Ujuja trajektoor külghoovuse puhul, kui suunda hoitakse teatud punktile.

■ Ülesande Püstitus

Ujuja ujub üle jõe laiusega *s*, ta ujumise kiirus on *vu*, jõe voolu kiirus on *vj*. Ujuja väljub punktist *A* ja suundub otse selle punkti vastas olevasse punkti *B*, mis asub punktist *A* samuti kaugusel *s*. Ujuja suundub kogu aeg punkti *B* suunas. Leida ujuja trajektoori võrrand, tema poolt läbitud tee pikkus ja trajektoori mööda ujumiseks kulunud aeg.

```
Remove["Global`*"](*See on kõige olulisem käsk, mis puhastab programmi mälu*)
```

Selleks, et liikumise võrrandid tuleksid võimalikult lihtsad, olgu lähtepunkti A koordinaadid (*s*; 0) ja ujuja liikugu punkti *B*(0; 0) suunas. Siis on suunavektor suunatud alati koordinaatide alguspunkti poole ja me saame seda esitada võimalikult lihtsal kujul. Jõgi voolaku *y*-teljega paralleelses suunas. Suvalisel ajahetkel olgu *x* olgu ujuja asukoha *x*-koordinaat ja *y* olgu ujuja asukoha *y*-koordinaat.

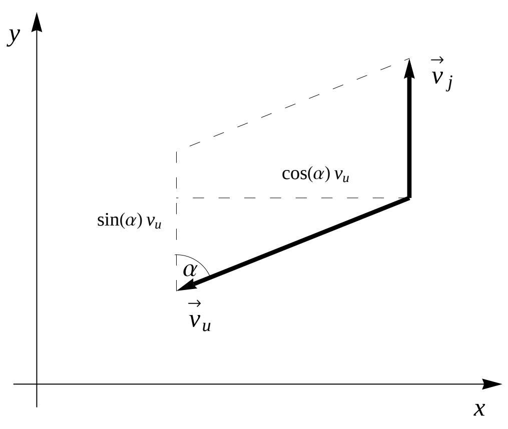
Need asukoha koordinaadid ja sõltuvad ajast *t* ja kahest kiirusest - jõe voolu kiirusest *vj* ning ujuja kiirusest paigalseisvas vees *vu*. Mõlemad kiirused on vektorid, mis on pikkuselt konstantsed. Jõe voolu kiiruse vektori suund on alati sama mis *y*-teljel, ujuja kiirusevektor on alati suunatud koordinaatide alguspunkti poole ja moodustab *x*-teljega vastupidise suunaga alati nurga α. Liikumise alghetkel on see nurk 0°, aga olenevalt jõe voolu kiiruse ja ujuja kiiruse suhtest võib see nurk suureneda kuni 90°-ni.

Ülesande üheks tulemuseks võiks olla ilmselt vektorväli, milles igale ujuja võimalikule asukohapunktile vastab ujuja kiirusevektor. Samas, üritaks ülesande lahendamisel vältida väljateooriat ja diferentsiaalvõrrandeid.
Esimeseks eesmärgiks on leida ujuja võimalike trajektooride võrrandid võimalikult lihtsal kujul.

Ülesannet saab küll ilmselt suhteliselt lihtsalt lahendada diferentsvõrrandite abil, aga siis on kursi ümberarvutused igast konkreetsest ujuja ja jõe kiiruse väärtusest lähtuvad (korraga saame kätte ainult ühe trajektoori!) ning trajektooriks oleva murdjoone kuju hakkab veidi sõltuma ka arvutussammu pikkuse (näiteks iga sekundi tagant, või hoopis iga 2 sekundi tagant) valikust.

Mudel muutuvate suurustega *x*, *y*, *t* ja α ning parameetritega *s*, *vj* ja *vu* :
y = *vj***t* - *vu**Sin[α]**t*
x = *s* - *vu**Cos[α]*t

Ujuja kiirusvektori vu lammutame kaheks komponendiks, üks neist on jõe voolusuuna ehk *y* - telje sihiline (*vu**Sin[α]), teine komponent on *x* - telje sihiline (*vu**Cos[α]).



Asukoha *y*- koordinaadi väärtust suurendab see, kui kaua jõgi on ujujat edasi kandnud (jõe voolu kiirusest vj ja ujumise aja t korrutis) ja vähendab ujuja kiirusvektori y-telje sihilise komponendi ja ujumise aja t korrutis. Asukoha x-koordinaadi algväärtus on s, seda hakkab vähendama ujuja kiirusvektori x-telje sihiline komponent.

