*/  
#define CZAS_MAX 200. /*maksymalny zalozony czas [s] trwania skoku*/  
#define LICZBA_PRZEDZIALOW 1000000. /*na ile przedzialow czasowych  
dzielimy*/  
  
#define KROK_CZASOWY (CZAS_MAX/LICZBA_PRZEDZIALOW)  
  
#define OPOR1 0.004 /* wartosc oporu przed rozlozeniem spadochronu */  
#define OPOR2 0.4 /* wartosc oporu po rozlozeniu spadochronu */  
  
/*PROTOTYPY FUNKCJI*****/  
  
double opor(double t); /*sila oporu w zaleznosci od czasu*/  
  
double f1(double czas, double wysokosc, double szybkość); /* */  
double f2(double czas, double wysokosc, double szybkość); /* */
```

```

/* M A I N *****/
int main()
{
/* DEKLARACJE ZMIENNYCH LOKALNYCH *****/
double k1, k2, k3, k4;
double q1, q2, q3, q4;

double h_mamy=H; /*wysokosc dla kroku N; inicjalizacja stala H*/
double v_mamy=V; /*szybkosc dla kroku N; inicjalizacja stala V*/

double h_obliczamy; /*wysokosc dla kroku N+1*/
double v_obliczamy; /*szybkosc dla kroku N+1*/

double moment=MOMENT_POCZATKOWY; /*informuja ile czasu uplynelo*/

FILE *plik_wyjsciowy; /*wskaznik na plik do ktorego zapisujemy wyniki*/
/*****/

/*otwieramy plik do zapisu*/
if((plik_wyjsciowy=fopen(PLIK_WYJSCIOWY,"w"))==NULL)
{
/*wykonywane w razie niepowodzenia przy otwarciu pliku do zapisu*/
printf("!BLAD! Nie mozna otworzyc pliku "PLIK_WYJSCIOWY" do zapisu!\n");
return P_FW_ERROR; /*zwroc kod bledu*/
}
else /*jesli sie udalo - wykonuje obliczenia*/
{
while((moment<CZAS_MAX)&&(h_mamy>0.0))
{
fprintf(plik_wyjsciowy,"%f ",moment);
fprintf(plik_wyjsciowy,"%f ",h_mamy);
fprintf(plik_wyjsciowy,"%f ",v_mamy);
fprintf(plik_wyjsciowy,"%f\n",f2(moment,h_mamy,v_mamy) );

k1= KROK_CZASOWY*f1(moment,h_mamy,v_mamy);
q1= KROK_CZASOWY*f2(moment,h_mamy,v_mamy);

k2= KROK_CZASOWY*f1((moment+KROK_CZASOWY/2), h_mamy+(k1/2), v_mamy+(q1/2));
q2= KROK_CZASOWY*f2(moment+(KROK_CZASOWY/2), h_mamy+(k1/2), v_mamy+(q1/2));

k3= KROK_CZASOWY*f1(moment+(KROK_CZASOWY/2), h_mamy+(k2/2), v_mamy+(q2/2));
q3= KROK_CZASOWY*f2(moment+(KROK_CZASOWY/2), h_mamy+(k2/2), v_mamy+(q2/2));

```

```

k4= KROK_CZASOWY*f1(moment+KROK_CZASOWY, h_mamy+k3, v_mamy+q3);
q4= KROK_CZASOWY*f2(moment+KROK_CZASOWY, h_mamy+k3, v_mamy+q3);

h_obliczamy=h_mamy+(k1/6.0)+(k2/3.0)+(k3/3.0)+(k4/6.0);
v_obliczamy=v_mamy+(q1/6.0)+(q2/3.0)+(q3/3.0)+(q4/6.0);

/*przygotowujemy dane na nastepny krok*/
h_mamy=h_obliczamy;
v_mamy=v_obliczamy;
moment+=KROK_CZASOWY;
};

/*zamykamy plik*/
fclose(plik_wyjsciowy);
printf("Dane wyjsciowe zapisane sa w pliku "PLIK_WYJSCIOWY"\n");
}
return 0;
}

/*DEKLARACJE FUNKCJI*****/

double opor(double t)
{
if (t<=CZAS_DO_OTWARCIA)
{
return OPOR1; /*najmniejszy opor - przed rozlozeniem spadochronu*/
}

if ( ( t>CZAS_DO_OTWARCIA)&&(t<=(CZAS_DO_OTWARCIA+CZAS_OTWARCIA)) )
{
return OPOR1+( ( OPOR2-OPOR1)* (t-CZAS_DO_OTWARCIA) ) / (CZAS_OTWARCIA) );
}

if (t>(CZAS_DO_OTWARCIA+CZAS_OTWARCIA))
{
return OPOR2; /*opor po rozlozeniu spadochronu*/
}
}

