VERIŽNA FONTANA

Področje: **FIZIKA**

Vrsta naloge: **Raziskovalna naloga**

Dijaka**:**

Tian Vesel, 2. A

Žiga Vaupotič, 2. A

Mentor: **Grega Celcar, prof. fiz.**

2022

Gimnazija Jožeta Plečnika Ljubljana

Vsebina

Seznam prilog

Povzetek

V tej raziskovalni nalogi smo raziskovali pojav verižne fontane. Jedro naloge stoji na predpostavljeni tuji teoriji. To bomo raziskali in po potrebi dodelali. Ob predpostavljeni teoriji to skušamo dokazati z različnimi poizkusi. Te rezultate bomo podatkovno obdelali. Hipoteze, ki smo si jih znotraj raziskovalne naloge zastavili bomo s pomočjo računalniške simulacije potrdili ali ovrgli.

V nalogi smo se spoprijeli z različnimi matematičnimi in fizikalnimi izzivi Pri tem smo morali razširiti svoje matematična obzorja v poglavja, ki jih gimnazijski program ne obravnava, npr. vektorska analiza.

Pomemben del naše naloge smo posvetili prav metodologiji zbiranja podatkov in delanju raziskav, saj je pojav obširen. To pomeni, da smo morali v samo izvajanje poizkusov, vključiti različne spremenljivke in parametre, ki vplivajo na potek in posledično vplivajo na rezultate pojava. Pri tem smo skušali z uporabo sodobne tehnologije čim natančneje izmeriti izhodne parametre pojava. S tem tudi želimo, da bi naša raziskovalna naloga imela praktično uporabo.

KLJUČNE BESEDE:
Verižna fontana, računalniška simulacija, klasična fizika, Newton

Abstract

In this project we took a deep dive into mistery of Newton″s beads. We compared and analyzed multiple theories based on which we could create and write an applicable and comprehensive end-theory. We tried to prove our theory by doing adequate amount of experiments, that have their basis in aforementioned theories. In pursue of understanding, we tackled different challenges, both mathematical and physical, despite only having our high-school knowladge to support us, which has made it impposible to resolve the mystery at hand without diving deeper in to the world of advanced mathematics such as vectory analysis. We have devoted important part of our thesis to the methodology of data collection and research, as the phenomenon is somewhat comprehensive, which means that we had to include wide range of variables and parameters in design of our thesis that influence the course and, consequently, the outcome of the phenomenon. We have tried to use modern technology to measure the output parameters of the phenomenon as accurately as possible. We also want that this project has an applicable outcome.

KEYWORDS:
Chain fountain, computer simulation, classical physics, Newton

1 Uvod

1.1 Namen

Fizika je naravoslovna veda, ki obsega področja kot so klasična mehanika, elektromagnetizem, termodinamika, itd.. Vsaka od teh področji ima neraziskane pojave, ki ostajajo uganka.

V preteklem letu je na spletni strani Youtube več ustvarjalcev s področja fizike prikazalo svoj pristop k reševanju problema verižne fontane. Ob ogledu videoposnetkov teh ustvarjalcev, smo se odločili, da temu pojavu končno pridemo do dna in s pomočjo raziskovalne naloge o verižni fontani, predstavimo in razrešimo ta zapleten problem.

1.2 Hipoteze

- Za začetek pojava verižne fontane ne potrebujemo začetne hitrosti.
- Višina fontane je odvisna od razlike višine med začetkom prostega pada in tlemi.
- Masa in velikost kroglic na verigi vplivata na pojav verižne fontane.

1.3 Metode za raziskovalno nalogo

Nalogo bomo razdelili na teoretičen in praktičen del.

Pri teoretičnem delu bomo s pomočjo matematike in Newtonovih zakonov skušali dokazati pojav verižne fontane na verigi. Ob tem bomo vse dobljene razultate primerjali z izvedljivim praktičnim delom naloge. S tem nam bo pomagala raučunalniška simulacija pojava, katero bomo pri sami raziskovalni nalogi naredili.

Pri praktičnem delu naloge smo primerljali rezultate opazovanega pojava. Pri tem smo želeli vključiti čim manj spremenljivk, ki bi vplivala na potek in prikaz pojava.

2 Pregled stanja

2.1 Kaj je verižna fontana

Verižna fontana ali Mouldov pojav, poznano tudi pod imenom Newtonove kroglice [1], je pojav, ki poteka pri verigah. Ta veriga prihaja iz mirujočega stanja v čaši preko roba čaše, kjer jo gravitacijska sila vleče navzdol. Pri tem se veriga začne dvigati v lok nad časo (slika 1). Večji kot je prosti pad verige, višji je lok verige.

Slika 1 – Prikaz verižne fontane.

3 Metodologije

3.1 Metodologija teorije

Teorijo bomo predpostavili in jo razvijali na nalogi, ki sta jo naredila John Biggins in Mark Werner [2].

3.2 Metodologija simulacije

3.2.1 Izbira vrste aplikacije

Da bo uporaba simulacije čim bolj natančna in hitrost tem boljša, smo naredili namizno simulacijo. Pri izdelavi simulacije smo uporabljali potrjene postopke iz teorije.

3.2.2 Izbira razvojnega orodja

Izdelava simulacije je zasnovana na različnih vrstah razvojnih orodij. Aplikacijo delimo na dva glavna dela. Zato bomo uporabljali dve razvojni orodji. Eden od delov je grafični vmesnik, drugi pa simuliranje. Razvojno orodje za GUI bo Microsoftov DirectX [3], ki je skupek knjižnic za izrisovanje na zaslonu. Microsoft DirextX je kompatibilen s programskim jezikom C++ [4]. Ob tem nam ponuja asinhrono delovanje in nam omogoča, da izrisovanje na zaslon in simulacija potekata sočasno. Simulacijo smo v programskem jeziku C++ napisali kar sami.

3.2.3 IDE

Za izdelavo aplikacije moramo izbrati integrirano razvojno okolje (IDE) [5], ki nam omogoča sestavljanje razvojnih orodij in zgradbo aplikacije. Izbrali smo Visual Studio [6] saj je eden izmed redkih IDE, ki omogoča zgradbo aplikacije za programski sistem Windows [7].

3.3 Metodologija raziskav

Metodologijo raziskav bomo razdelili na dva dela. V prvem delu bomo zbirali podatke s pomočjo snemanja videoposnetkov verige. V ta del bomo vključili tudi spreminjane spremenljivk, ki jih bomo razdelili na več kategorji:

- Tip oklepajočega telesa,
- višina padca,
- oblika verige,
- masa verige,
- velikost verige.

S spreminjanjem spremenljivk bomo zagotovili visoko natančnost. Dobljene razultate bomo primerjali z našo teorijo. Za primerjanje razultatov lahko uporabimo našo simulacijo, ki bo z vstavljanjem različnih spremenljivk omogočila hiter izračun razultatov.

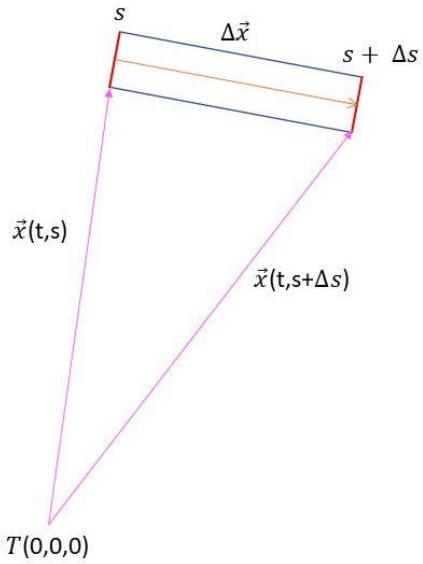
Drugi del raziskave predstavlja grafični prikaz meritev. Ob samem prikazu meritev bomo potrdili hipoteze in postavljeno teorijo. .

4 Teorija

4.1.1 Uvod v teorijo

Verigo tvorijo manjši členi (najpogosteje kroglice), ki se ponavljajo. Kroglice so povezane z vezmi. Verigo smo definirali kot tanko, fleksibilno in nestistljivo.

Da sledimo premikom naše verige smo uporabili dvodimenzionalni kartezični koordinatni sistem. Verigo razdelimo na več enakih členov, ki jih označimo kot s_1, s_2, s_3.



