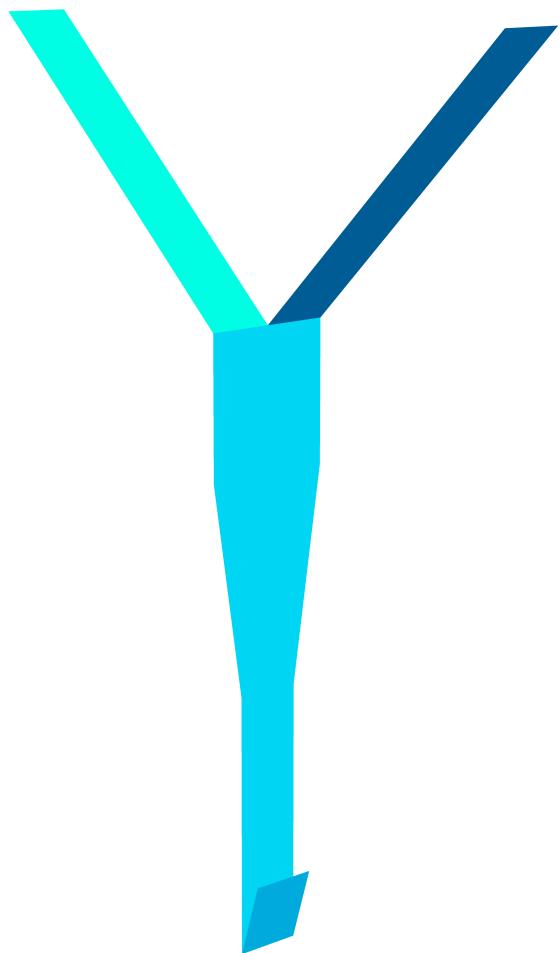


RAZISKOVALNA NALOGA

FIZIKALNI MODEL LETA PAPIRNIH LETAL

FIZIKA

2021



ZAHVALA

Zahvaljujem se mentorju za napotke, ki so mi bili v pomoč pri raziskovalni nalogi. Somentorici se zahvaljujem za nekaj slogovnih popravkov na začetku naloge. Zahvaljujem se tudi moji družini, ki me je podpirala in mi pomagala.

KAZALO

| | | |
|----------|---|-----------|
| 1 | POVZETEK | 5 |
| 2 | UVOD | 6 |
| 3 | TEORETIČNE OSNOVE SIMULACIJE LETAL | 7 |
| 3.1 | RAZDELITEV LETALA NA TRIKOTNIKE | 7 |
| 3.2 | MASA TRIKOTNIKA IN PLOSKEV LETALA | 10 |
| 3.3 | TEŽIŠČE TRIKOTNIKA IN PLOSKEV LETALA | 10 |
| 3.4 | VZTRAJNOSTNI MOMENT TRIKOTNIKA IN PLOSKVE LETALA | 11 |
| 3.5 | SILA ZRAKA NA TRIKOTNIK IN PLOSKEV LETALA | 14 |
| 3.6 | NAVOR ZRAKA NA TRIKOTNIK IN PLOSKEV LETALA | 16 |
| 3.7 | GIBANJE ENE PLOSKVE | 17 |
| 3.8 | SILE VZMETI MED PLOSKVAMI | 21 |
| 3.9 | NAVORI PREGIBOV MED PLOSKVAMI | 22 |
| 3.10 | GIBANJE LETALA | 24 |
| 3.11 | POPRAVEK Z RAVNANJIEM PLOSKEV | 27 |
| 3.12 | POPRAVEK Z OHRANITVIJO ENERGIJE | 27 |
| 3.13 | POPRAVEK Z DUŠENJEM VZMETI | 29 |
| 4 | SLEDENJE LETALOM IZ POSNETKA | 31 |
| 5 | PIRAMIDNO LETALO | 36 |
| 6 | EKSPEIMENTALNI DEL | 39 |
| 6.1 | SNEMALNI KOT KAMERE | 41 |
| 6.2 | REZULTATI ZA PIRAMIDNA LETALA | 42 |
| 6.3 | REZULTATI ZA HELIKOPTER LETALO | 44 |
| 6.4 | REZULTATI ZA W LETALO | 45 |
| 7 | RAZPRAVA IN ZAKLJUČEK | 46 |
| 8 | VIRI IN LITERATURA | 47 |
| 9 | PRILOGE | 48 |
| 9.1 | POENOSTAVLJENA ENAČBA ZA VZTRAJNOSTNI MOMENT TRIKOTNIKA V PROSTORU | 48 |
| 9.2 | MREŽE LETAL | 49 |
| 9.3 | PROGRAMI IN OSTALE PRILOGE | 50 |

KAZALO SLIK IN GRAFIK

| | | |
|----|--|----|
| 1 | Obračanje ploskve iz 3D v 2D | 7 |
| 2 | Večkotnik za razdelitev na trikotnike | 8 |
| 3 | Večkotnik razdeljen na trikotnike | 9 |
| 4 | Trikotnik | 10 |
| 5 | Trikotnik na koščke | 11 |
| 6 | Trikotnik na koščke v 2D | 12 |
| 7 | Sila zraka F_z | 14 |
| 8 | Koordinatni sistem ploskve | 17 |
| 9 | T-ploskev | 20 |
| 10 | Navor pregiba M_p | 23 |
| 11 | Poševno nihalo | 25 |
| 12 | Krčenje vzmeti | 29 |
| 13 | Vektorji RGB | 31 |
| 14 | Algoritem za pregledovanje pikslov | 32 |
| 15 | Sledenje gibanju letala v prostoru | 33 |
| 16 | Kamera in merjenje razdalje ΔY | 34 |
| 17 | Piramidno letalo | 36 |
| 18 | Sile na piramidno letalo | 37 |
| 19 | Irezana letala | 39 |
| 20 | Mehanizem za spuščanje papirnih letal | 40 |
| 21 | Merjenje snemalnega kota kamere θ_x | 41 |
| 22 | V_{max} meritve piramidno letalo | 42 |
| 23 | $z(t)$ meritve helikopter letalo | 44 |
| 24 | W letalo | 45 |

1 POVZETEK

Cilj raziskovalne naloge je bil fizikalno opisati gibanje papirnih letal. V tej raziskovalni nalogi je papirno letalo definirano kot poljuben predmet iz papirja.

V nalogi obravnavamo gibanje enoslojnih papirnih letal iz ravnih ploskev. Preprost primer je piramidno letalo. Za gibanje tega je izpeljana analitična rešitev.

Gibanje splošnega enoslojnega papirnega letala iz ravnih ploskev je sprogramirano kot fizikalna simulacija. Sprogramirano je tudi sledenje gibanju papirnih letal. To je kasneje uporabljeno pri analizi posnetkov leta.

Na koncu raziskovalne naloge je eksperimentalno preverjeno, kako dobro deluje model. Iz rezultatov je razvidno, da za nekatera letala deluje dobro, za druga pa ne deluje. Odvisno je od tega v kolikšni meri za posamezno letalo veljajo določene predpostavke. Programi so napisani v jeziku Wolfram Mathematica in dostopni na strani Github na naslovu <https://bit.ly/3a4x5Wn>.

KLJUČNE BESEDE

Papirna letala, sila curka, vztrajnostni moment, navor pregibov, simulacija, Wolfram Mathematica, analiza posnetka, piramidno letalo, helikopter letalo, W letalo

KEY WORDS

Paper planes, stream force, moment of inertia, bending torque, simulation, Wolfram Mathematica, video analysis, pyramid plane, helicopter plane, W plane

2 UVOD

Vedno sem se rad ukvarjal s papirnimi letali, saj so se mi zdela zabavna in fizikalno zanimiva. Na prvi pogled se zdijo papirna letala nekaj enostavnega, vendar pa so fizikalno zelo zapletena za obravnavo. Šele letos sem znal dovolj fizike, matematike in programiranja, da sem se lahko lotil fizikalnega opisa zelo preprostih papirnih letal.

