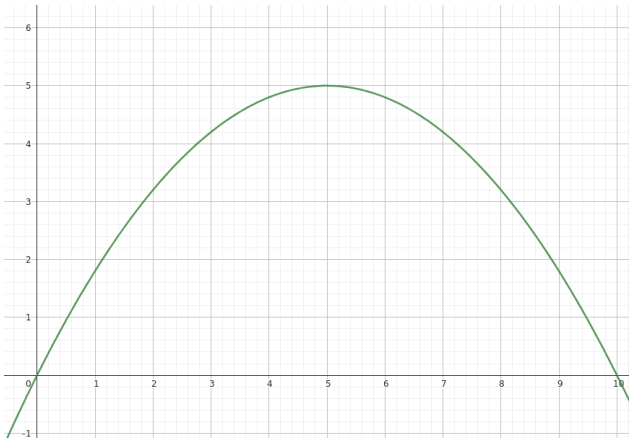
Kanon-kugler gennem luften

Kasteparabler i det 2-dimensionelle plan med forskellige beregninger, simulering og sammenligning heraf.

T. Pieter, M. Kongsted, S. Jakobsen

1 Indledningen

Når man kaster med sten eller skyder med kanoner, vil man observere at projektilen hverken vil følge en lige linje eller en tredjegradspolynomie, man vil rettere observere, at den ligner en parabel. Dette kaldes en kasteparabel. Med nogle kendte værdier og beregninger, kan man udregne forskellige værdier ud fra kasteparablen, så som højden, distancen eller hastigheden. Vi har lavet nogle eksperimenter med kanonen, og opstillet en digital simulering heraf, som vi vil regne på og sammenligne.



Skitseret eksempel på en parabel.

2 Teori

Følgende formel beskriver en kasteparabels forhold mellem hastighed, tid og strækning det i 2-dimensionelle plan:

$$y = \frac{1}{2} \cdot a \cdot t^2 + V_{0y} \cdot t$$

$$y[m] = \frac{1}{2} \cdot a[\frac{m}{s^2}] \cdot t^2[s^2] + V_{0y}[\frac{m}{s}] \cdot t[s]$$

Givet en modellering af en kasteparabel af et objekt i et 2-dimensionelt koordinatsystem med akserne x og y, beskriver denne formel forholdet mellem starthastigheden V_{0y}, tiden t og positionen på y-aksen.

For at beregne kuglens fart og hastigheder i parablen, kan man, givet to af, start hastigheden ad y-aksen v_{oy}, den totale hastighed v_o og kastevinklen θ, beregne den sidste. For at beregne hastigheden ud af x-aksen, kan man bruge den følgende formel med cosinus til vinklen, og sinus til hastigheden ud af y-aksen.

$$v_{0y} = v_0 \cdot \sin(\theta)$$

$$v_{0x}[\frac{m}{s}] = v_0[\frac{m}{s}] \cdot \sin(\theta)$$

$$v_{0x} = v_0 \cdot \cos(\theta)$$

$$v_{0x}[\frac{m}{s}] = v_0[\frac{m}{s}] \cdot \cos(\theta)$$

Følgende formel beskriver forholdet mellem strækningen ad x-aksen, starthastigheden ad y-aksen og tiden t:

$$s = v_{oy} \cdot t$$

$$s[m] = v_{oy}[\frac{m}{s}] \cdot t[s]$$

$$v_{oy} = \frac{s}{t}$$

$$t = \frac{s}{v_{oy}}$$

3 Eksempel på teori

I følgende eksempel, vil der blive vist, hvordan disse formler anvendes, med en kanonkugle skudt ud af en kanon.

I mange tilfælde er den mest signifikante kraft der påvirker objektet under et kast og skaber parablen tyngdekraften. Derfor antager man oftest, at den eneste påvirkende kraft er tyngdekraften. I kasteparablens formel, udskifter vi derfor accelerationen a, med tyngdeaccelerationen modsat y-aksen, da tyngdekraften, trækker en ned mod jorden. Vi kan regne tyngdekraften på følgende

måde, med objektets masse m og tyngdekraftkonstanten g, som vi antager er $9.82\frac{m}{s^2}$:

$$F_g = m \cdot g$$
$$= m \cdot 9.82[\frac{m}{s^2}]$$

Med kraften F_g, siger Newtons anden lov, at man man regne acceleration a, hvis man også har massen m, som vi kender.