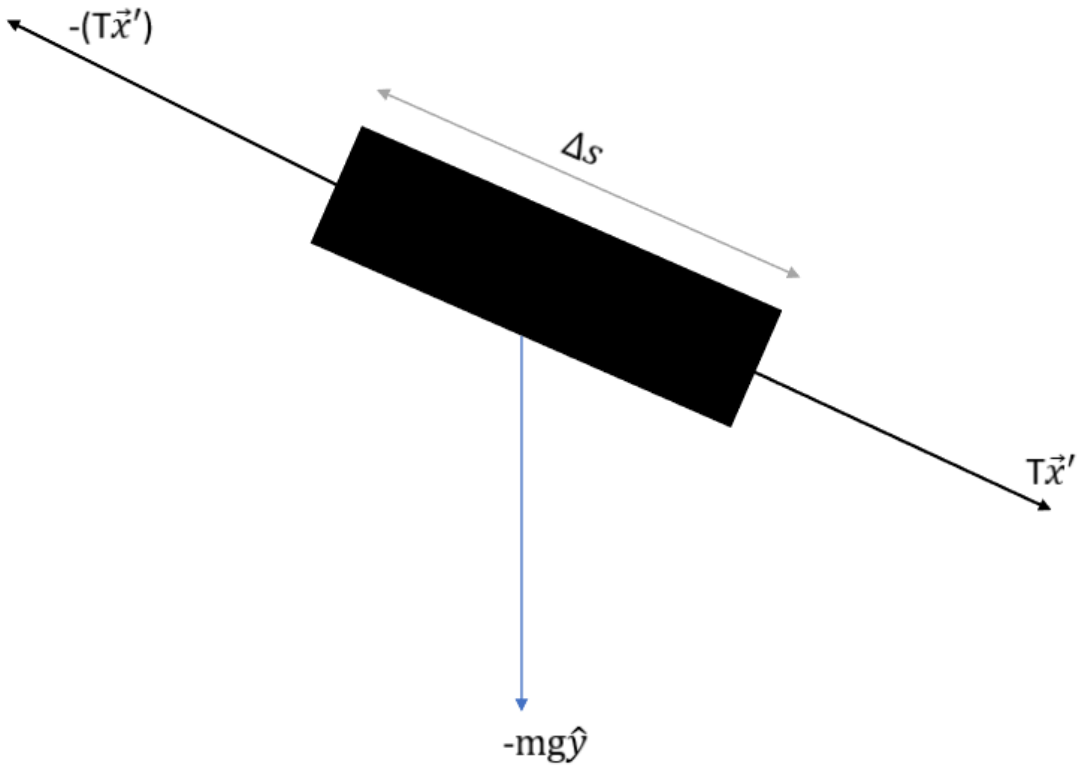
Slika 2 - Prikaz člena.

Čas in naše koordinate bomo zapisali kot funkcijo $\vec{x}$ (t,s). Razlika med dvema odsekoma na verigi je $\Delta\vec{x}$.

$\vec{x}(t,s)$ predstavlja vektor za vsako število t in s. Naša gol je definirati $\vec{x}$ za vsako točko s na verigi, kar lahko naredimo s tem da vzamemo odvod glede na število s

$$\vec{x}' = \frac{\partial \vec{x}}{\partial s}.$$

4.1.2 Uporaba 2. newtonovega zakona



Slika 3 - Prikaz sil, ki delujejo na člen.

Ko na člen narišemo sile (slika 3) opazimo, da imamo gravitacjisko silo, $Fg = -mg\hat{y}$ ($\hat{y}$ je enotni vektor). Ob tem imamo tudi dve sili, ki ju bomo poimenovali napetostni sili. Ti sili kažeta v nasprotnih smereh. S pomočjo $\vec{x}'$, defniramo eno silo kot $T\vec{x}'$, ki bo na lokaciji $s + \Delta s$. To silo bomo zapisali kot $(T\vec{x}')|_{s+\Delta s}$. Druga sila bo na lokaciji s in jo zapišemo kot $(T\vec{x}')|_s$.

Verigo smo opazovali v pospšenem gibanju. Zato moramo uporabiti drugi Newtonov zakon, ki je definiran kot $F_r = m \cdot a$. Na našo telo delujejo sila teže in napetostni sili. Rezultanto sil zapišemo kot.

$$F_r = (T\vec{x}')|_{s+\Delta s} - (T\vec{x}')|_s + (-mg\hat{y}),$$
$$m * a = (T\vec{x}')|_{s+\Delta s} - (T\vec{x}')|_s + (-mg\hat{y}).$$

Pospešek definiramo kot spremembo hitrosti v časovnem intervalu, ki ga bomo zapisali kot $a = \frac{\partial^2 \vec{x}'}{\partial t^2}$.

Maso definiramo kot produkt mase na enoto dolžine μ in dolžino člena Δs.

Z vsemi definiranimi spremenljivkami dobimo končno enačbo

$$(T\vec{x}')|_{s-\Delta s} - (T\vec{x}')|_s + (-\mu \Delta s g\hat{y}) = \mu \Delta s \cdot \frac{\partial^2 \vec{x}'}{\partial t^2}.$$

Enačbo lahko delimo s spremenljivko *s* in dobimo

$$(T\vec{x}')' - \mu g\hat{y} = \mu \frac{\partial^2 \vec{x}'}{\partial t^2 s}.$$

4.1.3 Iskanje stabilne rešitve

Spremenljivko x želimo zapisati v odvisnosti od poti, $\vec{x}(s)$. S tem dobimo parcialno diferencalno enačbo, ki jo lahko rešimo tako, da integriramo prejšjno rešitev po *s*. Ko enačbo integriramo dobimo

$$T\vec{x}' - \mu g\hat{y}s = \mu v^2 \vec{x} - \mu g\vec{k}.$$

Konstanta, ki jo dobimo pri integriranju, je enaka $\mu g\vec{k}$. Ko enačbo uredimo dobimo preprosto enačbo

$$\pm \mu g|\hat{y}s - \vec{k}|\vec{x}' = \mu g(\hat{y}s - \vec{k}).$$

Iz same enačbe nato izpeljemo $\vec{x}'$ in dobimo

$$\vec{x}' = \pm \frac{(\hat{y}s - \vec{k})}{|\hat{y}s - \vec{k}|}.$$

4.1.4 Striktni prehod v dveh dimenzijah

Kot že omenjeno, bomo našo rešitev prenesli v dvodimenzionalni prostor. S pomočjo prejšne ugotovitve bomo definirali komponente transformiranega vektorja x(s, t) kot x' in y'.

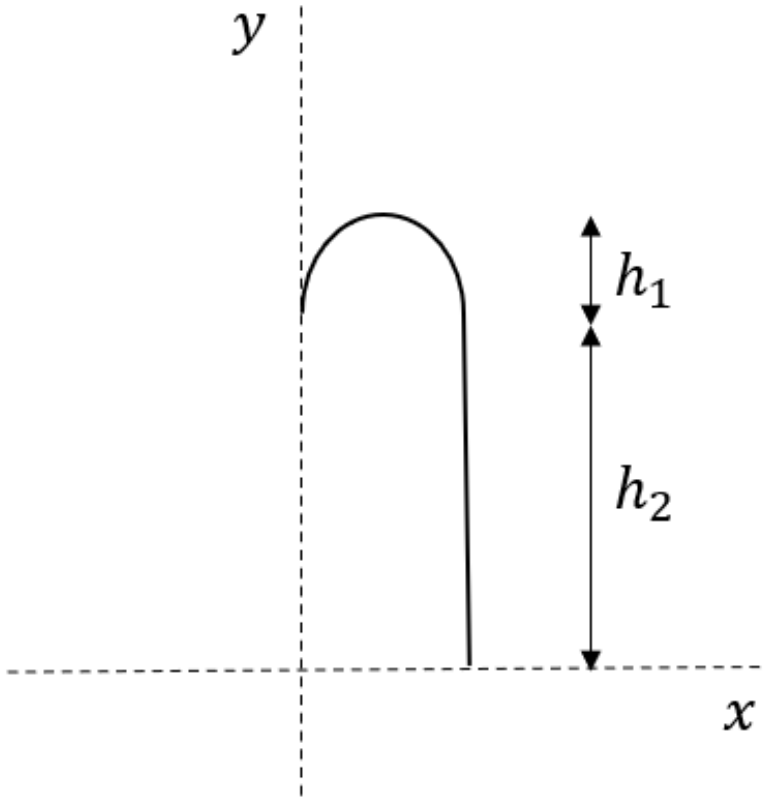
Ko definiramo x' dobimo enačbo

$$x' = \frac{\pm(-k_x)}{\sqrt{(k_x)^2+(s-k_y)^2}}.$$

Komponento y' lahko definiramo podobno kot vektor $\vec{x}'$ in sicer

$$y' = \frac{\pm(s-k_y)}{\sqrt{{k_x}^2+(s-k_y)^2}}.$$

4.1.5 Redefiniranje enotnih vektorjev



Slika 4 - Prikaz fontane

Ob pogledu na naši komponenti x' in y' ugotovimo, da imamo v zgornjem delu nedefiniran predznak, ki ni potreben, saj lahko predvidevamo, da bo y' vedno potekal v smeri navzdol, kakor tudi naša veriga (slika 4). Tako, lahko zapišemo zgornji del enačbe kot $-(s-k_y)$, s čimer je končna enačba za y' enaka