Moj prvi cilj je bil splošna simulacija vsakega možnega papirnega letala, s pravim pretokom zraka. Naloga se je izkazala za neizvedljivo s srednješolskim znanjem. Zato sem se omejil na papirna letala iz enega sloja papirja, z ravnimi ploskvami in približkom mirujočega zraka okoli letala. S temi predpostavkami lahko nekatera letala opišemo presenetljivo dobro, drugih pa ne moremo. Cilji, ki sem si jih zastavil in jih tudi izpolnil so bili:

- Simulacija papirnih letal z zgornjimi predpostavkami
- Sprogramirano sledenje gibanju papirnih letal iz posnetka
- Eksperimentalno preverjanje modela

Programiral sem v jeziku Wolfram Mathematica, saj ima ta vgrajenih veliko matematičnih funkcij, funkcij za delo s seznamimi, simbolično računanje, izris grafik, skratka vse, kar sem za raziskovalno nalogu potreboval. Za enostavnejše programiranje sem uporabil tudi aplikacijo Touch Portal in makro gumbe na miški, v katere sem shranil ukaze (funkcije) in njihove pogoste kombinacije. Tudi vse skice v raziskovalni nalogi sem naredil v Mathematici. Za pisanje raziskovalne naloge sem uporabil program LaTeX.

Za simulacijo je bilo potrebnih več korakov. Najprej sem ugotovil algoritmom, ki papirno letalo razdeli na trikotnike. Nato sem izpeljal enačbe za maso, težišče in vztrajnostni moment posamezne ploskve papirnega letala ter silo in navor zraka nanjo. Potem sem izpeljal še enačbe za medsebojne sile in navore med ploskvami. Vse te elemente sem naposled povezal v simulacijo. Za boljše delovanje simulacije, sem raziskoval tudi tri različne popravke.

Za eksperimentalni del naloge sem izpeljal enačbe za sledenje gibanju papirnega letala iz posnetka, tudi prostorskemu, ki pa v praksi deluje le pri zelo dobrih pogojih. V eksperimentalnem delu sem na primeru treh vrst papirnih letal ugotavljal, kako dobro deluje simulacija.

Naloga je izvirna, saj so vsi obstoječi opisi papirnih letal, ki se jih je dalo najti le kvalitativni in ne fizikalni.

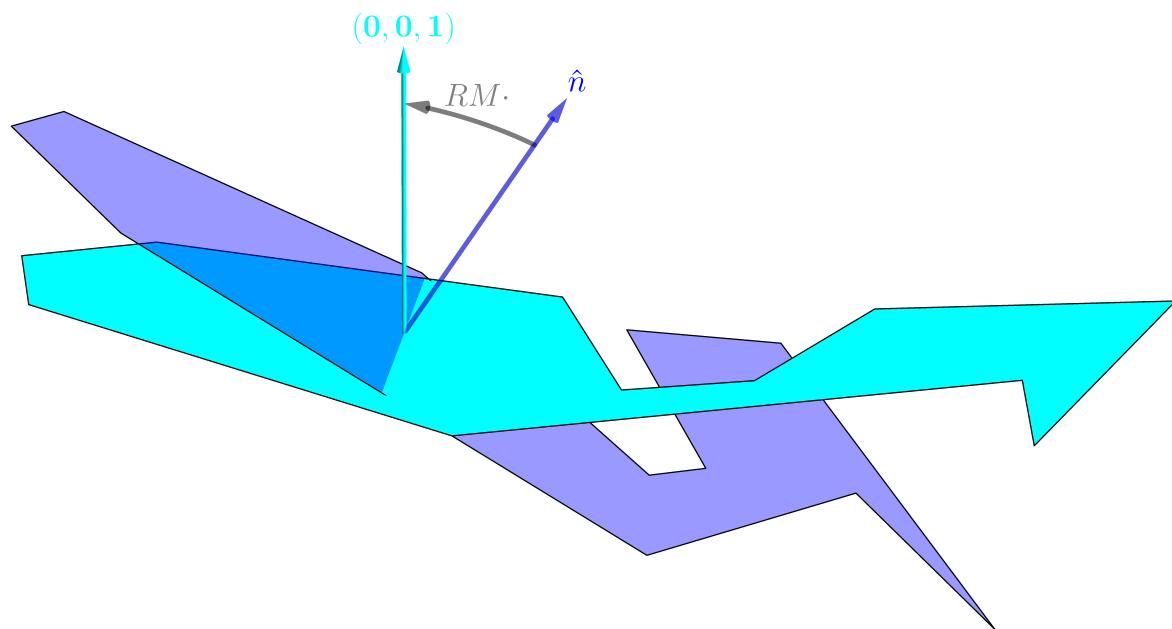
Z raziskovanjem bom nadaljeval, saj se da simulacijo izboljšati še na veliko načinov. Nekaj jih je naštetih v zaključku.

3 TEORETIČNE OSNOVE SIMULACIJE LETAL

3.1 RAZDELITEV LETALA NA TRIKOTNIKE

Ukvarjali se bomo s papirnimi letali, ki imajo ravne ploskve. Temu najbolj ustrezajo letala iz trdega papirja, ki letijo pri nizki hitrosti. Če je letalo sestavljeno iz samih ravnih ploskev, lahko vsako od teh razdelimo na trikotnike. Razdelitev nam bo v pomoč pri nadaljnji obravnavi letala.

Treba je ugotoviti algoritem, ki bo vsako ploskev razdelil na trikotnike. Ploskve bomo v simulaciji podali kot seznam zaporednih oglišč v prostoru. Lažje bi bilo, če bi bila ploskev podana v $2D$, zato moramo najprej spremeniti koordinate oglišč ploskev v $2D$. To lahko naredimo tako, da zasučemo ploskev v vodoravni položaj, z množenjem pozicijskih vektorjev oglišč ploskev z ustreznim rotacijskim matrikom. To je matrika transformacije, ki zavrti normalni vektor ploskev \hat{n} v vektor $(0, 0, 1)$ (glej sliko).

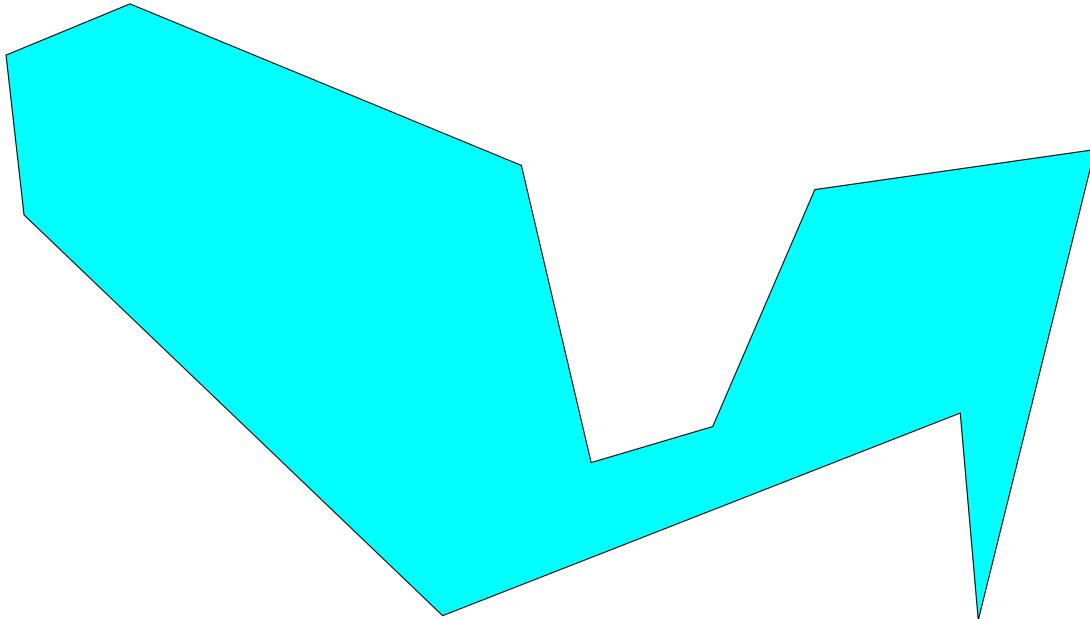


Slika 1: Obračanje ploskve iz $3D$ v $2D$

Wolfram Mathematica, v kateri so narejene simulacije, ima že vgrajeno funkcijo `RotationMatrix[u,v]`, ki z numeričnimi metodami poišče rotacijsko matriko transformacije, ki zavrti vektor u tako, da kaže v smeri vektorja v .