$$a = \frac{F_g}{m}$$

Og hvis man udvider kraften F_g, kan man opnå følgende:

$$a = \frac{m \cdot 9.82[\frac{m}{s^2}]}{m}$$
$$\downarrow$$
$$a = 9{,}82[\frac{m}{s^2}]$$

Accelerationen a er indtil videre ikke retningsbestemt, men efter vi ved at tyngdeacceleration er lodret, kan vi stille følgende vektor:

$$\vec{a} = \begin{bmatrix} 0\frac{m}{s^2} \\ -9.82\frac{m}{s^2} \end{bmatrix}$$

Eftersom accelerationen i retningen af x-aksen $\vec{a}_x$ er lig 0, betragter vi kun accelerationen langs y-aksen. Vi kan nu stille følge formel:

$$y = \frac{1}{2} \cdot (-9.82[\frac{m}{s^2}]) \cdot t^2 + V_{0y} \cdot t$$

Kasteparablens strækning over x-aksen kan findes med denne formel, givet at kastet delvist er foregået over aksen, og man kender enten tiden t eller starthastigheden v_{0y}. Vi starter med at sætte formlen lig 0:

$$0 = \frac{1}{2} \cdot (-9.82[\frac{m}{s^2}]) \cdot t^2 + V_{0y} \cdot t$$

Ligningen har nu 2 ubekendte, derfor kan vi kun finde strækningen hvis vi kender en af de 2. Jeg vil derfor angive, at kuglen bliver skudt ud med en vinkel på $40°$ og en fart på $20\frac{m}{s}$. Farten er samlet for både fart ud af x- og y-aksen, og vi vil derfor finde den hastighed ad y-aksen v_{0y}, som vi gør med den relevante formel fra tidligere:

$$v_{0y} = v_0 \cdot \sin(\theta)$$
$$\downarrow$$
$$v_{0y} = 20[\frac{m}{s^2}] \cdot \sin(40°)$$

Vi kan bruge definitionen af starthastigheden ud af y-aksen v_{0y} i ligningen:

$$0 = \frac{1}{2} \cdot (-9.82[\frac{m}{s^2}]) \cdot t^2 + 20[\frac{m}{s^2}] \cdot \sin(40°) \cdot t$$
$$\downarrow$$
$$t_0 = 0\,s, t_1 = 2.618279468\,s$$

Af de 2 løsninger, er den første løningen t_0 lig 0, da vi i dette eksempel startede kastet i 0 på y-aksen. Den anden løsningen er tiden det tog for kuglen af flyve gennem koordinatsystemet i parablen og igen ramme y-aksen.

Med tiden kendt, kan man regne strækningen s med følgende formel fra tidligere:

$$s = v_{oy} \cdot t$$
$$\downarrow$$
$$s = 20[\frac{m}{s^2}] \cdot \sin(40°) \cdot t_1$$
$$\downarrow$$
$$s = 33.68207075\,m$$

Vi ved nu at kanonen har skudt kanonkuglen ca. 33.68 meter ad x-aksen.

For at tilføje til eksemplet, hæves kanonen. Dette kompenseres for i beregningen, ved at sætte parabelligningen lig den negative højde kanonen er hævet, og omvendt for sænkning af kanonen. Vi angiver at kanonen hæves $10\,m$, det vil aspejles sådan i ligningen:

$$y=\frac{1}{2}\cdot(-9.82[\frac{m}{s^2}])\cdot t^2+V_{0y}\cdot t$$
$$y=-10\,m$$
$$10[m]=\frac{1}{2}\cdot(-9.82[\frac{m}{s^2}])\cdot t^2+20[\frac{m}{s^2}]\cdot\sin(40°)\cdot t$$
$$\downarrow$$
$$t_0=-0{,}62748779\,m, t_1=3{,}24573626\,m$$

Vi kan også finde parablens højeste punkt. Til det starter vi med formlen for hastighed ad y-aksen:
$v_y=(-g)\cdot t+v_{0y}$

Parablens højeste punkt, givet at kurven har maximumpunkt i toppen, hvilket er guarenteret af tyngdeaccelerationen, vil være der, hvor polynomiets stigning vil skifte fra voksende til aftagende, altså stigningen er lig 0. Punktet kan derfor findes ved at sætte v_y i formlen lig 0:

$$0=(-g)\cdot t_{top}+v_{0y}$$

Med værdierne genbrugt fra tidligere, får vi:

$$0=(-9.82[\frac{m}{s^2}])\cdot t_{top}+20[\frac{m}{s^2}]\cdot\sin(40°)$$
$$\downarrow$$
$$t_{top}=1{,}309139734\,s$$