Saadud võrrandisüsteem on küll füüsikaliselt ilus, aga see sisaldab nelja muutuvat suurust - nurka α, aega *t* ja kahte koordinaati *x* ja *y*. Neli muutuvat suurust kahes võrrandis on ilmselgelt liiga palju, hakkame nende suuruste arvu vähendama.

Kõigepealt kasutame ära meie poolt kasutusele võetud taustsüsteemi peamist eelist. selles taustsüsteemis on Sin[α] ja Cos[α] esitatatavad kujul:
Sin[α] $= \frac{y}{\sqrt{x^2+y^2}}$ ja Cos[α] $= \frac{x}{\sqrt{x^2+y^2}}$.

Nii vabaneme suurustest sin α ja cosα ja saame võrrandisüsteemi:

$$y == vj * t - vu * \frac{y}{\sqrt{x^2+y^2}} * t$$
$$x == s - vu * \frac{x}{\sqrt{x^2+y^2}} * t$$

Sellest kolme muutuja (x, y, t) ja kolme (s, vj, vu) parameetriga võrrandisüsteemist elimineerime ehk maakeeli kõrvaldame muutuja t.
Esmapilgul võib selline tegevus küll ka jabur tunduda, (sest me kaotame liikumisvõrranditest aja!), aga et meie eesmärgiks on saada kätte konstantsetest parameetritest s, vj ja vu sõltuvad trajektoorid ja meil on ainult kaks liikumisvõrrandit, siis on hea, kui meil jääb järgi ainult kaks tundmatut.

$$\textbf{Eliminate}\left[\left\{y == vj * t - vu * \frac{y}{\sqrt{x^2+y^2}} * t,\ x == s - vu * \frac{x}{\sqrt{x^2+y^2}} * t\right\},\ t\right]$$

$$-vj\ x + s\left(vj - \frac{vu\ y}{\sqrt{x^2+y^2}}\right) == 0\ \&\&\ x^2 + y^2 \neq 0$$

Lahendame võrrandisüsteemi *x* ja *y* suhtes :

$$\textbf{Solve}\left[-vj\ x + s\left(vj - \frac{vu\ y}{\sqrt{x^2+y^2}}\right) == 0\ \&\&\ x^2 + y^2 \neq 0,\ \{y, x\}\right]$$

Solve::svars : Equations may not give solutions for all "solve" variables. ≫

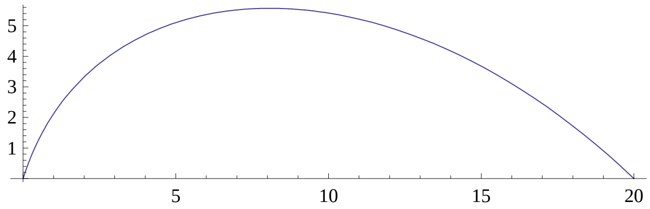
$$\left\{\left\{y \to -\frac{vj\ x\sqrt{-s^2 + 2\ s\ x - x^2}}{\sqrt{s^2\ vj^2 - s^2\ vu^2 - 2\ s\ vj^2\ x + vj^2\ x^2}}\right\},\ \left\{y \to \frac{vj\ x\sqrt{-s^2 + 2\ s\ x - x^2}}{\sqrt{s^2\ vj^2 - s^2\ vu^2 - 2\ s\ vj^2\ x + vj^2\ x^2}}\right\}\right\}$$

Selle võrrandisüsteemi kahest lahendist esimene on negatiivne lahend, selle võime kõrvale jätta. Tulemuseks saime liikumise graafiku, milles *y* sõltub muutuvast suurusest *x* ja parameetritest *s, vu* ja *vk.*

Anname parameetritele *s*, *vu* ja *vj* mingid algväärtused ja joonistame trajektoori graafiku nende algväärtuste puhul:

```
s := 20; vu := 2.01; vj := 2;
```

$$\text{Plot}\left[\left\{\frac{vj\,x\,\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2\,vj^2-s^2\,vu^2-2\,s\,vj^2\,x+vj^2\,x^2}}\right\},\ \{x,\,0,\,s\},\ \text{AspectRatio}\to\text{Automatic}\right]$$