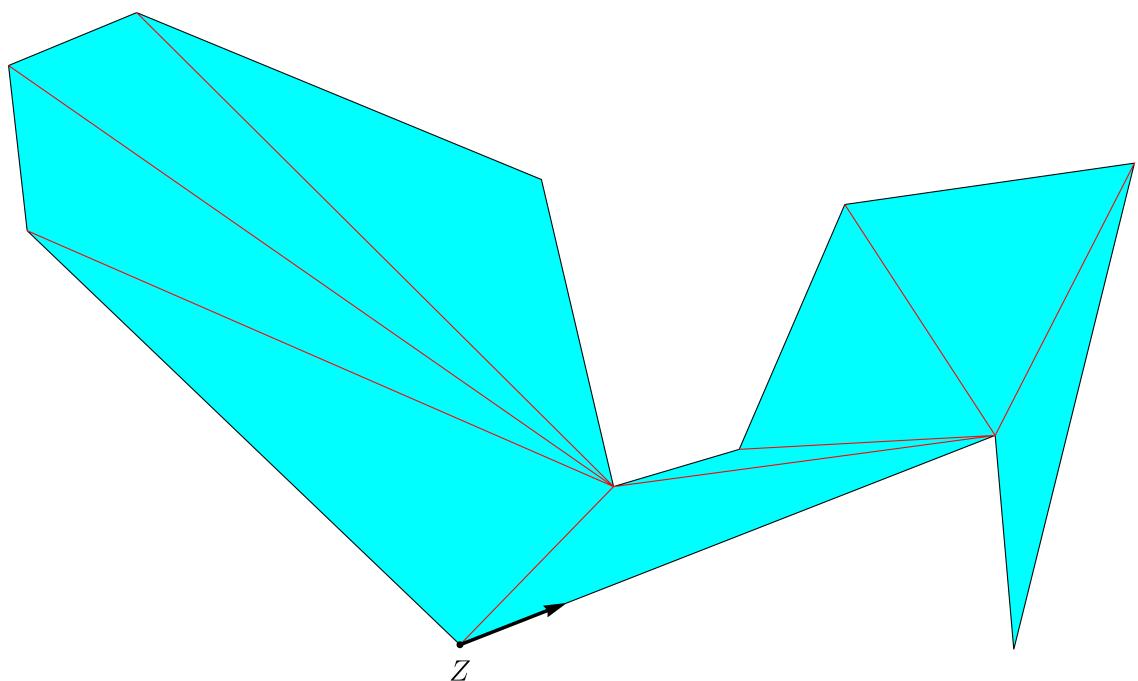
Po množenju oglišč z omenjeno matriko problem razdelitve večkotnika na trikotnike postane $2D$.

Če moramo razdeliti na trikotnike naslednji večkotnik in to poskusimo narediti s svinčnikom na papir, je naloga precej očitna.



Slika 2: Večkotnik za razdelitev na trikotnike

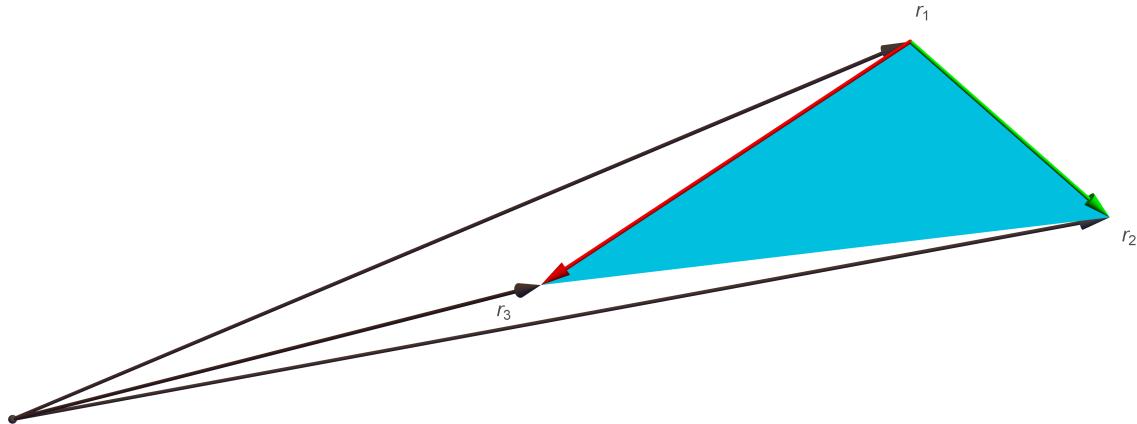
Če razmislimo, kako smo se odločali, kje bomo narisali črto, dobimo preprost algoritem. Zamislimo si, da stojimo v enem oglišču. V nasprotni smeri urinega kazalca gremo po eno oglišče naprej od oglišča do oglišča. Pri vsakem ugotovimo, ali prejšnje, to in naslednje oglišče tvorijo trikotnik, ki ga lahko odrežemo od večkotnika. Če v nekem oglišču zavijemo v desno, tega trikotnika ne moremo odrezati, saj je to "vdolbina" v večkotnik. Če zavijemo v levo, pa moramo še preveriti, ali morebitna povezava seka katero izmed ostalih stranic. Če je ne, lahko odrežemo ta trikotnik. S tem smo dobili nov večkotnik. Z dobljenim večkotnikom proces ponavljamo, dokler nam ne ostane trikotnik. Za vsak dobljeni trikotnik na koncu še ugotovimo indekse njegovih oglišč, med oglišči začetnega večkotnika in jih zberemo v seznam. Naslednja slika prikazuje razdelitev prejšnjega večkotnika na trikotnike. Črka Z na sliki označuje začetno oglišče.



Slika 3: Večkotnik razdeljen na trikotnike

3.2 MASA TRIKOTNIKA IN PLOSKEV LETALA

Trikotnik naj bo podan z oglišči \mathbf{r}_1 , \mathbf{r}_2 in \mathbf{r}_3 (glej sliko).