$$y' = \frac{k_y-s}{\sqrt{{k_x}^2+(s-k_y)^2}}.$$

Ob pogledu na sliko 4, lahko predpostavimo, da je $x' > 0$, saj lahko predvidevamo, da se vrv giblje v desno stran in posledično se pot veča v desno smer. S tem ugotovimo, da je $x' > 0$, kar posledično pomeni, da je tudi $k_x > 0$. Ko vključimo to v našo dobljeno enačbo ugotovimo, da zgornji del ulomka postane $-(-k_x)$. S tem dobimo končni del naše enačbe

$$x' = \frac{k_x}{\sqrt{(k_x)^2+(s-k_y)^2}}.$$

4.1.6 Integriranje enotnih vektorjev po poti

Če želimo priti do pozicije x in y v odvisnosti od poti, moramo integrirati x' in y' po poti.

$$x(s) = \int x' ds = k_x \ln\left[\frac{\sqrt{{k_x}^2 + (k_y - s)^2} - k_y + s}{\sqrt{{k_x}^2 + {k_y}^2} - k_y}\right]$$

Za integriranje y' po poti, moramo definirati nove spremenljivko Y_z, ki bo predstavljala začetno višino pojava

$$y(s) = \int y' ds = Y_z + \sqrt{{k_x}^2 + {k_y}^2} - \sqrt{{k_x}^2 + \left(k_y - s\right)^2}.$$

V prejšnjih enačbah smo definirali silo napetosti, kot funkcijo poti v tridimenzionalni prostoru, ki je:

$$T - \mu v^2 = \pm \mu g |\hat{y} s - \vec{k}|$$

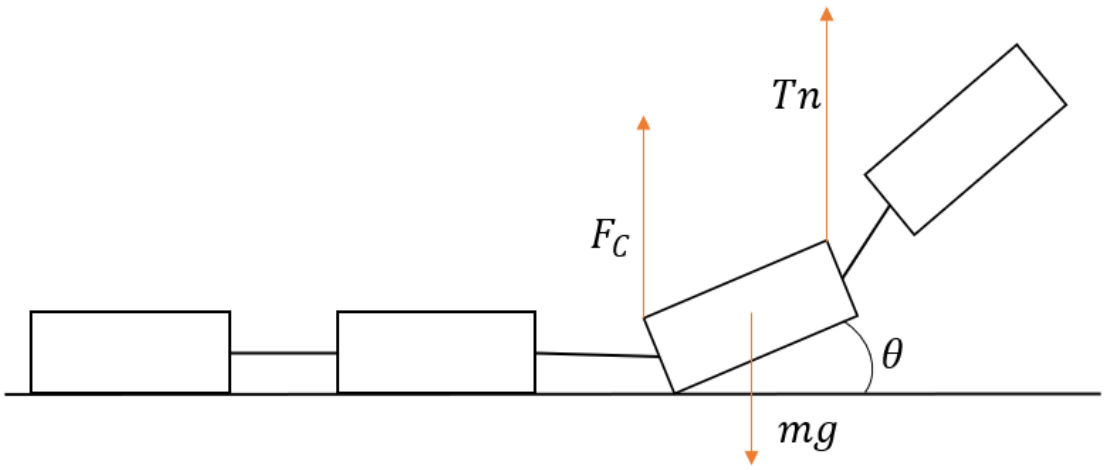
Za nadaljno izpeljavo, bi bilo dobro, da definiramo silo napetosti kot funkcijo poti tudi v dvodimenzionalnem prostoru, kar lahko naredimo s transformacijo, ki zgleda tako

$$T(s) = \mu v^2 - \mu g \sqrt{{k_x}^2 + \left(k_y - s\right)^2}.$$

Kot lahko opazite pa je zadnji del enačbe enak $y(s)$. S tem podatkom lahko z lahkoto zapišemo funkcijo sile napetosti v odvisnosti od y.

$$T(y) = \mu g(y - Y_z) + \mu v^2 - \mu g \sqrt{{k_x}^2 + {k_y}^2}$$

4.1.8 Zakaj pride do pojava



Slika 5 - Prikaz delovanja sil na člen med dvigovanjem.

Za razumevanje zakaj pojav sploh nastane, moramo stopiti nekaj korakov nazaj. Vemo da je pri pojavu verižne fontane $h_2 > 0$ (slika 3). Torej pogoj $h_2 > 0$ je da je skupna napetostna sila $T_n < \mu v^2$. Ta napetostna sila pa ni zadostna, za nastanek pojava verižne fontane. Obstajata še dve sili, ki delujeta na telo. To sta sili podlage, F_C in sila T_f, ki deluje tik nad tlemi. Ti dve sili tvorita rezultanto sil. Silo gravitacije v našem primeru lahko zanemarimo. Tako dobimo

$$T_n + F_C = \mu v^2.$$

Na izhodiščni podlagi morajo biti vse sile biti enake μv^2, kar pomeni da lahko F_C in T_f opišemo s konstantama α in β

$$F_C = \alpha\mu v^2 \text{ in } T_f = \beta\mu v^2$$

Če sta kostanti $\alpha = \beta = 0$, opazimo, da bo verigala padala kot v prostem padu (do pojava ne bo prišlo). S tem lahko združimo naše razultate in dobljene spremenljivke. Naši razultati kažejo na to, da sta h_2 in h_1 sorazmerni. Tako lahko predpostavimo, da je

$$\frac{h_2}{h_1} = \frac{\alpha}{(1-\alpha-\beta)} \text{ in } v = \sqrt{\frac{Y_z g}{1-\propto-\beta}}$$

Če je $\alpha = 0$ potem je tudi $h_2 = 0$.

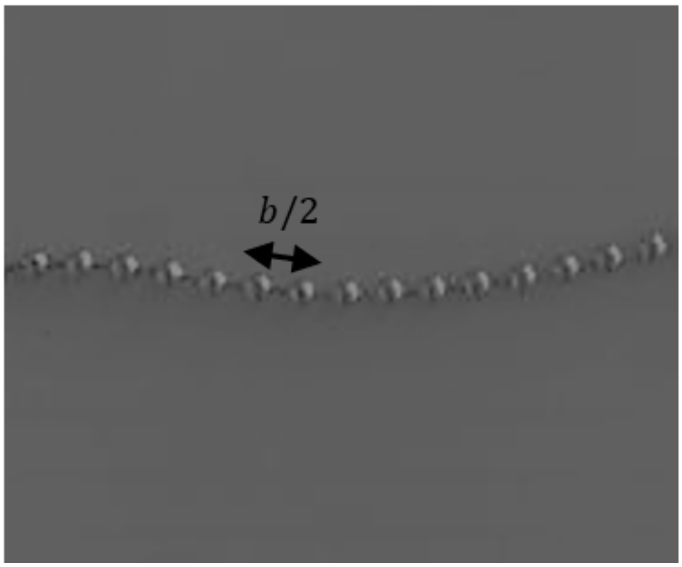
4.1.9 Zakon o ohranitvi energije pri pojavu

Izrek o kinetični in potenecialni energije nam pove, da se odseku spremeni kinetična energija za $\Delta Wk = \frac{\mu v^2}{2}$, med tem pa se mu potencialna energija spremeni za $\Delta Wp = \mu g Y_z$. Na podlagi prejšnih poglavji lahko zapišemo sorazmerje med njima kot

$$\frac{\Delta Wk}{\Delta Wp} = \frac{\frac{v^2}{2}}{gY_z} = \frac{\frac{Y_z g}{(1-\propto-\beta)}}{gY_z} = \frac{1}{2(1-\alpha-\beta)}$$

Kakor vidimo je $\frac{\Delta Wk}{\Delta Wp} = \frac{1}{2}$, ko sta konstanti $\alpha = \beta = 0$, kar pomeni da se polovico energije izgubi v procesu »pobiranja«. Tako je lahko v najboljšem primeru $\alpha = \frac{1}{2}, \beta = 0$, kar predstavlja sistem verižne fontane brez energijskih izgub.

4.1.10 Kaj predstavlja konstanta α



Slika 6 - Prikaz parametra b/2.