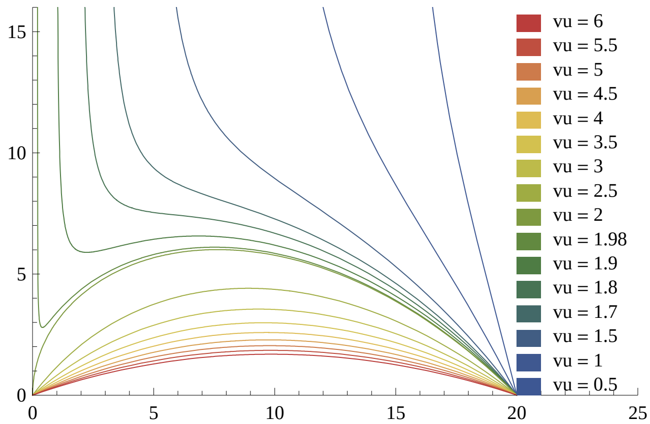
Defineerime ka nelja muutujaga funktsiooni, milles suurus *y* sõltub suurusest *x*, aga saab muuta ka parameetrite *s*, *vj* ja *vu* väärtusi.

$$F[s\_,\ vj\_,\ vu\_,\ x\_] := \frac{vj\,x\,\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2\,vj^2-s^2\,vu^2-2\,s\,vj^2\,x+vj^2\,x^2}};$$

Niiviisi parameetreid muutes saab joonistada välja terveid selliste joonte parvesid. Ilmselt on tark tegu jätta parameetrid jõe laius ja voolukiirus iga katse puhul muutumatuteks, ning varieerida parameetrit ujuja kiirus. Kui niiviisi teha, (näiteks *s* =*20* ja *vj*=2 puhul nagu alljärgnevas), siis saame terveid graafikute parvesid, mis kirjeldavad ujuja trajektoori sõltuvust tema ujumise kiirusest:

```
vuList = {0.5, 1, 1.5, 1.7, 1.8, 1.9, 1.98, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6};
jooned = F[20, 2, #, x] & /@ vuList;
cols = Table[ColorData["DarkRainbow"][i], {i, 0, 1, 1 / Length@vuList}];
legend = Table[{cols[[i]], Rectangle[{s, i - 1}, {s + 1, i - 0.3}],
    Black, Text["vu = " <> ToString[vuList[[i]]], {s + 1.5, i - 1}, {-1, -1}]},
   {i, 1, Length@vuList}];
```

```
Plot[jooned, {x, 0, s}, AspectRatio → Automatic, PlotRange → {{0, 25}, {0, 16}},
 PlotStyle → cols, Epilog → legend]
```



Vaatame funktsiooni

$$y = \frac{vj\,x\,\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2\,vj^2-s^2\,vu^2-2\,s\,vj^2\,x+vj^2\,x^2}}$$

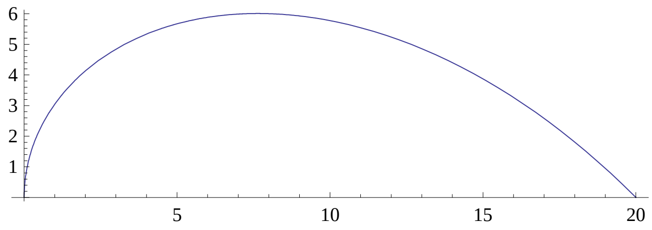
ja paneme tähele, et selle funktsiooni lugejas ja nimetajas on mõlemasjõe voolu kiirus ja ujumise kiirus esimeses astmes. Ruutjuur arvu ruudust tähendab ka esimest astet. Kui jagada selle funktsiooni paremal poole oleva avaldise lugejat ja nimetajat läbi vj - ga, ning tähistada seejärel $\frac{vu}{vj}=k$, siis oleme vähendanud parameetrite arvu kolmelt kahele. Suurus *k* tähendab nüüd arvu, mis näitab kui mitu korda on ujuja liikumise kiirus jõe voolu kiirusest suurem.Ilmneb, et liikumistrajektoori kuju ja võrrand ei sõltugi ujuja ujumiskiirusest vu ning jõe voolu kiirusest vj

$$y = \frac{x\,\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2-k^2\,s^2-2\,s\,x+x^2}}$$

Joonistame selle funktsiooni graafiku, kui s = 20 ja k = 1 :