Slika 4: Trikotnik

Zeleni vektor lahko izrazimo kot $\mathbf{r}_2 - \mathbf{r}_1$, rdečega pa kot $\mathbf{r}_3 - \mathbf{r}_1$. Recimo, da je letalo iz papirja s ploščinsko gostoto ρ_p . Ta je natančno izmerjena in podana za vsak papir. Masa posamezne ploskve letala je vsota mas trikotnikov:

$$\sum_i \rho_p S_i = \sum_i \rho_p \frac{|(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})|}{2} = \frac{\rho_p}{2} \sum_i |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})|$$

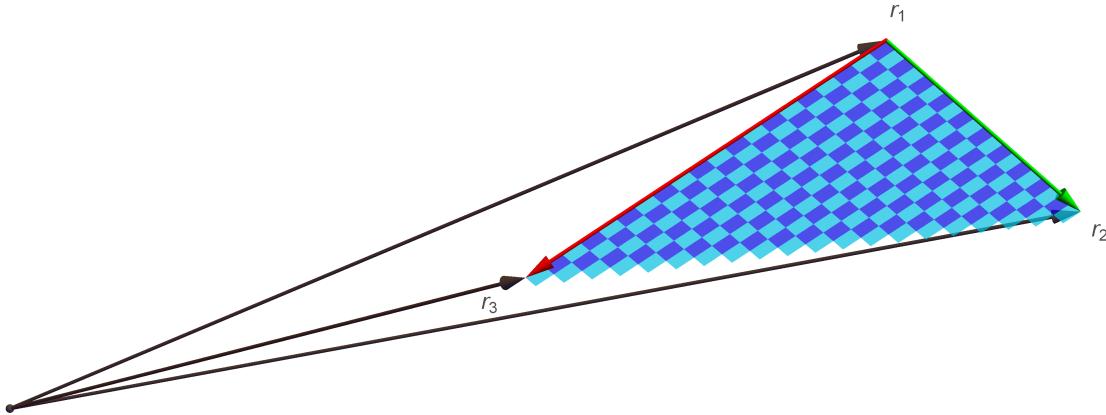
3.3 TEŽIŠČE TRIKOTNIKA IN PLOSKEV LETALA

Za izračun težišča uporabimo znano dejstvo, da je težišče trikotnika na povprečju njegovih oglišč $\mathbf{r}_* = \frac{\mathbf{r}_1 + \mathbf{r}_2 + \mathbf{r}_3}{3}$. Težišče celotne ploskve je zato

$$\mathbf{r}_* = \frac{\sum_i m_i \mathbf{r}_{*i}}{\sum_i m_i} = \frac{\sum_i |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})| (\mathbf{r}_{1i} + \mathbf{r}_{2i} + \mathbf{r}_{3i})}{3 \sum_i |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})|}$$

3.4 VZTRAJNOSTNI MOMENT TRIKOTNIKA IN PLOSKVE LETALA

Trikotnik lahko razdelimo na neskončno majhne koščke (paralelograme), kot kaže slika.



Slika 5: Trikotnik na koščke

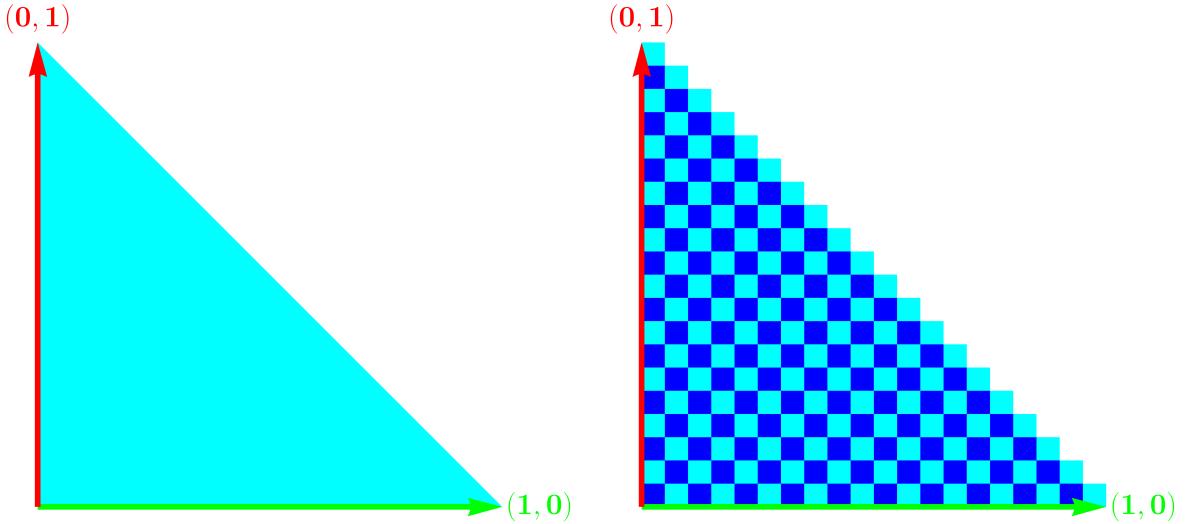
Recimo, da smo razdelili trikotnik na koščke z mrežo, sestavljeno iz $\frac{1}{dC_1}$ vrstic, vzporednih rdečemu vektorju in $\frac{1}{dC_2}$ zelenemu vektorju. Lego koščka glede na težišče ploskve r_* lahko opišemo z vektorjem:

$$\delta r(C_1, C_2) = \mathbf{r}_1 + C_1(\mathbf{r}_2 - \mathbf{r}_1) + C_2(\mathbf{r}_3 - \mathbf{r}_1) - \mathbf{r}_*$$

Kjer je:

- \mathbf{r}_* pozicijski vektor težišča ploskve, katere del je ta trikotnik
- C_1 delež zelenega vektorja
- C_2 delež rdečega vektorja

Če si v $2D$ predstavljamo, da je zeleni vektor $\hat{i} = (1, 0)$, rdeči pa $\hat{j} = (0, 1)$, se lahko hitro prepričamo, da za točke v trikotniku določenem z baznima vektorjema \hat{i} in \hat{j} velja $C_1 + C_2 \leq 1$. V tem primeru prejšnji predpis predstavlja točke med premico $y = -x + 1$ in koordinatnima osema. Lahko si predstavljamo transformacijo, ki trikotnik med baznima vektorjema v $2D$ spremeni v poljubni trikotnik. Ker sta C_1 in C_2 v splošnem deleža rdečega in zelenega vektorja, velja zveza tudi za poljubni trikotnik.



Slika 6: Trikotnik na koščke v 2D

Izračunajmo maso dm enega koščka. Ploščina kvadratka, ki ga oklepata \hat{i} in \hat{j} je 1. Ploščina paralelograma, ki ga v prostoru oklepata zeleni in rdeči vektor je zato $|(\mathbf{r}_2 - \mathbf{r}_1) \times (\mathbf{r}_3 - \mathbf{r}_1)|$ -krat večja od tiste v osnovnem 2D primeru. Ploščina koščka bi bila v osnovnem 2D primeru enaka $dC_1 dC_2$. Zato velja:

$$dm = \rho_p |(\mathbf{r}_2 - \mathbf{r}_1) \times (\mathbf{r}_3 - \mathbf{r}_1)| dC_1 dC_2$$

S tem lahko izrazimo vztrajnostni moment dJ enega koščka (glede na težišče ploskve \mathbf{r}_*):

$$dJ = \begin{bmatrix} (\delta\mathbf{r} \cdot \hat{y})^2 + (\delta\mathbf{r} \cdot \hat{z})^2 & -(\delta\mathbf{r} \cdot \hat{x})(\delta\mathbf{r} \cdot \hat{y}) & -(\delta\mathbf{r} \cdot \hat{x})(\delta\mathbf{r} \cdot \hat{z}) \\ -(\delta\mathbf{r} \cdot \hat{x})(\delta\mathbf{r} \cdot \hat{y}) & (\delta\mathbf{r} \cdot \hat{x})^2 + (\delta\mathbf{r} \cdot \hat{z})^2 & -(\delta\mathbf{r} \cdot \hat{y})(\delta\mathbf{r} \cdot \hat{z}) \\ -(\delta\mathbf{r} \cdot \hat{x})(\delta\mathbf{r} \cdot \hat{z}) & -(\delta\mathbf{r} \cdot \hat{y})(\delta\mathbf{r} \cdot \hat{z}) & (\delta\mathbf{r} \cdot \hat{x})^2 + (\delta\mathbf{r} \cdot \hat{y})^2 \end{bmatrix} dm$$