Na začetku pojava imamo opravka z enakomernim pospeškom a in kotnim pospeškom $\dot{\omega}$. Pospešek a lahko zapišemo kot

$$a = (\frac{b}{2})\,\dot{\omega},$$

kjer je b velikost člena (slika 6). S pomočjo tega si lahko zdaj predstavljamo potovanje člena verige. Člen je definiran na podlagi mase m, velikosti b in vzrajnostnega momenta J. Tako se v našem primeru vsak člen dvigne s pomočjo napetostne sile T_n in sile podlage F_C. Z dobljenimi meritvami in drugim Newtonovim zakonom lahko sedaj zapišemo

$$T_n + F_C = ma$$

Sedaj namesto pospeška a vstavimo dobljeno povezavo in zamenjamo maso $m = \frac{J}{b^2}$ in dobimo enačbo, ki je že urejena

$$T_n + F_C\left(\frac{b}{2}\right) = J\dot{\omega}.$$

S pomočjo sistema enačb lahko končno izrazimo F_C, ki je enak

$$F_C = \frac{1}{2}ma(1 - \frac{J}{\left(\frac{1}{4}\right)mb^2})$$

Iz enačbe prejšnjega poglavja $F_C = \propto \mu v^2$, pa lahko izrazimo tudi

$$\alpha = \frac{F_C}{\mu v^2} = \frac{1}{2}(1 - \frac{J}{\left(\frac{1}{4}\right)mb^2})$$

Vztrajnosti moment za kroglo je $J = 2\left(\frac{m}{3}\right)\left(\frac{b}{2}\right)^2 = \frac{mb^2}{6}$. Ko to povezavo vstavimo v zgornjo enačbo dobimo enačbo za α, ki je

$$\alpha = \frac{F_C}{\mu v^2} \cong \frac{1}{6}.$$

4.1.11 Končni izračun k_x na podlagi novo dobljenih spremenljivk

Z novo dobljenimi spremenljivkami, lahko končno izračunamo prej uvedeno spremenljivko k_x.

To lahko naredimo s pomočjo dobljenih enačb iz prejšnega poglavja:

$$F_C = \mu g \sqrt{{k_x}^2 + {k_y}^2} = \alpha \mu v^2$$

Iz dobljenih enačb lahko s pomočjo sistema enačb izrazimo v:

$$\mu g\, Y_z = (1 - \alpha - \beta)\mu v^2$$

$$v = \sqrt{\frac{g Y_z}{1 - \alpha - \beta}}$$

Zdaj lahko vzamemo prvo enačbo in jo rešimo za k_x

$$\sqrt{{k_x}^2 + {k_y}^2} = \alpha \frac{Y_z}{1 - \alpha - \beta}$$

$$k_x = \sqrt{(\frac{\alpha Y_z}{1 - \alpha - \beta})^2 - {k_y}^2}$$

4.1.12 Končne izepljave $x(s)$ in $y(s)$

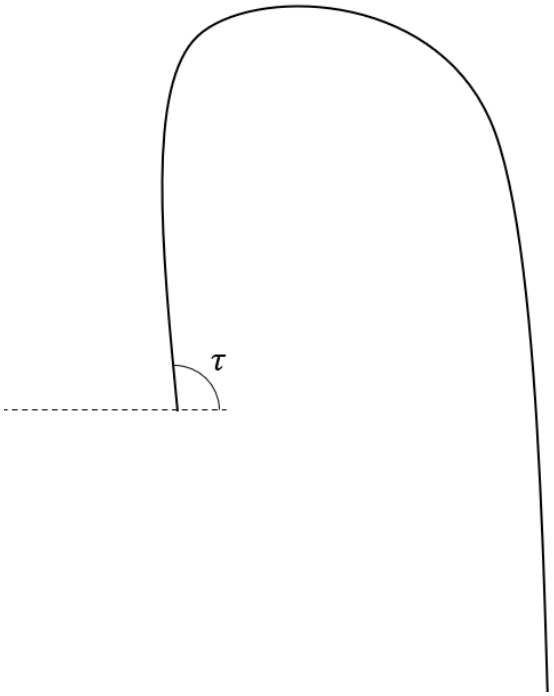
Ko vstavimo naš novo dobljen k_x v formulo $x(s)$ in formulo $y(s)$, dobimo poenostavljeno enačbo:

$$x(s) = \sqrt{(\frac{\alpha Y_z}{1 - \alpha - \beta})^2 - {k_y}^2} \ln\left[\frac{\sqrt{(\frac{\alpha Y_z}{1 - \alpha - \beta})^2 - 2k_y s + s^2} - k_y + s}{(\frac{\alpha Y_z}{1 - \alpha - \beta}) - k_y}\right]$$

$$y(s) = Y_z + \left(\frac{\alpha Y_z}{1 - \alpha - \beta}\right) - \sqrt{\frac{\alpha Y_z}{1 - \alpha - \beta} - 2k_y s + s^2}$$

4.1.13 Končni izračun k_y

Ob ponovnem pogledu na sliko 4 ugotovimo, da sta začetni smeri $y'(0) = \cos(90 - \tau);\ \tau < 90°$, τ je kot med podlago in obliko verige (slika 7). Ta kot je ponavadi okoli 90°.



Slika 7 - Kot τ prikazan na sliki

Predpostavimo da je $x(s) = 0$ in torej ugotovimo, da je k_y:

$$y'(0) = \frac{k_y}{\frac{\alpha Y_z}{1-\alpha-\beta}} = \cos(90-\tau);\ \tau < 90° => k_y = \cos(90-\tau)\frac{\alpha Y_z}{1-\alpha-\beta}; \tau < 90°$$

S končno enačbo $y(s)$:

$$y(s) = \frac{Y_z(1-\beta)}{1-\alpha-\beta} - \cos(90-\tau)\left|\frac{\alpha Y_z}{1-\alpha-\beta} - s\right|$$

Torej to pomeni, da bo najvišja možna lega y_{max}:

$$y_{max} = \frac{Y_z(1-\beta)}{1-\alpha-\beta}$$

S pomočjo te enačbe lahko izračunamo najvišjo možno lego, pri čemer pa ne smemo pozabiti, da to velja samo ob popolnih pogojih, ki jih v resničnem življenju ne moramo doseči, saj ta enačba ne vključuje trenja s podlago, sile gravitacije, trenja med verigo in oviro in še trenja med vezmi.

Kot zaključek bi pa pa radi poudarili, da je kljub izpeljanimi enačbami večino teh poenostavljenih. Na žalost pri tem pa izgubimo popolno natančnost in nekaj možnih rešitev.

5 Razultati z diskusijo

5.1 Simulacija

V okviru projekta smo izdelali tudi nizko nivojsko grafično računalniško simulacijo, ki s pomočjo teoretičnih predpostavk in neovrednotenih enačb projeciran in simulira posamezne verige. Ta jih sicer postavi v bolj laboratorjiske okoljšičine, ampak kljub temu lahko še vedno s pomočjo simulacije predvidimo višino in obliko verige s povprečno relativno napako 5 %.

Za delovanje potrebuje aplikacija le operacjiski sistem WINDOWS in datoteko z vnesenimi podatki. Tako je ta lahka za uporabo ob tem pa je lahko uporabljena za simuliranje efekta v skoraj vseh okoliščinah.

5.1.1 Uporabniški vmesnik

Eden glavnih delov simulacije je grafično prikazati pojav za lažje opazovanje in primerjanje pojava z resničnim življenom. To smo zagotovili s pomočjo orodja Microsoft DirectX [8]. Ta nam s svojimi funkcijami omogoča risanje na ročaj aplikacije sistema Windows. Najprej smo s pomčjo Windows knjižnice registrirali novo okno aplikacije (slika 7).

```
m_pHWND = CreateWindow(
    TEXT("Simulation"),
    TEXT("Chain Fountain Effect Simulation"),
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    NULL,
    NULL,
    hInstance,
    NULL);
```

Slika 8 - Registracija novega okna.

Nato smo s pomočjo nosilca m_pHWND začeli DirectX process (slika 8, 9).

```
ShowWindow(m_pHWND, iCmdShow);
InitD3D(m_pHWND);
Simulation::CallsInit(m_pD3DDev, m_pHWND);
UpdateWindow(m_pHWND);
```

Slika 9 - Začetek DirectX procesa.