Ved det, kan man se, at kanonkuglen når sit højeste punkt ca. 1.3 sekunder inde i kastet. Nu kan vi bruge tiden, til at finde parablens maximale y-værdi, med parablems formel:

$$y_{max}=\frac{1}{2}\cdot(-g)\cdot t_{top}^2+V_{0y}\cdot t_{top}$$
$$\downarrow$$
$$y_{max}=\frac{1}{2}\cdot(-9.82[\frac{m}{s^2}])\cdot t_{top}^2+20[\frac{m}{s^2}]\cdot\sin(40°)\cdot t_{top}$$
$$\downarrow$$
$$y_{max}=8{,}414987910\,m$$

Vi har nu fundet ud af at kanonkuglens højeste punkt er på ca. $8{,}4$ meter.

4 Forsøg

Vores forsøg omhandler at observere stien som en kanonkugle tager gennem luften når den bliver affyret fra kanonen.

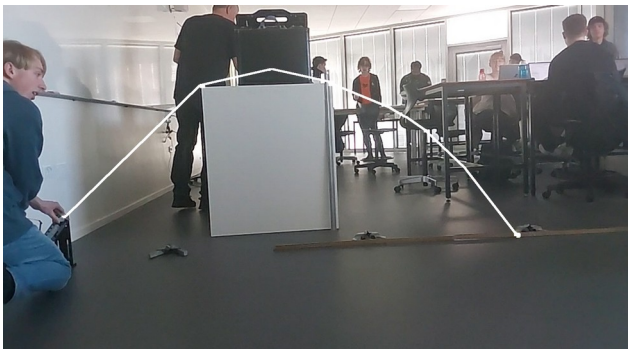
Til det har vi brugt en forsøgskanon og bold dertil, som vi har fyret af i et kontrolleret miljø, hvorefter vi måler strækning, tid og toppunkt.

Vi vil gerne se, om vores observerede tal fra forsøget matcher hvad vi ville kunne regne os til.

Vores fremgangsmetode var at stille en kanon med en kendt vinkel og højde for enden af 3 1-meters linealer, og med en lineal på højkant, til at aflæse højden. Vi havde oplevet at kanonen på laveste indstilling ikke skød mere end 3 meter. Vi affyrede kanonen og filmede det med en telefon. Derefter kigger vi på filmen og aflæste derfra de observerede værdier.

Vi observerede en strækning s på $2{,}33\,m$ og en tid på $1{,}416\,s$ før den ramte jorden. Kanonen stod i en vinkel på $45°$ og starthøjden h $0{,}25\,m$ fra jorden.

Kanonen ses i en 45-graders vinkel, med munden 25 cm over jorden.



Vores forsøg, med en 'traced' kasteparabel.

Vi vil starte med at regne starthastigheden v_{ox} ad den vandrette akse, x-aksen, ud fra strækningen s og tiden t:

$$s = v_{0x} \cdot t$$
$$\downarrow$$
$$v_{0x} = \frac{s}{t}$$
$$\downarrow$$
$$v_{0x} = \frac{2{,}33\,m}{1{,}416\,s}$$
$$\downarrow$$
$$v_{0x} = 1{,}645480226\,\frac{m}{s}$$

Derefter finde vi starthastighedens skalarværdi:

$$v_0 = \frac{v_{0x}}{\cos(\alpha)}$$
$$\downarrow$$
$$v_0 = 2{,}327060453\,\frac{m}{s}$$

Derefter bruger vi sinus, for at finde starthastighedens y-komponent v_{0y}:

$$v_{0y} = v_0 \cdot \sin(\alpha)$$
$$\downarrow$$
$$v_{oy} = 1{,}645480226\,\frac{m}{s}$$

Vi kender formlen for skrå kast:

$$y = \frac{1}{2} \cdot a \cdot t^2 + v_{0y} \cdot t + h$$

I vores tilfælde er vores acceleration tyngdeaccelerationen. Hvis vi sætter y lig 0 og løser for tiden t, finder vi hvor meget tid, det burde tage, for et skrå kast med en starthastighed v_0 på $2{,}32\,\frac{m}{s}$ ved $45\,°$ og $25\,cm$ starthøjde og tyngekraft.