Vztrajnostni moment i -tega trikotnika ploskve (glede na njeno težišče \mathbf{r}_*) je vsota vztrajnostnih momentov vseh neskončno majhnih koščkov, na katere smo ga razdelili, torej integral:

$$\begin{aligned} J_i &= \int dJ = \int_0^1 \int_0^{1-C_1} \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} dm = \\ &= \rho_p |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})| \int_0^1 \int_0^{1-C_1} \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} dC_2 dC_1 \end{aligned}$$

Vztrajnostni moment ploskve (glede na njeno težišče \mathbf{r}_*) pa je vsota vztrajnostnih momentov trikotnikov, ki jo sestavljajo (glede na težišče ploskve \mathbf{r}_*):

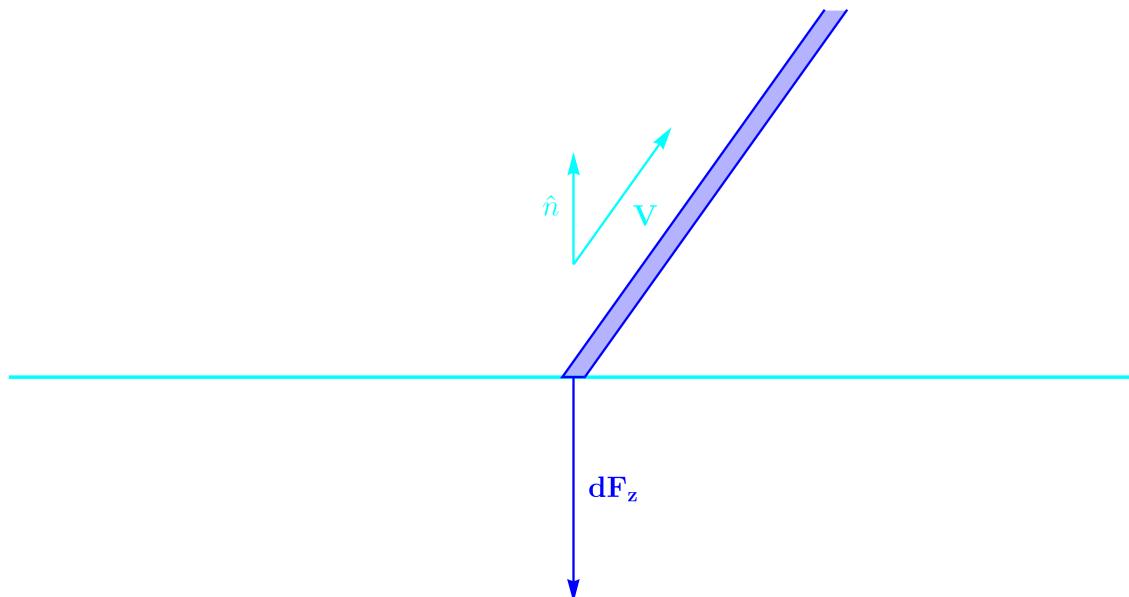
$$J = \sum_i J_i = \rho_p \sum_i |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})| \int_0^1 \int_0^{1-C_1} \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} dC_2 dC_1$$

Ko to računalnik integrira in poenostavi, dobimo zelo dolgo enačbo. Ta je med prilogami.

3.5 SILA ZRAKA NA TRIKOTNIK IN PLOSKV LETALA

Za izračun sile in navora zraka na trikotno ploskev moramo, kot smo to naredili že pri izračunu vztrajnostnega momenta, integrirati prispevke po celi ploskvi. Da lahko izračunamo te prispevke, moramo poznati tlak in gradient hitrosti zraka glede na ploskev, v vsaki točki na obeh straneh ploskve. Da bi to dvoje poznali, bi morali narediti simulacijo pretoka zraka okoli letala. Takšne simulacije se dela z numeričnim reševanjem Navier-Stokesovih enačb. To je zelo zapleteno in presega gimnazijsko znanje, poleg tega pa so take simulacije tudi zelo počasne na računalniku. Zato sem v raziskovalni nalogi vzel velik približek, da zrak okoli letala miruje in lahko silo in navor zraka izračunamo kot silo in navor curka zraka. Približek v veliko situacijah zadošča, saj je viskoznost zraka relativno majhna.

Recimo, da se točka na ploskvi premika s hitrostjo \mathbf{V} . Potem po približku mirajočega zraka, zrak na ploskev v tej točki vpada s hitrostjo $-\mathbf{V}$ (glej sliko).



Slika 7: Sila zraka F_z

Uporabili bomo enačbo za silo curka:

$$F_c = \rho S V^2$$

V našem primeru bo to

$$dF_z = \rho (\mathbf{V} \cdot \hat{n})^2 dS$$

Sila curka $d\mathbf{F}_z$ je vektor, usmerjen vzporedno vektorju \hat{n} , stran od \mathbf{V} . To z enačbo zapišemo tako:

$$d\mathbf{F}_z = \begin{cases} \mathbf{V} \cdot \hat{n} \leq 0, 1 \\ \mathbf{V} \cdot \hat{n} > 0, -1 \end{cases} \quad dF_z \hat{n} = \begin{cases} \mathbf{V} \cdot \hat{n} \leq 0, 1 \\ \mathbf{V} \cdot \hat{n} > 0, -1 \end{cases} \quad \rho (\mathbf{V} \cdot \hat{n})^2 \hat{n} dS$$

Hitrost \mathbf{V} gibanja točke na trikotni ploskvi pa je $\mathbf{V} = \mathbf{V}_* + \boldsymbol{\omega} \times \delta\mathbf{r}$, kjer $\delta\mathbf{r}$ izrazimo tako, kot prej pri izračunu vztrajnostnega momenta trikotnika:

$$\delta\mathbf{r} = C_1 (\mathbf{r}_2 - \mathbf{r}_1) + C_2 (\mathbf{r}_3 - \mathbf{r}_1) + \mathbf{r}_1 - \mathbf{r}_*$$

Tudi celotno silo zraka na i -ti trikotnik \mathbf{F}_{zi} izračunamo, kot smo vztrajnostni moment trikotnika:

$$\begin{aligned} \mathbf{F}_{zi} &= \int d\mathbf{F}_{zi} = \int_0^1 \int_0^{1-C_1} \begin{cases} \mathbf{V} \cdot \hat{n} \leq 0, 1 \\ \mathbf{V} \cdot \hat{n} > 0, -1 \end{cases} \rho (\mathbf{V} \cdot \hat{n})^2 \hat{n} dS = \\ &= \rho |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})| \hat{n} \\ &\quad \int_0^1 \int_0^{1-C_1} \begin{cases} \mathbf{V}(C_1, C_2) \cdot \hat{n} \leq 0, 1 \\ \mathbf{V}(C_1, C_2) \cdot \hat{n} > 0, -1 \end{cases} (\mathbf{V}(C_1, C_2) \cdot \hat{n})^2 dC_2 dC_1 \end{aligned}$$

Za ta integral ne poznamo analitične rešitve, lahko pa ga računalnik izračuna numerično. Ker je v simulaciji velikokrat izračunana sila upora \mathbf{F}_{zi} na trikotnike, to simulacijo zelo upočasnuje. Lahko pa vzamemo približek, pri katerem zanemarimo vrtenje trikotnika. To simulacijo pospeši in da rezultate, ki se v preizkušenih primerih ujemajo na tri mesta natančno. Če vzamemo ta približek, je hitrost $\mathbf{V}(C_1, C_2) = \mathbf{V}_*$ povsod na trikotniku. Zato je izraz za \mathbf{F}_{zi} tak, kot za $d\mathbf{F}_z$:

$$\mathbf{F}_{zi} = \begin{cases} \mathbf{V}_* \cdot \hat{n} \leq 0, 1 \\ \mathbf{V}_* \cdot \hat{n} > 0, -1 \end{cases} \quad \rho (\mathbf{V}_* \cdot \hat{n})^2 S \hat{n}$$