Ob tem pa tudi naložimo nastavitve in parametre simulacije v notranji polnilnik naše aplikacije. S pomočjo objekta m_pD3D (slika 10) imamo dostop do risalnega nosilca Windowsove aplikacije. To nam omogoča da dodajamo nove like, ki bodo nato izrisani.

```
void InitD3D(HWND hWnd)
{
    m_pD3D = Direct3DCreate9(D3D_SDK_VERSION);

    D3DPRESENT_PARAMETERS d3dpp;

    ZeroMemory(&d3dpp, sizeof(d3dpp));
    d3dpp.Windowed = TRUE;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.hDeviceWindow = hWnd;


    m_pD3D->CreateDevice(D3DADAPTER_DEFAULT,
        D3DDEVTYPE_HAL,
        hWnd,
        D3DCREATE_SOFTWARE_VERTEXPROCESSING,
        &d3dpp,
        &m_pD3DDev);
}
```

Slika 10 - Funkcija za začetek DirectX procesa.

5.1.2 Osveževanje simulacije

S prva smo nameravali, da bi simulacijo in izrisevanje grafičnega vmesnika osveževali asinhrono. Ugotovili pa smo, da to seveda ne bo potrebno, ker simulacija ne uporablja težkih matematičnih računov. Tako bomo simulacijo in grafično izrisevanje osveževali naenkrat oz. takrat ko DirectX izrisuje like na zaslon. To lahko naredimo s pomočjo »callback« funkcije WndProc(). Ko dobimo "callback" WM_PAINT (slika 11), lahko začnemo s simuliranjem in izrisevanjem.

```
LRESULT CALLBACK WndProc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    HDC          hdc;
    PAINTSTRUCT  ps;

    switch (message)
    {
    case WM_CREATE:
        AllocConsole();
        freopen("conin$", "r", stdin);
        freopen("conout$", "w", stdout);
        freopen("conout$", "w", stderr);
        return 0;
    case WM_PAINT:
        Simulation::Calls::OnPaint(m_pD3DDev);
        return 0;
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
}
```

Slika 11 - "Callback" funkcija

5.1.3 Nalaganje parametrov na notranji polnilnik

Preden lahko izvedemo simulacijo moramo prejeti vse parametre in jih naložiti na notranji polnilnik. To nam omgoča takojšni dostop do teh prarametrov znotraj simulacije. Parametere bomo naložili iz params.txt. To datoteko lahko preberemo, preden pa lahko to naredimo pa smo se morali odločiti kateri protokol zapisovanja in branja datoteke bomo uporabili. Nato, da bomo imeli le nekaj parametrov smo se odločili, da bomo za protokol branja in zapisovanja datotek uporabljali kar funkcije znotraj paketa winbase.h. Tako smo naredili preprost sistem za nalaganje parametrov (slika 12).

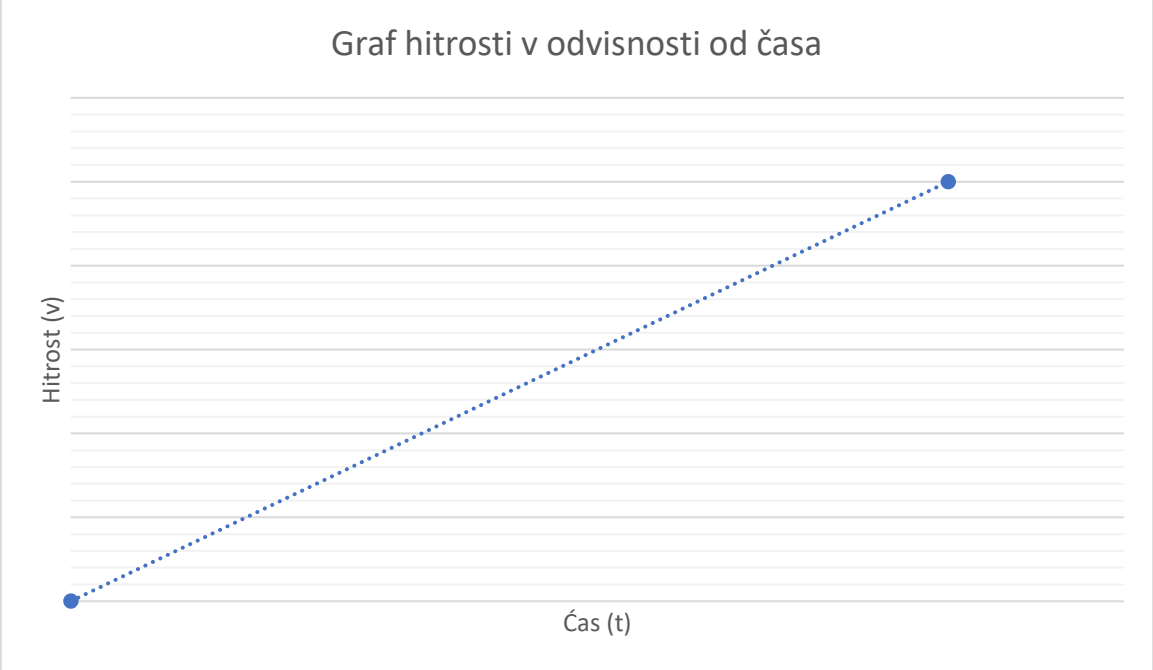
```
void Load()
{
	std::string m_strFile = std::string(m_pPath) + "\param.ini";

	if (!FileExists(m_strFile))
	{
		Save();
		return;
	}

	char value_l[32] = { '\0' };

	for (auto value : m_pFloats)
	{
		GetPrivateProfileStringA(value->category.c_str(), value name.c_str(),
"", value_l, 32, m_strFile.c_str());
			*value->value = atof(value_l);
	}
}
```

Slika 12 - Funkcija za nalaganje parametrov.

5.1.4 Simuliranje pojava

Pojav bomo simulirali na podlagi teorije, ki smo jo opisali. Eden glavnih problemov s katerimi smo se srečali je, da smo definirali $y(s)$ ob končni hitrosti. Ta problem je bil rešen s pomočjo interpolacije. Funkcijo $y(s)$ smo interpolirali glede na graf hitrosti v, na katerega v intervalu $t = [0, t_{v_k}]$ vpliva enakomerni pospešek a. Posledično bo graf hitrosti v odvisnosti od časa do t_{v_k} linearen, kar pomeni, da lahko uporabimo linearno interpolacijo. Graf 1 prikazuje v(t), ki se razteza od 0 in do končne hitrosti, ki jo dobimo s pomočjo enačbe

$$v_k = \sqrt{\frac{gY_z}{1-\alpha-\beta}}.$$



Graf 1 - Prikaz grafa hitrosti v odvisnosti od časa.

Interpolirali bomo po standardnem pravilu interpolacije med dvema točkama $y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$. Za naše potrebe smo enačbo prilagodili. Funkcijo y(s) bomo interpolirali glede na točko y(0).S pomočjo te enačbe lahko sprogramiramo funkcijo interpolacije (glej sliko 12)

$$y = y_0 + \frac{v}{v_k}(y(x) - y_0).$$

```
float Interpolate(float y0, float y, float t)
{
    return y0 + t * (y - y0);
}
```

Slika 13 - Interpolacije funkcije y(s)

Simulacija je zgrajena na predpostavki, da jo lahko simuliramo le s pomočjo poti. Simulirali jo bomo tako, da bomo za vsak korak dodali pot glede na hitrost in hitrost osveževanja

$$s = s + v * \frac{1}{n},$$

kjer n predstavlja osvežitev na sekundo. Parameter n dobimo s pomočjo meritve iz prejšnje simulacije (slika 14). Simulirali jo bomo tako, da bomo za vsak korak dodali pot glede na hitrost in hitrost osveževanja.

```
m_flCurrentTicks = clock();
Simulation::Calls::OnPaint(m_pD3DDev, m_flFPS);

m_flDeltaTicks = clock() - m_flCurrentTicks;
if (m_flDeltaTicks > 0)
{
    m_flFPS = CLOCKS_PER_SEC / m_flDeltaTicks;
}
```

Slika 14 - Merjenje števila osvežitev.

Preostane nam le še modeliranje poti (slika 15) s pomočjo zgoraj zapisane enačbe. Ob tem pa dodajamo še stare količine s v nabor objetkov za izrisevanje simulacije.

```
void OnPhyUpdate(float m_flFPS)
{
	Globals::m_pObjects.push_back(Globals::s);
	Globals::s += VAL * (1 / m_flFPS);
}
```

Slika 15 - Moduliranje s na osvežitev.