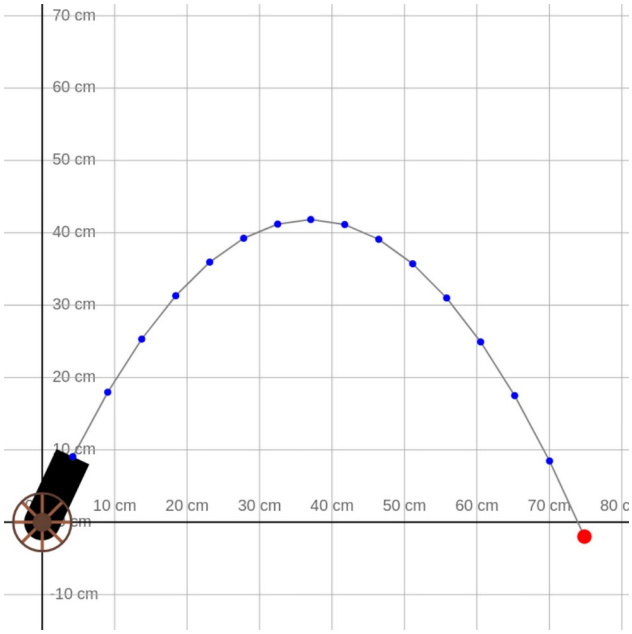
$$0 = \frac{1}{2} \cdot (-9.82[\frac{m}{s^2}]) \cdot t^2 + v_{0y} \cdot t + 0.25[m]$$
$$\downarrow$$
$$t_0 = -0{,}113495\,s, t_1 = 0{,}4486234\,s$$

Det regner vi så ud til at være $0{,}44$ sekunder, hvilket er en markant fejlmargin, på ca. 1 sekund, som ikke giver mening, da vi så skulle have observeret vores mål til at være langt fra hvad de var.

Vi tror at fejlen er opstået, fordi tiden blev målt, ved at tælle 'frames' i videoen, og hvis videoen har været afspillet med lavere hastighed, eller videoen har ændre mængden af 'frames' i sekundet.

5 Simulation

Vi har lavet et computer program, til grafisk simulering og visualisering af en vilkårlig kasteparabel.



En kørsel af vores simulator, med kanonen til højre justeret til $65\,°$*, kuglen landende på* x*-aksen ved ca.* $73\,cm$ *i rød, og kuglens fulgte parabel.*

Simulatoren består af en kanon og en kanonkugle på et kordinatsystem. Man kan justere på starthastighed og kanonens vinkel. Når man har indtastet sine ønskede værdier, trykker man på 'affyr'. En kugle vil nu i 'real time' flyve over skærmen, til den lander på x-aksen. Herved kan man aflæse parablens højde på y-aksen og dens strækning hen ad x-aksen. Siden simulationen kører i 'real time', aspejler kuglens flyvetid virkeligheden.

Simulatoren regner kuglens kasteparabel ud løbene, som kuglen flyver, dvs. for hvert 'frame' computeren applikationen renderer, regnes kuglens næste position.
Til beregningen af kuglens position p, skal vi kende kuglens hastighed v, hvilket også involvere kuglens acceleration a og masse m, de kræfter der påvirker den F og tiden siden sidste 'update' Δt.

Til beregning af kuglens acceleration a, bruger vi Newtons 2. lov, da vi kender kugles masse m og vi kan regne den resulterende kraft F på kuglen.

$$F = m \cdot a$$
$$\downarrow$$
$$a = \frac{m}{F}$$
$$a\left[\frac{m}{s^2}\right] = \frac{m[kg]}{F\left[N = \frac{m \cdot kg}{s^2}\right]}$$

Med accelerationen a og tidforskellen siden sidste 'update' Δt, kan vi regne kuglens nye hastighed v_1 med den tidligere hastighed v_0.

$$v_1 = v_0 + a \cdot \Delta t$$
$$v_1\left[\frac{m}{s}\right] = v_0\left[\frac{m}{s}\right] + a\left[\frac{m}{s^2}\right] \cdot \Delta t[s]$$

Med hastigheden, kan vi regne kuglens nye position p_1, ud fra den tidlere position p_0, accelerationen og igen tiden siden sidste 'update'.

$$p_1 = p_0 + v_1 \cdot \Delta t$$
$$p_1[m] = p_0[m] + v_1\left[\frac{m}{s}\right] \cdot \Delta t[s]$$

Disse beregninger bliver regnet igennem hvert 'frame', dvs. fx. 30 til 144 'frames' i sekundet 'FPS', på moderne hardware.