/***/

double f1(double czas, double wysokosc, double szybkosc)
{

```

```

return szybkość;
}

/***/

double f2(double czas, double wysokość, double szybkość)
{
    return ( (-G)+(opor(czas)*(szybkość*szybkość)) );
}

```

## 2. OPIS DZIAŁANIA

Program ten jest implementacją metody Runge-Kutta rozwiązywania równań różniczkowych, w tym wypadku równania opisującego ruch spadochroniarza w polu grawitacyjnym ziemi. Na podstawie czasu, wielkości kroku czasowego oraz wartości prędkości i wysokości w kroku czasowym  $n$  program oblicza wysokość skoczka oraz wartość prędkości i przyspieszenia w kroku czasowym  $n+1$ . Obliczenia są prowadzone aż do osiągnięcia przez skoczka poziomu ziemi, nawet jeśli wartość czasu nie osiągnęła swej maksymalnej wartości zadanej w odpowiedniej stałej symbolicznej.

### Format danych wejściowych

Program w postaci binarnej nieprzyjmuje danych wejściowych. Wszelkie dane początkowe dotyczące ruchu spadochroniarza są zawarte w kodzie źródłowym programu zdefiniowane jako stałe przy użyciu dyrektywy preprocesora `#define`. Są to liczby typu `double`.

### Format danych wyjściowych

W wyniku działania programu uzyskane zostają liczby typu `double`. Dane wyjściowe zapisywane są w pliku o nazwie zdefiniowanej przez stałą `PLIK_WYJŚCIOWY`. Ma on postać czystego pliku tekstowego.

Oto kilka przykładowych linijek z pliku wynikowego programu:

```

5.999800 854.238639 -37.208819 32.242134
6.000000 854.231197 -37.202368 32.264093
6.000200 854.223758 -37.195913 32.286019
6.000400 854.216319 -37.189454 32.307912

```

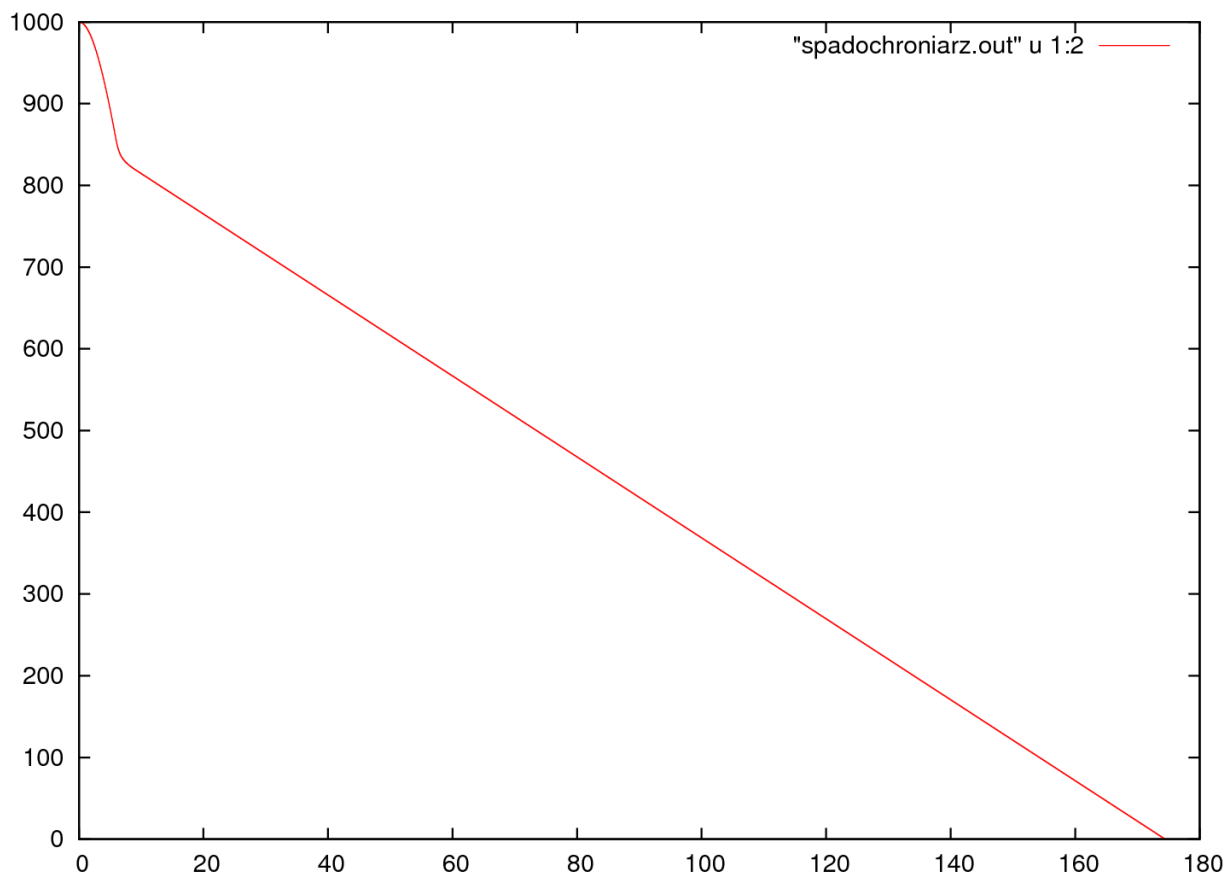
Kolejne kolumny oznaczają odpowiednio czas w sekundach, wysokość w metrach, szybkość w metrach na sekundę, przyspieszenie w metrach na sekundę do kwadratu.

### Złożoność obliczeniowa

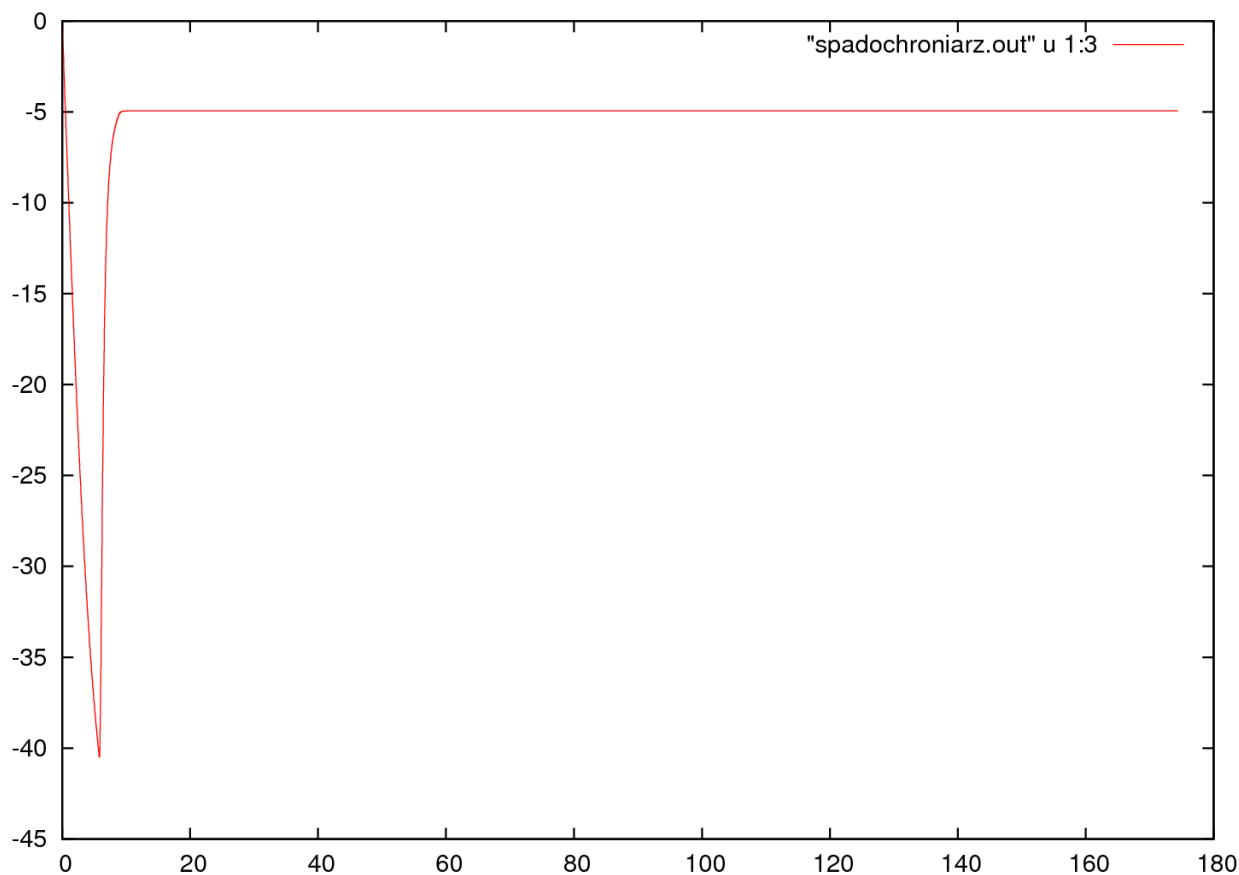