Kot že prej, je tudi sila zraka na ploskev enaka vsoti sil zraka na trikotnike, ki jo sestavljajo:

$$\mathbf{F}_z = \sum_i \mathbf{F}_{zi}$$

3.6 NAVOR ZRAKA NA TRIKOTNIK IN PLOSKEV LETALA

Navor zraka na i -ti trikotnik ploskve (glede na njeno težišče \mathbf{r}_*) \mathbf{M}_{zi} bomo izračunali na enak način, kot silo zraka \mathbf{F}_{zi} nanj:

$$\begin{aligned}\mathbf{M}_{zi} &= \int d\mathbf{M}_{zi} = \int \delta\mathbf{r} \times d\mathbf{F}_{zi} = \\ &= \rho |(\mathbf{r}_{2i} - \mathbf{r}_{1i}) \times (\mathbf{r}_{3i} - \mathbf{r}_{1i})| \\ &\int_0^1 \int_0^{1-C_1} \left(\begin{cases} \mathbf{V}(C_1, C_2) \cdot \hat{n} \leq 0, 1 & (\mathbf{V}(C_1, C_2) \cdot \hat{n})^2 \delta\mathbf{r}(C_1, C_2) \times \hat{n} \\ \mathbf{V}(C_1, C_2) \cdot \hat{n} > 0, -1 \end{cases} \right) dC_2 dC_1\end{aligned}$$

Tudi ta integral ni analitično rešljiv, ampak zanj nimamo dovolj dobrega približka. Zato bomo uporabili numerično integriranje. Da to simulacije ne bo preveč upočasnilo, bo program na novo izračunal \mathbf{M}_{zi} manjkrat na sekundo.

Kot že prej, je tudi navor zraka na ploskev (glede na njeno težišče \mathbf{r}_*) enak vsoti navorov zraka na trikotnike, ki jo sestavljajo (glede na težišče ploskve \mathbf{r}_*):

$$\mathbf{M}_z = \sum_i \mathbf{M}_{zi}$$

3.7 GIBANJE ENE PLOSKVE

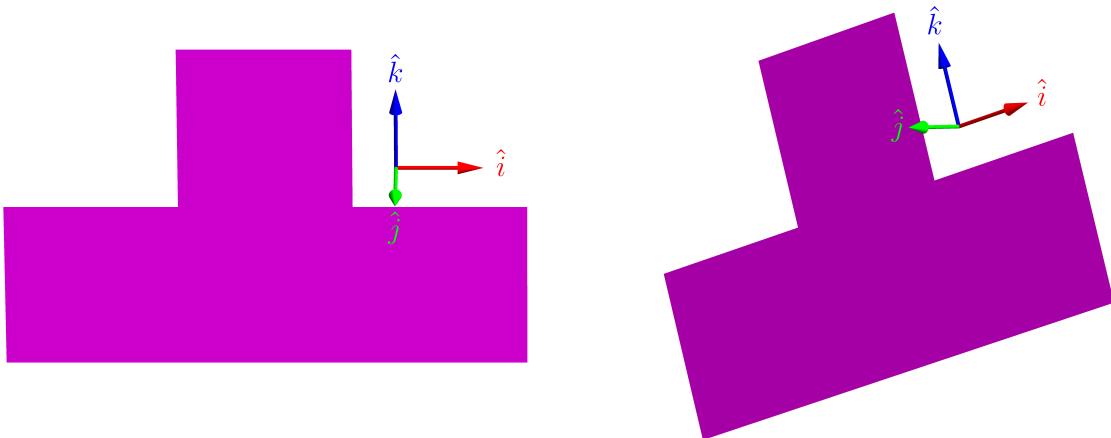
Gibanje ploskve lahko razdelimo na vrtenje in gibanje težišča (postopno gibanje). Na ploskev delujeta teža in sila zraka. Ta povzroči tudi navor.

Gibanje težišča ploskve lahko opišemo z 2. Newtonovim zakonom:

$$\ddot{\mathbf{r}}_* = \mathbf{g} + \frac{\mathbf{F}_z}{m}$$

Izraza za m in \mathbf{F}_z sta v prejšnjih poglavjih.

Opisati vrtenje je bolj zapleteno. Zamislimo si, da hkrati s ploskvijo vrtimo tudi začetne bazne vektorje. V koordinatnem sistemu zavrteneh vektorjev imajo oglišča ploskve vedno iste koordinate, če postavimo izhodišče v težišče ploskve. Posledično je v tem koordinatnem sistemu konstanten tudi vztrajnostni moment $J = J_0$. Poimenujmo ta koordinatni sistem koordinatni sistem ploskve (glej sliko).



Slika 8: Koordinatni sistem ploskve

Zavrtene začetne bazne vektorje lahko zberemo v stolpce matrike:

$$T = \begin{bmatrix} \hat{i}_1 & \hat{j}_1 & \hat{k}_1 \\ \hat{i}_2 & \hat{j}_2 & \hat{k}_2 \\ \hat{i}_3 & \hat{j}_3 & \hat{k}_3 \end{bmatrix}$$

T je tudi matrika transformacije, ki naš koordinatni sistem spremeni v ploskvinega. Vemo pa tudi, da je vztrajnostni moment J matrika transformacije, ki vektor ω spremeni v Γ . J_0^{-1} lahko pretvorimo v naš koordinatni sistem tako:

$$J^{-1} = TJ_0^{-1}T^{-1}$$

Dokaz za to je v viru [1]. Kotno hitrost ploskve ω lahko zato izrazimo z njeno vrtilno količino Γ :

$$\omega = TJ_0^{-1}T^{-1}\Gamma$$

Oglišča ploskve \mathbf{r}_i , bazni vektorji koordinatnega sistema ploskve $\hat{i}, \hat{j}, \hat{k}$ in normalni vektor ploskve \hat{n} se v času dt zavrtijo za kot ωdt okoli ω . To dosežemo z množenjem z ustrezeno rotacijsko matriko. Tudi za rotacijsko matriko transformacije vrtenja za kot Φ okoli vektorja w , ima Wolfram Mathematica vgrajeno funkcijo `RotationMatrix[\Phi,w]`. To bomo označili z $RM(\Phi, w)$. S tem lahko zapišemo naslednji izraz:

$$\hat{i}(t + dt) = RM(\omega(t) dt, \boldsymbol{\omega}(t)) \cdot \hat{i}(t)$$

Tak izraz bo prišel tudi pri \hat{j}, \hat{k} in \hat{n} .

Pri koordinatah oglišč \mathbf{r}_i je pomembno tudi gibanje težišča. Zato pred množenjem z RM premaknemo ploskev tako, da je težišče ploskve v koordinatnem izhodišču in po vrtenju nazaj, na koncu pa dodamo premik težišča (rotacijo opišemo glede na težišče):

$$\mathbf{r}_i(t + dt) = RM(\omega(t) dt, \boldsymbol{\omega}(t)) \cdot (\mathbf{r}_i(t) - \mathbf{r}_*(t)) + \mathbf{r}_*(t) + \mathbf{V}_*(t) dt$$

V zadnjih dveh enačbah ω izrazimo iz Γ po enačbi pred tem. Γ v simulaciji dobimo tako, da ga na začetku izračunamo kot

$$\Gamma_0 = J_0 \cdot \boldsymbol{\omega}_0$$

Potem ob vsakem časovnem koraku vrtilni količini, po izreku o vrtilni količini, prištejemo sunek navora:

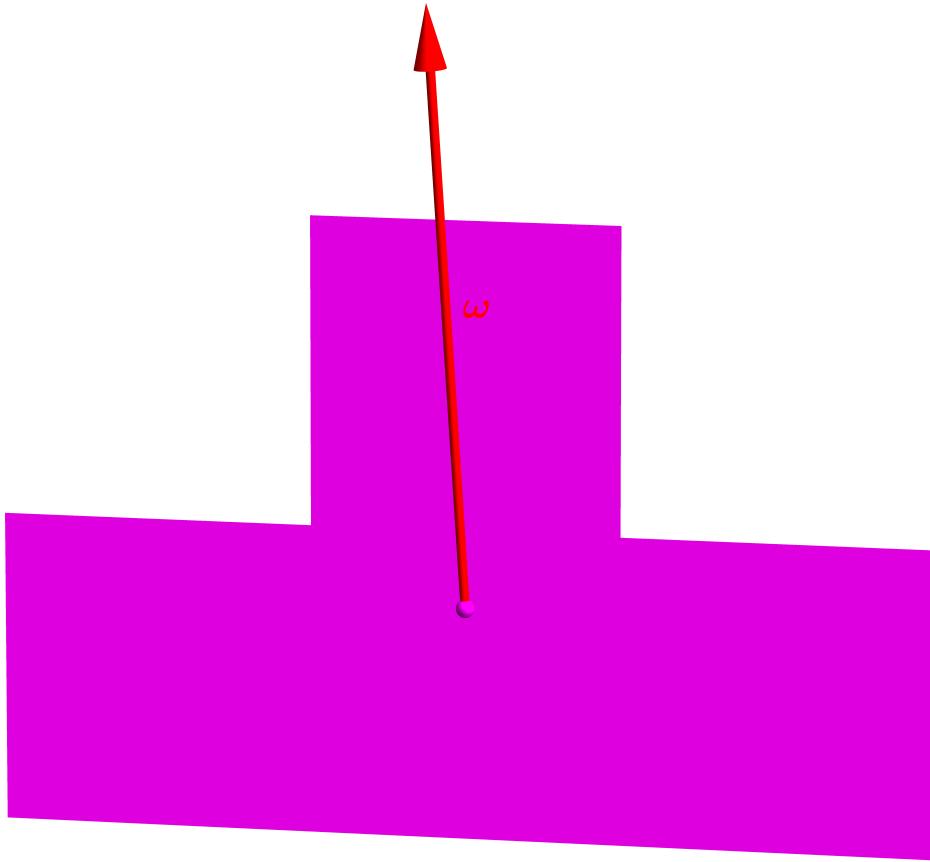
$$\Gamma(t + dt) = \Gamma(t) + \mathbf{M}_z(t) dt \iff \dot{\Gamma} = \mathbf{M}_z$$

Zdaj lahko povzamemo potek simulacije tako:

- Vnesemo začetne podatke:
 - Pozicijski vektorji oglišč ploskve \mathbf{r}_i
 - Hitrost težišča ploskve \mathbf{V}_*
 - Kotna hitrost ploskve ω
 - Ploščinska gostota papirja ρ_p
 - Gostota zraka ρ
 - Gravitacijski pospešek g
 - Časovni korak v simulaciji dt

- Program enkrat:
 - Razdeli ploskev na trikotnike
 - Izračuna ploščino S ploskve
 - Izračuna maso m ploskve
 - Izračuna težišče \mathbf{r}_* ploskve
 - Izračuna vztrajnostni moment J_0 ploskve
 - Izračuna začetno vrtilno količino Γ_0 ploskve
- Program ponavlja korake:
 - Zavrti in premakne oglišča \mathbf{r}_i ter zavrti bazne vektorje $\hat{i}, \hat{j}, \hat{k}$ koordinatnega sistema ploskve in normalni vektor ploskve \hat{n}
 - Premakne težišče \mathbf{r}_*
 - Izračuna silo zraka \mathbf{F}_z
 - Izračuna navor zraka \mathbf{M}_z
 - Izračuna pospešek težišča ploskve $\ddot{\mathbf{r}}_*$
 - Izračuna novo hitrost težišča ploskve \mathbf{V}_*
 - Izračuna novo vrtilno količino Γ
 - Izračuna novo kotno hitrost ω
 - Vsako $1/60\text{ s}$ v simulaciji izvozi sliko
 - Gre za dt naprej

Simulacijo lahko preizkusimo na preprostem primeru. Zamislimo si ploskev, oblikovano kot T-kocka v Tetrisu. Recimo, da je v breztežnostnem prostoru in, da na začetku kotna hitrost ω kaže malo stran od simetrijske osi ploskve (glej sliko).



Slika 9: T-ploskev

V viru [2] je razloženo, da v takem primeru pride do gibanja po učinku teniškega loparja. Če deluje pravilno, se mora to zgoditi tudi v simulaciji. Ko poženemo simulacijo, se zgodi naslednje:

- Če je razmerje ρ_p/ρ majhno, se ploskev hitro zaustavlja in gre proti vodoravnji legi. Da se v tej legi ploskev stabilneje vrati, lahko intuitivno pojasnimo tako: če je kotna hitrost približno pravokotna na ploskev, je navor zraka majhen, če pa se ploskev malo nagne, jo navor zraka potiska nazaj v to lego. To je le za predstavo, ni pa dokaz.
- Če je razmerje ρ_p/ρ zelo veliko, se dobro vidi učinek teniškega loparja. Pomembno je le, da je časovni korak dt v simulaciji dovolj majhen, sicer se rezultat hitro oddaljuje od pravilnega. Simulacijo bi se dalo izboljšati tako, da bi uporabljala RK-metodo višjega, npr. 4. reda. Potem bi bila simulacija z enakim časovnim korakom dt natančnejša.

3.8 SILE VZMETI MED PLOSKVAMI

Papirno letalo je sestavljeno iz več med seboj povezanih ploskev. Te med seboj druga na drugo delujejo s silami in navori. Ni očitno, kako bi lahko dobili sile med ploskvami. Nekaj podobnega je simulacija dvojnega fizičnega nihala, brez približka za majhne kote [4]. Že pri navadnem dvojnem nihalu se stvar silno zaplete. Pri papirnem letalu je tak problem v treh dimenzijah, poleg tega pa mora simulacija delovati za poljubno število poljubnih in poljubno povezanih ploskev. Brezupno bi bilo problem reševati na tak način. Zato si izmislimo enostavnejši način.

Predstavljammo si, da so ploskve med seboj v stikajočih se oglisčih povezane z namišljenimi vzmetmi, brez mase, osnovne dolžine 0. To seveda ni resnično in ni pravilno, ampak, ko gre prožnostni koeficient vzmeti k proti ∞ (v praksi veliki vrednosti), to v limiti pravilno opisuje gibanje ploskev papirnega letala. To je zato, ker gre v limiti razmaknjenost ploskev proti 0. Če to ne bi bilo res, bi imeli protislovje z ohranitvijo energije. To je zato, ker je prožnostna energija vzmeti, za katero velja Hookov zakon, če gre k proti ∞ enaka

$$\lim_{k \rightarrow \infty} W_{pr} = \lim_{k \rightarrow \infty} \frac{k \Delta l^2}{2}$$

Vidimo, da bi bil zgornji izraz enak ∞ , če Δl ne bi šel proti 0. Rezultat, da gre razmik med ploskvami proti 0, zagotavlja, da se ploskve držijo skupaj tudi ob predpostavki, da so ploskve povezane z opisanimi vzmetmi.