S pomočjo vhodnih parametrov nam sedaj preostane še simuliranije $y(s)$ in sestavljanje kostante α. Funkcijo $y(s)$ lahko zapišemo s pomočjo enačbe

$$y(s) = \frac{Y_z(1-\beta)}{1-\alpha-\beta} - \cos(90-\tau)\left|\frac{\alpha Y_z}{1-\alpha-\beta} - s\right|.$$

α zapišemo s pomočjo enačbe $\alpha = \frac{F_c}{\mu v^2}$. Programska koda je prikazana na sliki 16.

```
    float a()
    {
        return (BHALFS / 2) * ACCELERATION;
    }
    float Fc()
    {
        return 0.5 * MASS * a() * 0.3f;
    }
    float Alpha()
    {
        return Fc() / (MU * powf(VELOCITY, 2));
    }
    float MaxY()
    {
        return ((STARTHEIGHT * (1 - BETA)) / (1 - Alpha() - BETA));
    }
    float SimulateY(float s)
    {
        return ((STARTHEIGHT * (1 - BETA)) / (1 - Alpha() - BETA) - abs(cos(90 *
PI / 180.0 - THETA * PI / 180.0) * ((Alpha() * STARTHEIGHT) / (1 - Alpha() -
BETA)) - s));
    }
```

Slika 16 - Simuliranje y(s) in konstante α.

5.1.5 Izrisevanje simulacije

Ker bi radi s pomočjo simulacije dobili kar se da zanesljive razultate, bomo v simulacijo vpeljali tako imenovan »bias protokol«. Pri tem bomo simulacijo ponovili večkrat s pomočjo spreminjanja parametraβ, ki je po naših meritvah najmanj zanesljiv parameter. Tega bomo spreminjali za podan parameter s na intervalu $n = [\beta - k, \beta + k]$ (k je podan parameter). To bomo dosegli z zanko, ki nariše simulacijo (slika 17).

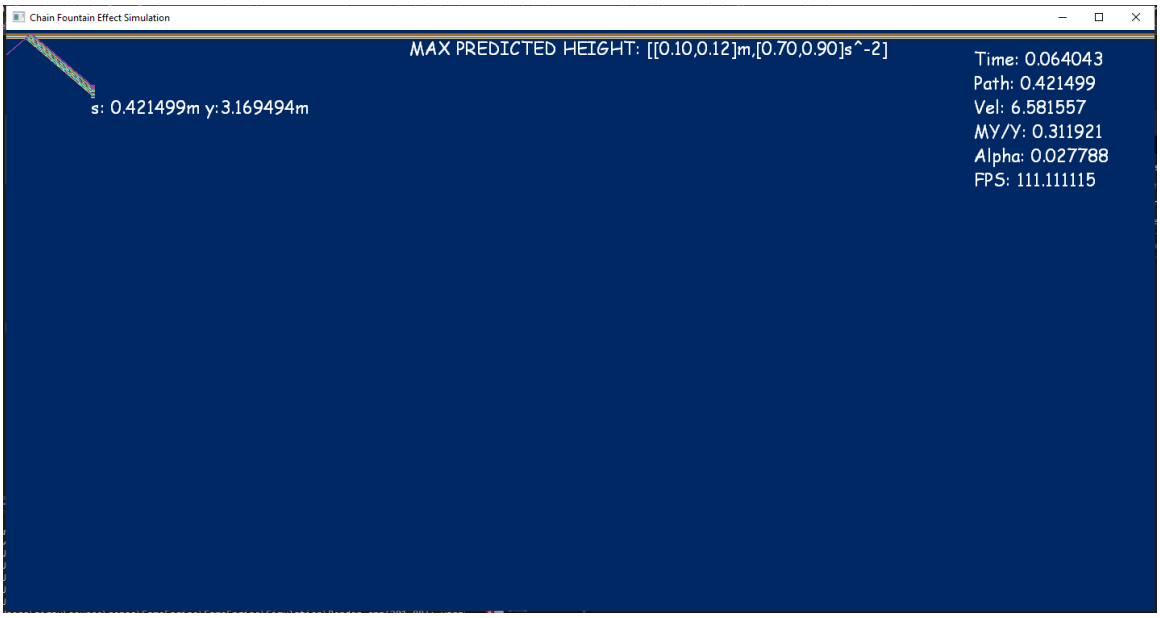
```
            for (float i = BETA - K; i < BETA + K; i+=S)
            {
                float acc = BETA;
                BETA = i;

                m_nColor++;

                Render::DrawFillRect(m_pD3DDev, SimulateX2(Globals::s) * 5,
Globals::m_vecScreenSize.y - SimulateY(Globals::s) * m_flExtendFactor, 5, 5,
m_pRandomColors[m_nColor].r, m_pRandomColors[m_nColor].g,
m_pRandomColors[m_nColor].b);

                for (int j = 0; j < Globals::m_pObjects.size(); j++)
                {
                    Render::DrawFillRect(m_pD3DDev, SimulateX2(Globals::m_pObjects[j])
* 5, Globals::m_vecScreenSize.y - SimulateY(Globals::m_pObjects[j]) *
m_flExtendFactor, 1, 1, m_pRandomColors[m_nColor].r, m_pRandomColors[m_nColor].g,
m_pRandomColors[m_nColor].b);
                }

                Render::DrawFillRect(m_pD3DDev, 0, Globals::m_vecScreenSize.y -
STARTHEIGHT * m_flExtendFactor, (MaxY() - STARTHEIGHT) / tan(90 * PI / 180.0 - THETA *
PI / 180.0), 5, m_pRandomColors[m_nColor].r, m_pRandomColors[m_nColor].g,
m_pRandomColors[m_nColor].b);
                Render::DrawFillRect(m_pD3DDev, 0, Globals::m_vecScreenSize.y - MaxY()
* m_flExtendFactor, 2000, 1, m_pRandomColors[m_nColor].r, m_pRandomColors[m_nColor].g,
m_pRandomColors[m_nColor].b);

                m_flMaXHeights.push_back(std::make_pair(MaxY() - STARTHEIGHT, BETA));

                BETA = acc;
            }
```

Slika 17 - Znaka za risanje simulacije.

Tako smo simulirali $y(s)$ po naši teoriji. Dodamo še nekaj elementov za spremljanje poteka simulacije (slika 18). S tem smo tudi končali našo simulacijo (sliko 19).

```
Render::RenderText(std::string("Time: " + std::to_string(Globals::s / v())), 1200,
20);
Render::RenderText(std::string("Path: " + std::to_string(Globals::s)), 1200, 50);
Render::RenderText(std::string("Vel: " + std::to_string(v())), 1200, 80);
Render::RenderText(std::string("MY/Y: " + std::to_string(MaxY() -
SimulateY(Globals::s))), 1200, 110);
Render::RenderText(std::string("Alpha: " + std::to_string(Alpha())), 1200, 140);
Render::RenderText(std::string("FPS: " + std::to_string(m_flPrevFPS)), 1200, 170);
```

Slika 18 - GUI elementi za spremljanje simulacije.



Slika 19 - Prikaz simulacije

5.2 Zbiranje meritev in raziskovanje pojava

Veliko časa nam je pobrala priprava na dejansko izvedbo eksperimenta. Pojav pri eksperimentu je bil zelo kratko trajen, zato smo morali biti hitri in natančni pri opažanjih. V ta namen so nam v pomoč prišli video posnetki. Te posnetke smo med samo izvedbo eksperimentov posneli. Odčitali smo opažene podatke in jih zapisali v tabelo 1.

Skupaj smo izvedli 28 eksperimetov, 24 v navadni čaši in štiri v skodelici. Na voljo smo imeli tri različne verige. Dve verigi sta imeli kroglice z različnima velikostma, medtem ko je ena imela sponke. Vendar smo pri naših ekspermentih uporabili samo verige s kroglicami, saj je pojav pri teh bil najbolj opažen. Naši rezultati so bili zelo podobni kot v naših simulacijah.

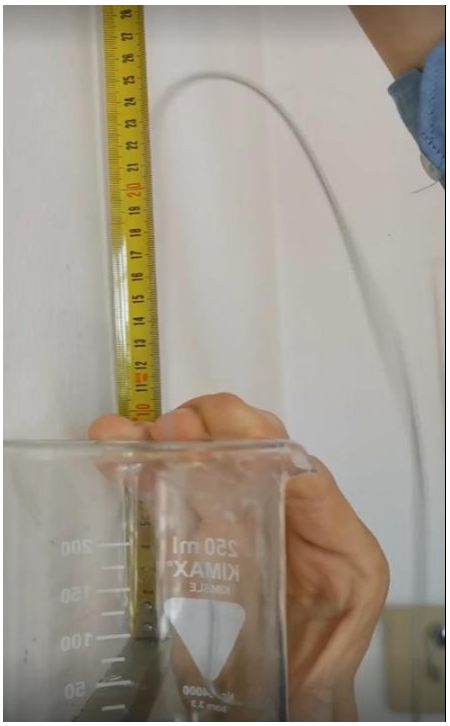
Slika 20 - Zbiranje meritev.