Czas generowania wyników oraz rozmiar pliku wynikowego zależą wprost proporcjonalnie do liczby zadanych mu przedziałów czasowych. Jest to zależność liniowa. Jednak dla czasu wykonywania możliwe są odstępstwa powyżej pewnych wartości liczby przedziałów czasowych wynikające z konfiguracji sprzętowej maszyny na której program jest wykonywany.

### 3. WYKRESY UZYSKANE NA PODSTAWIE DANYCH WYJŚCIOWYCH

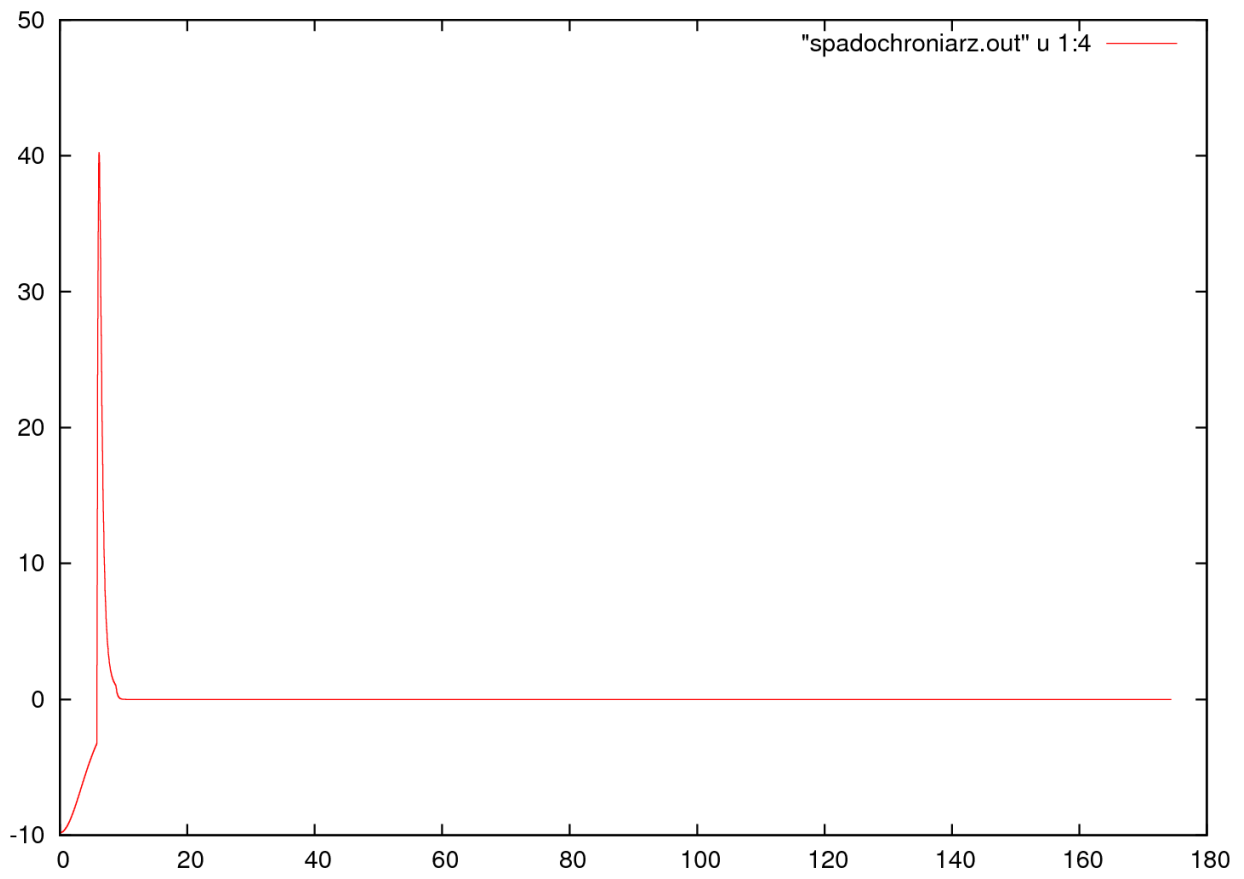
Do ich wykonania wykorzystano program GNUplot.



Wykres 1:  $h(t)$



Wykres 2:  $v(t)$



**Wykres 3:  $a(t)$**