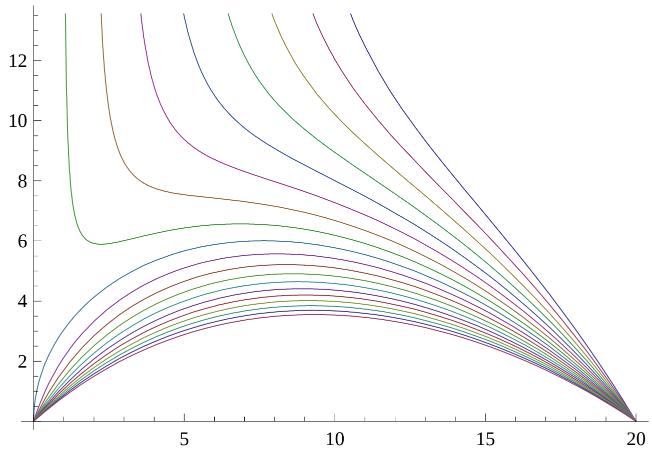
```
s = 20; k = 1;
```

$$\text{Plot}\left[\frac{x\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2-k^2\,s^2-2\,s\,x+x^2}},\ \{x,\ 0,\ s\},\ \text{AspectRatio}\to\text{Automatic}\right]$$



$$\text{funs} = \text{Table}\left[\frac{x\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2-k^2\,s^2-2\,s\,x+x^2}},\ \{k,\ 0.6,\ 1.5,\ 0.05\}\right];$$

```
Plot[funs, {x, 0, s}, AspectRatio → Automatic]
```



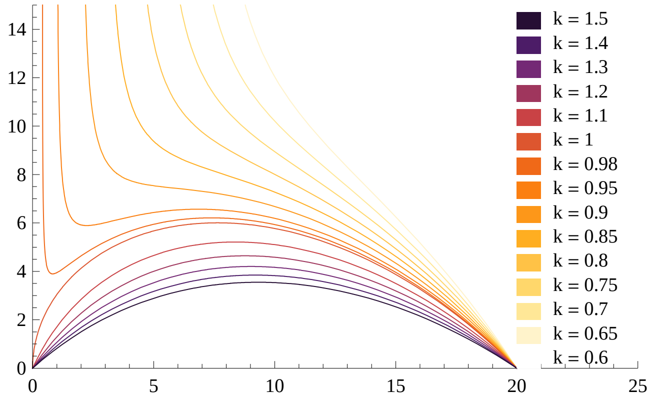
Vaatleme ka selle funktsiooni käitumist, kui k = 1 ja sellele lähedaste väärtuste puhul :

```
ki = {0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.98, 1, 1.1, 1.2, 1.3, 1.4, 1.5};
```

$$\text{funs} = \left(\frac{x\sqrt{-s^2+2\,s\,x-x^2}}{\sqrt{s^2-k^2\,s^2-2\,s\,x+x^2}}\ /.\ k\to\#\right)\ \&\ /@\ \text{ki};$$

```
cols = Reverse@Table[ColorData["SunsetColors"][i], {i, 0, 1, 1 / Length@ki}];
legend = Table[{cols[[i]], Rectangle[{s, i - 1}, {s + 1, i - 0.3}],
    Black, Text["k = " <> ToString[ki[[i]]], {s + 1.5, i - 1}, {-1, -1}]}, {i, 1, Length@ki}];
```

```
Plot[funs, {x, 0, s}, AspectRatio → Automatic,
 PlotRange → {{0, 25}, {0, 15}}, PlotStyle → cols, Epilog → legend]
```



Kui $k < 1$, siis ei jõua ujuja kunagi sihtpunkti ja ta teekond venib lõpmata pikaks. Kui $k >= 1$, siis on ujuja teekond lõplik ja selleks kuluva aja ning teepikkuse saab välja arvutada. Kui viimases funktsioonis

$$y = \frac{x\sqrt{-s^2 + 2 s x - x^2}}{\sqrt{s^2 - k^2 s^2 - 2 s x + x^2}}$$

võtta näiteks ülalolevas valemis k = 1 ja s = 20 (sel juhul saame maksimaalse pikkusega *lõpliku tee* juhul, kui s=20) ning kasutada seejärel üldtuntud kaarepikkuse arvutusvalemit

$$S = \int_{x1}^{x2} \sqrt{1 + (f'(x))^2}\, dx,$$

siis saaksime funktsiooniks $f(x) = \frac{x\sqrt{-400+40x-x^2}}{\sqrt{-40x+x^2}}$ ja selle tuletiseks

$$f'[x] = \frac{x\left(8000-1600x+80x^2-x^3\right)}{\sqrt{-(-20+x)^2}\ \left((-40+x)\,x\right)^{3/2}}$$

Viimase avaldise analüütilise integreerimisega tekkisid programmil Mathematica 6.0 teatavad raskused, seepärast proovisin numbrilist integreerimist :

$$\texttt{Smax20 = NIntegrate}\left[\sqrt{1 + \left(\frac{x\left(8000 - 1600x + 80x^2 - x^3\right)}{\sqrt{-(-20+x)^2}\ \left((-40+x)\,x\right)^{3/2}}\right)^2},\ \{x, 0, 20\}\right]$$