V eksperimentih smo se poigrali z različnimi parametri. Spreminjali smo razdaljo čaše do tal, začetno hitrost, položaj verige, zamenjava pospeška zemlje z električnim motorjem, odvzemom čaše in s postavljanjem verige na ravno podlago.

Slika 21 - Prikaz pojava v skodelici.

Eksperimeti niso imeli veliko ekstremov in so bili v mejah normale in pričakovanj.

Kot kamero smo uporabili preprosti pametni telefon znamke Huawei P30 Pro. Ta nam je omogočil upočasnitev video posnetkov in lažje odčitavanje. Pri obdelavi posnetkov smo uporabili tudi program za obdelavo posnetkov.



Slika 22 - Prikaz dobrega pojava verižne fontane.

Samo izvedba eksperimentov smo se lotili na več načinov.

Verigo smo lepo razporejeno položili na ravna tla in jo z roko vlekli za en konec. Pri tem primeru je bil pojav opazen. V nadaljevanju smo zamenjali vleko z roko z električnim vrtalnikom okrog katere smo napeljali konec verige. Pri tem poskusu je bil Mouldov efekt bolj izrazit.

Pri nadaljnih eksperimentih smo se iz učilnice preselili na stopnišče, kjer smo verige spuščali na daljših razdaljah. Naše dobljene rezultate smo usklajevali z našo simulacijo. Ti rezultati so se dokaj ujemali, vendar smo naleteli na nekaj ekstremov.

Slika 23 - Slika poteka merjenja na stopnišču.

Eksperimente, ki smo jih izvedli na stopnišču z daljšim padcem in ekspreimente v učilnici smo primerjali med seboj. Primerjali smo, če se dobljeni rezultati ujemajo z našo teorijo. V večini primerov se naši dobljeni rezultati ujemajo s teorijo. Tu pa tam smo imeli kakšen ekstrem.

Za boljši efekt smo rob čaše in skodelice namazali z milom. S tem smo zmanjšali trenje med kroglicami in robom čaše. To je predvsem pomembno na začetku, saj predno pride do dviga verige, veriga drsi po robu navzdol. S tem smo tudi dosegli bolj opazen pojav.

5.2.1 Eksperimentalna telesa

Pri izvedbi eksperimentov in zbiranju podatkov smo uporabili verige ki so setavljene iz večjih okroglih manjših členov, ki so povezani s tanjšim kovinskim povezovalnim členi. Verige so fleksibilne in trdne.

Za eksperimente smo uporabili tri različne verige. Vse verige so imele podobne lastnosti, vendar so se ločevale po velikosti in masi posameznih kroglic. Krogice 1 so najmanjše po velikosti in masi. Kroglice 1 so tudi najkrajše po dolžine, saj merijo le 507,2 cm. Kroglice 2 so srednje velikosti vendar najdaljše po dolžini verige, saj merijo 1049,1 cm. Kroglice 3 so največje po velikosti, masi in merijo 700 cm (sliki 24 in 25).

Tabela 1 - Podatki o verigah

Ime telesa	Števila členov	Dolžina člena[cm]	Dolžina verige[cm]
Kroglice 1	2601	0,195	507,2
Kroglice 2	3040	0,345	1049,1
Kroglice 3	2100	0,333	700

Slika 24 - Kroglice 2

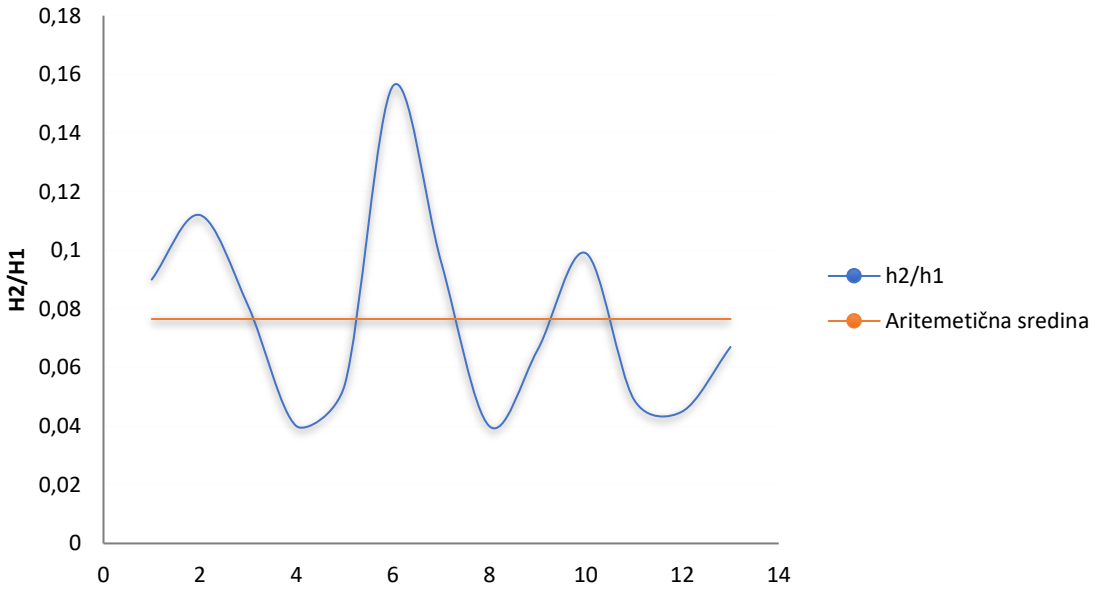
Slika 25 - Kroglice 3

5.2.2 Rezultati testiranja telesa kroglice 2

Razultati testiranja pojava, s pomočjo telesa kroglice 2 nam je dalo najboljše razultate. S tem telesom smo ves čas dobivali najboljše razultate. Vzrok za to je, da je konstanta β pri tej verigi, v primerjavi z drugimi verigami, zelo nizka.

Indeks	h_1[cm]	h_2[cm]	$\frac{h_2}{h_1}$ [m]	h_t[cm]	Tip oklepajočega telesa	Ime telesa
1.	215	19,5	0,091	9,3	Čaša	Kroglica 2
2.	215	24,2	0,112	9,3	Čaša	Kroglica 2
3.	215	17,5	0,081	9,3	Čaša	Kroglica 2
4.	215	21,4	0,099	9,3	Čaša	Kroglica 2
5.	215	11,8	0,054	9,3	Čaša	Kroglica 2
6.	82	12,8	0,156	9,3	Čaša	Kroglica 2
7.	215	19,5	0,091	9,3	Čaša	Kroglica 2
8.	320	12,6	0,039	9,3	Čaša	Kroglica 2
9.	300	19,9	0,066	9,3	Čaša	Kroglica 2
10.	300	12,2	0,041	9,3	Čaša	Kroglica 2
11.	300	14,8	0,049	9,3	Čaša	Kroglica 2
12.	300	13,7	0,046	9,3	Čaša	Kroglica 2
13.	300	20,2	0,067	9,3	Čaša	Kroglica 2
aritmetična sredina			0,08			

Tabela 2 - Prikaz razultatov telesa kroglice 2 v telesu čaša



Graf 2 - h2 v odvisnosti od h1 pri telesu krogljice 2.(AS predstavlja aritmetično sredino)

Kakor lahko vidimo pri grafu 2 so razultati testiranj zelo različni. To je predvsem zato, ker so stranski paramteri, kot je trenje in zapletanje verige, močno razlikovali. Tako smo s pomočjo aritmetične sredine dobili povprečno razmerje $\frac{h_2}{h_1}$. S pomočjo dobljenega razmerja lahko sedaj izračunamo *konstanto* beta β. V tem primeru je ta enaka $\beta \cong -1{,}7$. Ko vpisemo sledeče podatke v simulacijo nam ta vrne, da h_2 z začetno višino $Y_z = 3m$ ($Y_z = h_1$) leži na intervalu $k = [0{,}18; 0{,}2]$, to seveda velja za brezhibne pogoje. Vseeno pa lahko opazimo, da je višina h_2, kljub slabim pogojem, zelo podobna dobljeni višini simulacije.

Indeks	h_1[cm]	h_2[cm]	$\frac{h_2}{h_1}$ [m]	h_t[cm]	Tip oklepajočega telesa	Ime telesa
1.	82	13,4	0,163	5,8	Skodelica	Kroglica 2
2.	215	14,2	0,066	5,8	Skodelica	Kroglica 2
3.	215	12,5	0,058	5,8	Skodelica	Kroglica 2
4.	215	10	0,046	5,8	Skodelica	Kroglica 2
aritmetična sredina			0,06			

Tabela 3 - Prikaz razultatov telesa kroglice 2 v telesu skodelica.