Simulatorens implementering af disse beregninger, ser sådan ud:

```
class Cannonball {
    mass: Kg;
    velocity: Vector2d<Meters>;
    position: Vector2d<Meters>,

    update(deltaTime: Seconds) {
        const drag = dragForce(
            sphereDragCoefficient,
            airDensity,
            SphereCrossSectionalArea(
                sphereRadius
            ),
            this.velocity,
        );
        this.velocity.add(
            acceleration([
                gravityForce(this.mass),
                drag,
            ], this.mass).extend(deltaTime),
        );
        this.position.add(
            this.velocity.extend(deltaTime),
        );
    }
}
```

Et redigeret kodeudsnit fra vores simulator.

```
function acceleration(
    forces: Vector2d<Newton>[],
    mass: Kilogram
): Vector2d<MetersPerSecond2> {
    return forces
        .reduce(
              (sum, force) =>
                  sum.add(force),
              new Vector2d(0, 0)
        )
        .divideComponents(mass);
}
```

Et redigeret kodeudsnit af "src/physics.ts" fra vores simulator.

Den resulterende kraft F består i vores simulation af tyngdekraft F_g og vindmodstand F_v. Vindmodstand kan vælges fra. Vi regner tyngdekraft med følgende formel:

$$F_g = m \cdot g$$

Vi bruger en realistisk tyngdekraftkonstant, så vi kan opnå resultater magen til vores andre beregninger, $9.82 \frac{m}{s^2}$. Implementeringen af dette, ser sådanne ud:

```
function gravityForce(
    mass: Kilogram,
): Vector2d<Newton> {
    const gravityConstant = 9.82;
    return new Vector2d(
        0,
        -gravityConstant * mass,
    );
}
```

Et redigeret kodeudsnit fra vores simulator.

Simulatoren har også evne til at simulere luftmodstand. Til dette bruger vi denne formlen for luftmodstand:

$$F_v = \frac{1}{2} \cdot c_w \cdot \rho \cdot A \cdot v^2$$

Denne formel beskriver forholdet mellem luftmodstanden, objektets formfaktor c_w, objektets tværsnitareal A, luftens densitet ρ og hastighedenforskellen mellem objektet og luften v. Siden vores objekt er en kuglerund kugle, vil objektets c_w-værdi det samme som en kugerund kugle; $0{,}5$. Tværsnitsarealet kan vi beregne, da vi kender radius for kanonkuglen. Densiteten angiver vi til at være sammenlignelig med luftens densitet i virkeligheden. Hastigheden varierer gennem kuglens forløb gennem kastet.

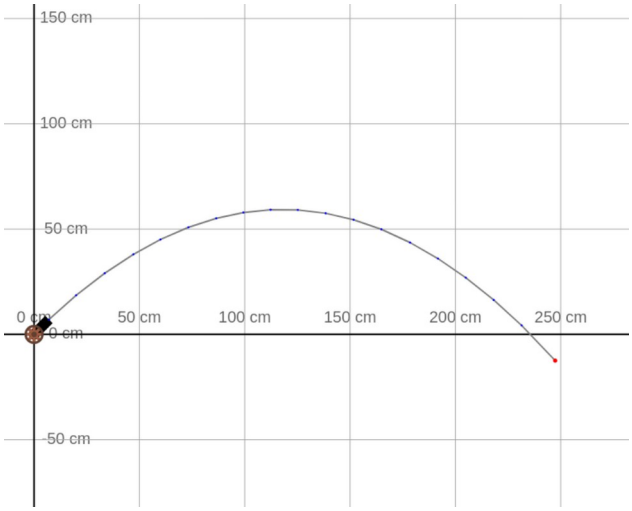
Luftmodstandsberegningen i vores simulation er implementeret sådan:

```
function sphereCrossSectionalArea(
    radius: Meters,
) {
    return Math.PI * radius ** 2;
}
const sphereDragCoefficient = 0.5;

/* approximate air density at 0c */
const airDensity = 1.3 KgPerMeters3;

function dragForce(
    dragCoefficient: number,
    fluidDensity: KilogramPerMeters3,
    area: Meters2,
    DeltaVelocity: Vector2d<MetersPerSecs>,
): Vector2d<Newton> {
    return deltaVelocity.clone()
        .raiseComponents(2)
        .extend(-0.5
            * dragCoefficient
            * fluidDensity
            * area,
         );
}
```

Et redigeret kodeudsnit fra vores simulator.