24.896

Kui näiteks võtta k = 4 ja s = 20 ning arvutada siis tuletis ja numbriliselt integreerida, siis saame tee pikkuse, mis on pikem kui otsetee 20 pikkusühikut, aga lühem, kui äsja leitud pikim lõplik tee (k=1, s=20 puhul):

$$\texttt{k = 4; s = 20; Tul2 = Simplify}\left[D\left[\frac{x\sqrt{-s^2 + 2 s x - x^2}}{\sqrt{s^2 - k^2 s^2 - 2 s x + x^2}},\ x\right]\right]$$

$$\frac{2\,400\,000 - 352\,000\,x + 10\,400\,x^2 + 80\,x^3 - x^4}{\sqrt{-(-20+x)^2}\ \left(-6000 - 40x + x^2\right)^{3/2}}$$

$$\texttt{NIntegrate}\left[\sqrt{1 + \left(\frac{2\,400\,000 - 352\,000\,x + 10\,400\,x^2 + 80\,x^3 - x^4}{\sqrt{-(-20+x)^2}\ \left(-6000 - 40x + x^2\right)^{3/2}}\right)^2},\ \{x, 0, 20\}\right]$$

20.2104

Kui aga võtta näiteks k = 0.5 ja s = 20 ning arvutada siis tuletis ja üritada arvutusvalemi järgi numbriliselt integreerida, siis need üritused ei kanna vilja, sest integraal ei anna lõplikku summat:

```
k = 1/2; s = 20; Tul2 = Simplify[D[x Sqrt[-s^2 + 2 s x - x^2] / Sqrt[s^2 - k^2 s^2 - 2 s x + x^2], x]]
```

$$\frac{-120000 + 26000\,x - 2200\,x^2 + 80\,x^3 - x^4}{\sqrt{-(-20+x)^2}\,\left(300 - 40\,x + x^2\right)^{3/2}}$$

```
NIntegrate[Sqrt[1 + ((-120000 + 26000 x - 2200 x^2 + 80 x^3 - x^4) / (Sqrt[-(-20 + x)^2] (300 - 40 x + x^2)^(3/2)))^2], {x, 0, 20}]
```

NIntegrate::slwcon :

Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration is 0, highly oscillatory integrand, or WorkingPrecision too small. ≫

Omaette küsimus on see, kui kaua selline ujumine aega võtab. Trajektoori pikkus on käes, ujuja ja vee voolu kiirusi teame me kah, ujumise aja välja arvutamine ei peaks olema raske. Paraku sõltub aga ujumise aeg sellest, kui suur osa trajektooril rakendatavast ujuja ujumiskiirusest tuleb kulutada vastuvoolu ujumisele.
Proovisin asja lihtsal tavapärasel viisil, võtsin ette esialgsed liikumisvõrrandid, sisestasin sinna parameetrid *s*, *vj* ja *vu* ning proovisin lahendada tekkinud võrrandisüsteemi, elimineerides muutujad *x* ja *y*. Paraku ei tahtnud selle võrrandisüsteemi lahendamine tavapärase Solve käsuga õnnestuda. Seetõttu proovisin numbrilise lahendamise käsku NSolve. Programm leidis süsteemile lahendeid, ja kohati olid need lahendid suhteliselt normaalsed, aga paraku esines ka juhtumeid, kus saadud lahendid läksid vastuolllu terve mõistusega. Ilmselt tasub ujumisaegade arvestuse puhul mõelda veel füüsikaliste kaalutluste peale ja mitte hakata võrrandisüsteemi vägisi murdma...

```
Remove["Global`*"]

s = 20; vj = 2; vu = 2;

NSolve[{y == vj * t - vu * y/Sqrt[x^2 + y^2] * t, x == s - vu * x/Sqrt[x^2 + y^2] * t}, t, {x, y}]
```

NSolve::infsolns :

Infinite solution set has dimension at least 1. Returning intersection of solutions with $t - \frac{85901\,x}{60742} + \frac{28373\,y}{30371} == 1.$

```
{{t → 7.18649}}
```

Kokkuvõte - ujumisaja määramise üle peab veel mõtlema.

Mudeli keerukamad, legendidega joonised ja vektorite joonise tegi: Maris Tõnso

Mudeli matemaatilis - füüsikaline pool ja lihtsam programmeerimine-arvutamine: Tõnu Tõnso
18.04.2014 kell 21.43.