Če primerjamo tabelo 3 s tabelo 2 lahko opazimo, da so dobljeni razultati različni kljub enakemu telesu. To je predvsem zaradi oblike skodelice in kota s katerim veriga pospešuje iz oklepajočega telesa.

5.2.3 Razultati testiranja telesa kroglice 1

Indeks	h_1[cm]	h_2[cm]	$\frac{h_2}{h_1}$ [m]	h_t[cm]	Tip oklepajočega telesa	Ime telesa
1.	215	11,0	0,051	9,3	Čaša	Kroglica 1
2.	215	10,4	0,048	9,3	Čaša	Kroglica 1
3.	215	11,7	0,054	9,3	Čaša	Kroglica 1
aritmetična sredina			0,05			

Tabela 4 - Prikaz razultatov telesa kroglice 1.

Kakor lahko opazimo so pri telesu kroglice 1 razultati bistveno nižji, to je predvsem zaradi oblike kroglice in masi, kajti te so precej manjše in lažje. Zaradi tega se spremeni tudi vztrajnostni moment, kar posledično vpliva na spremenljivko α.

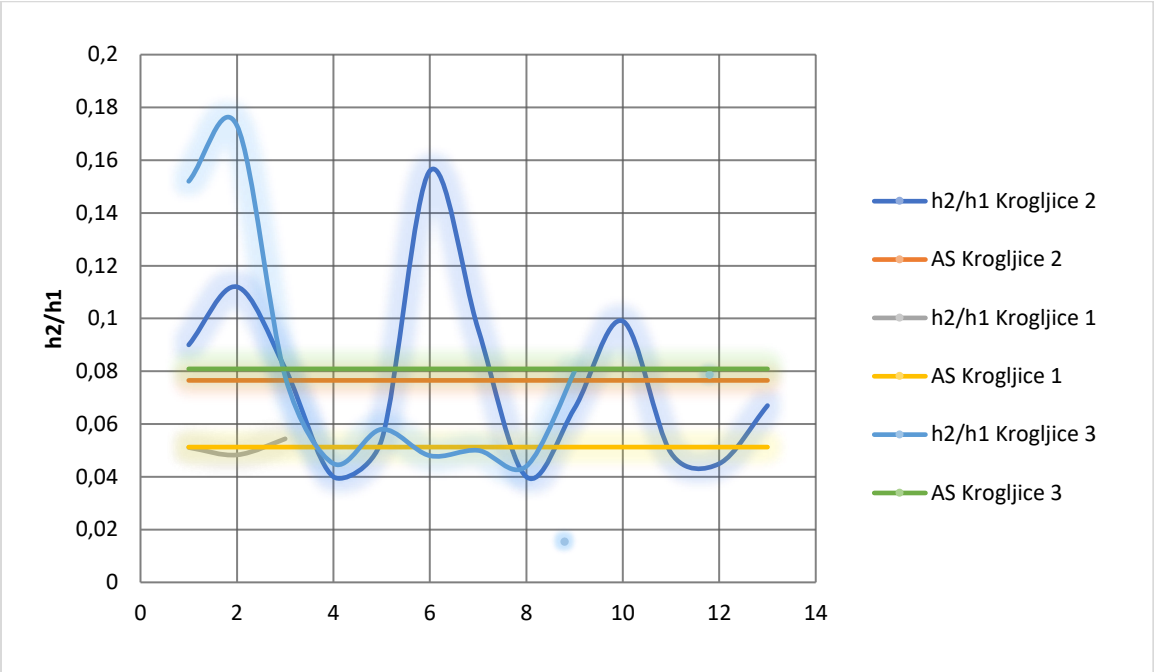
5.2.4 Razultati testiranja telesa kroglice 3

Indeks	h_1[cm]	h_2[cm]	$\frac{h_2}{h_1}$ [m]	h_t[cm]	Tip oklepajočega telesa	Ime telesa
1.	82	12,5	0,152	9,3	Čaša	Kroglica 3
2.	82	14,2	0,173	9,3	Čaša	Kroglica 3
3.	215	16,8	0,078	9,3	Čaša	Kroglica 3
4.	300	13,5	0,045	9,3	Čaša	Kroglica 3
5.	300	17,6	0,059	9,3	Čaša	Kroglica 3
6.	300	12,4	0,041	9,3	Čaša	Kroglica 3
7.	320	16,3	0,051	9,3	Čaša	Kroglica 3
8.	320	14.1	0,044	9,3	Čaša	Kroglica 3
aritmetična sredina			0,08			

Tabela 5 - Prikaz razultatov telesa kroglice 3.

Kroglice 3 so imele najbolj primerno obliko za podan vztrajnosti moment, kar pomeni, da so zaradi tega razultati tudi najboljši. Ob enem pa lahko opazimo, da se višina h_2 ne zviša drastično glede na h_1, kar se nam je zdelo dokaj čudno. Ugotovili smo, da se to zgodi zaradi ogromne sile trenja verige, ki ga ne opazimo pri ostalih verigah. Posledično ta veriga ni nikoli pospešila do končne hitrosti zaradi njene kratke dolžine.

5.2.5 Zadnje besede o rezultatih



Graf 3 - Prikaz vseh meritev. (AS predstavlja aritmetično sredino)

Kot lahko opazimo iz grafa 3, je razmerje $\frac{h_2}{h_1}$ močno odvisno od višine in mase kroglic, kar potrdi našo teorijo. Poleg tega oblika in masa kroglic močno vpliva na ostale zunanje sile, kar vpliva na končne rezulatate.

6 Zaključek

Naša razreštiev fizikalnega problema se je izkazala za uspešno, zanesljivo kot tudi natančno rešitev. Med raziskovanjem so se nam porodile mnoge nove zamisli, s katerimi bi lahko pojav še lažje ali zanesljivejše opisali.

Simulacija se je izkzala za zelo uporaben pripomočke pri preverjanju razultatov. Izvorna koda simulacije je kar se da prilagodljiva, kar pomeni, da lahko kdor koli z znanjem programskega jezika C++ prilagodi simulacijo.

Dobljene rezultate smo primerjali z našimi začetnimi hipotezami in ugotovili naslednje:

- Za začetek pojava verižne fontane ne potrebujemo začetne hitrosti. To hipotezo lahko brez težev takoj potrdimo in dokažemo z našimi opažanji in video posnetki, saj zadostuje samo masa lastne verige.

- Višina fontane je odvisna od razlike višine med začetkom prostega pada in tlemi. To hipotezo smo lahko potrdili s tako pomočjo teorije, kot razultatov. Pri formuli $y(s) = \frac{Y_z(1-\beta)}{1-\alpha-\beta} - \cos(90-\tau)\left|\frac{\alpha Y_z}{1-\alpha-\beta} - s\right|$ lahko opazimo, da v formulo vstopa spremenljivka Y_z, ki opisuje začetno višino pojava.

- Pojav verižne fontane se ne zgodi samo na verigah.

To hipotezo lahko tako potrdimo kot tudi ovržemo. Sicer do pojava verižne fontane lahko pride tudi na drugih telesih, ampak je ta pojav značilen predvsem zaradi oblike verige njenega vzrajnostnega momenta in sile napetosti.

- Masa in velikost kroglic na verigi vplivata na pojav verižne fontane.

Končna ugotovitev te hipoteze nas je presenetila, saj smo iz naših opažanj eksperimentov ugotovili, da lahko to hipotezo potrdimo. Oblike verige predvsem vpliva na vztrajnostni moment telesa, medtem ko masa, vpliva na spremenljivko β, ki nam določa višino verige.

7 Zahvale

Posebej se zahvaljujemo profesorici Simoni Granfol za podporo. Zahvaljujemo se tudi šolskemu mentorju Gregi Celcarju, ki nam je pomagal pri zbiranju podatkov in izvajanju poizkusov.

8 Viri

[1] https://en.wikipedia.org/wiki/Chain_fountain

[2] https://royalsocietypublishing.org/doi/10.1098/rspa.2013.0689

[3] https://en.wikipedia.org/wiki/DirectX

[4] https://sl.wikipedia.org/wiki/C%2B%2B

[5] https://sl.wikipedia.org/wiki/Integrirano_razvojno_okolje

[6] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[7] https://sl.wikipedia.org/wiki/Microsoft_Windows

[8] https://en.wikipedia.org/wiki/DirectX