6 Sammenligning

Her er samme opsætning i vores simulator, som i vores forsøg:

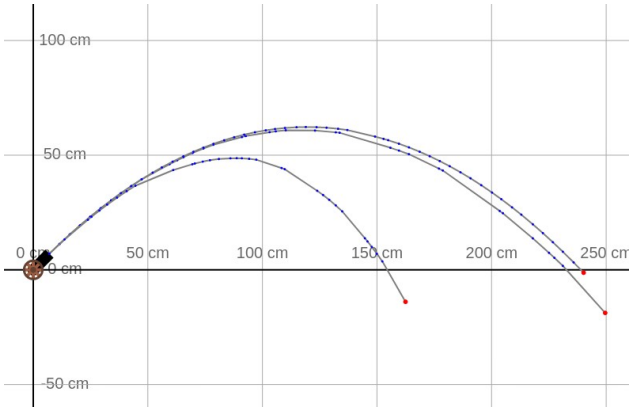


En kasteparabel fra vores simulator.

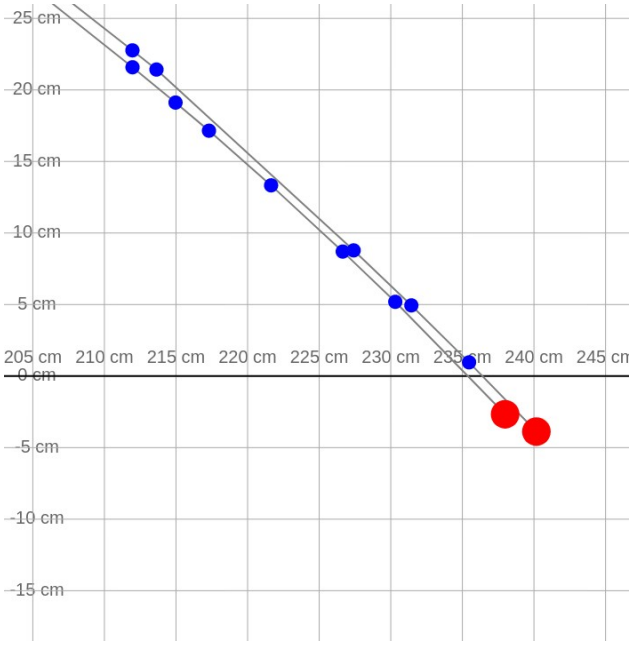
Kanonen har samme $45\,°$ vinkel, som i forsøget og kanonen er hævet ca. $25\,cm$ over x-aksen. Luftmodstand er slået fra. Med 'trial and error' fandt vi os frem til en starthastighed på $4{,}75\,\frac{m}{s}$, hvor kuglen ramte y-aksen samme sted, som i vores forsøg. I forsøget regnede vi os frem til en starthastighed på $2{,}32\,\frac{m}{s}$, hvilket er næsten præcist halvt så meget. Det vil tyde på en fejl, hvilket vi tror ligger i simulatoren, eftersom vi stoler på beregningerne i forsøget. Vi vil derfor, herfra, betragte den doppelte starthastighed i simulatoren som starthastigheden.

Vi timede tiden kastet ovenfor tog i simulationen. Det tog ca. $0{,}5$ sekunder. Det er meget tæt på de $0{,}44$ sekunder vi regnede det skulle tage i forsøget.

Vi prøvede at simulere forsøgsopstilling uden luftmodstand, med luftmodstand og med 10x luftmodstand. Vi observerede at kastet med og uden luftmodstand var så tæt på hinanden, at de nogle gange byttede plads på grund af upræcisioner i metoden vi simulerer.



Udvalgt simulation af forsøg unden, med og med 10x luftmodstand, med forskel på med og uden.



Simulation med og uden luftmodstand, hvor man kan se hvor lille forskel luftmodstanden gør.

Upræcisionen i overnstående simulering viser en af ulæmperne ved måden vi simulere på. Når man vi beregner, som vi gjorde i forsøget, får man det korrekte svar, uden videre. Men fordi simulationen regner i trin, istedet for at regne kastet som en helhed, bliver resultaten en konkatenering af approksimationer, som akkumulerer upræcisioner. Præcisionen afhænger derfor af hvor mange trin simulatoren kan bruge på en simulering, og fordi simulationen er ‘real time’ og endda også bliver renderet samtidig, afhænger mængden af trin merkant på computerens ydekapacitet, dvs. dræcisionen af simulationen afhænger af computerens ydeevne.

Tilgængen er simuleringsmetoden mere fleksibel end de numeriske beregninger på det punkt, at man nemt kan tilføje fx. andre krafter, som fx. luftmodstand.