Sile in navore vzmeti na ploskve izračunamo po Hookovem zakonu:

$$\mathbf{F}_V = \sum_i \mathbf{F}_{Vi} = \sum_i k \delta \mathbf{r}_i = k \sum_i (\mathbf{r}_{i2} - \mathbf{r}_{i1})$$

$$\mathbf{M}_V = \sum_i \mathbf{M}_{Vi} = \sum_i \delta \mathbf{r}_i \times \delta \mathbf{r}_i k = k \sum_i (\mathbf{r}_{i1} - \mathbf{r}_*) \times (\mathbf{r}_{i2} - \mathbf{r}_{i1})$$

Kjer je:

- $\delta \mathbf{r}_i$ vektor med koncem na prvi in koncem na drugi ploskvi i -te vzmeti
- \mathbf{r}_{i1} konec i te vzmeti na prvi ploskvi
- \mathbf{r}_{i2} konec i te vzmeti na drugi ploskvi

3.9 NAVORI PREGIBOV MED PLOSKVAMI

V modelu manjkajo še navori pregibov M_p . Ti so odvisni od velikega števila dejavnikov:

- Dolžina pregiba (premo sorazmerje)
- Vrsta papirja
- Osnovni kot v pregibu
- Sprememba kota v pregibu
- “Razmajanost” pregiba
- Drugi dejavniki

Navori v pregibih so premajhni, da bi jih lahko natančno izmerili in prišli do kakih ugotovitev. Zato bomo za model poenostavljen predpostavili, da je navor v pregibu sorazmeren z dolžino pregiba in je poljubna funkcija spremembe kota med ploskvama v pregibu. Največkrat vzamemo kar linearni model s sučnim koeficientom D . Naj bo \mathbf{u}_p vektor od enega do drugega skupnega oglišča med stikajočima ploskvama. Da se pri programiranju ne bi preveč zapletlo, je narejeno samo za primere, pri katerih se sosednji ploskvi stikata v dveh ogliščih. Definirajmo funkcijo $\varphi(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$. Ta funkcija naj pove kot, za katerega moramo zavrteti vektor \mathbf{u}_1 , okoli vektorja \mathbf{u}_3 , po pravilu desne roke, da dobimo vektor \mathbf{u}_2 :

$$\varphi(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) = \begin{cases} (\mathbf{u}_1 \times \mathbf{u}_2) \cdot \mathbf{u}_3 \geq 0, \phi(\mathbf{u}_1, \mathbf{u}_2) \\ (\mathbf{u}_1 \times \mathbf{u}_2) \cdot \mathbf{u}_3 < 0, 2\pi - \phi(\mathbf{u}_1, \mathbf{u}_2) \end{cases}$$

Kjer je $\phi(\mathbf{u}_1, \mathbf{u}_2)$ manjši kot med vektorjema \mathbf{u}_1 in \mathbf{u}_2 . Za spremembo kota med ploskvama papirnega letala pa vzamemo razliko od osnovnega kota med ploskvama in omejimo spremembo kota med $-\pi$ in π . Predznak bo pomemben za smer navora med ploskvama.:

$$\Delta\varphi' = \varphi(\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \mathbf{u}_p, t = t_1) - (\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \mathbf{u}_p, t = 0)$$

$$\Delta\varphi = \begin{cases} \Delta\varphi' < -\pi, \Delta\varphi' + 2\pi \\ -\pi \leq \Delta\varphi' \leq \pi, \Delta\varphi' \\ \Delta\varphi' > \pi, \Delta\varphi' - 2\pi \end{cases}$$

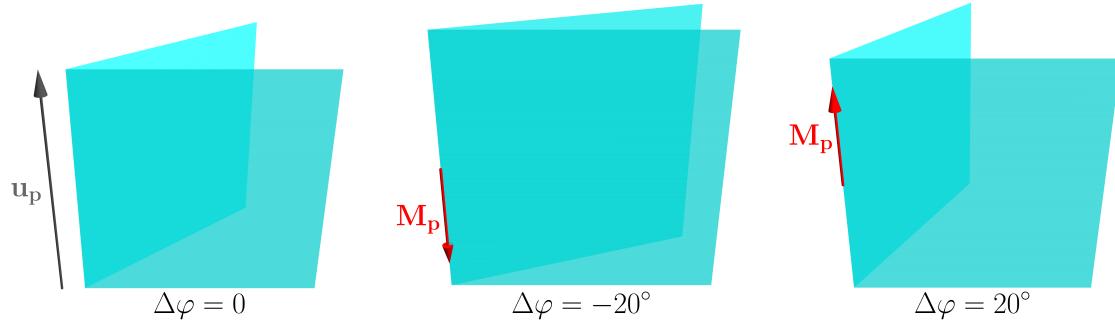
Iz že prej napisanega sledi izraz za navor pregiba na ploskev M_p :

$$M_p = u_p f(\Delta\varphi) \hat{u}_p = f(\Delta\varphi) \mathbf{u}_p$$

In linearni model za $f(\Delta\varphi) = D\Delta\varphi$:

$$M_p = D\Delta\varphi \mathbf{u}_p$$

Naslednja slika prikazuje navor pregiba M_p na sprednjo ploskev.



Slika 10: Navor pregiba M_p

3.10 GIBANJE LETALA

Gibanje papirnega letala bomo obravnavali z računanjem gibanja za vsako ploskev posebej. Zato lahko uporabimo večino razmislekov in izpeljav, ki smo jih naredili za gibanje ene same ploskve. V enačbi za gibanje težišča ploskve moramo upoštevati še sile vzmeti \mathbf{F}_v na ploskev:

$$\ddot{\mathbf{r}}_* = \mathbf{g} + \frac{\mathbf{F}_z + \mathbf{F}_v}{m}$$

Prav tako je treba v enačbi za vrtilno količino upoštevati še navore vzmeti M_v in navore pregibov M_p na ploskev:

$$\dot{\Gamma} = M_z + M_v + M_p$$

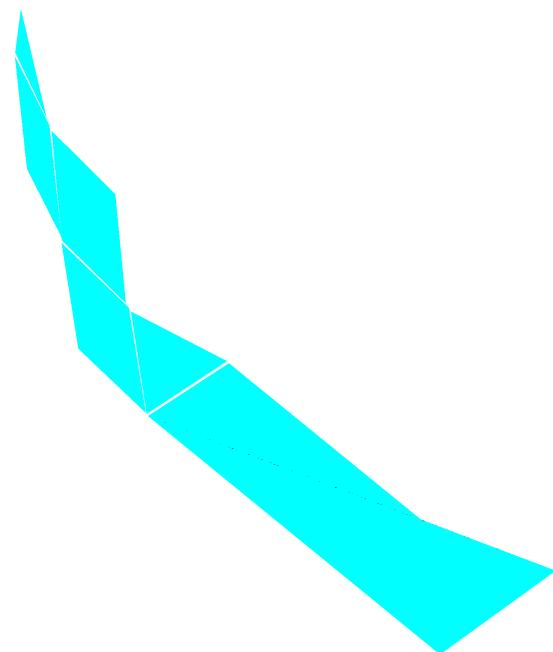
Vse ostale enačbe ostanejo nespremenjene. Tudi v simulaciji moramo dodati nekaj reči. Potek simulacije gibanja papirnega letala izgleda tako:

- Vnesemo začetne podatke:
 - Pozicijski vektorji oglišč ploskve \mathbf{r}_i
 - Hitrosti težišč ploskev \mathbf{V}_*
 - Kotne hitrosti ploskev ω
 - Prožnostni koeficient vzmeti k
 - Sučni koeficient pregibov D ali druga funkcija $f(\Delta\varphi)$
 - Ploščinska gostota papirja ρ_p
 - Gostota zraka ρ
 - Gravitacijski pospešek \mathbf{g}
 - Časovni korak v simulaciji dt
- Program enkrat:
 - Razdeli ploskve na trikotnike
 - Poišče indekse oglišč in ploskev, ki so med seboj povezane
 - Izračuna ploščine S ploskev
 - Izračuna mase m ploskev
 - Izračuna težišča \mathbf{r}_* ploskev
 - Izračuna vztrajnostne momente J_0 ploskev
 - Izračuna začetne vrtilne količine Γ_0 ploskev

- Program ponavlja korake:

- Zavrti in premakne oglišča r_i ter zavrti bazne vektorje $\hat{i}, \hat{j}, \hat{k}$ koordinatnih sistemov ploskev in normalne vektorje ploskev \hat{n}
- Premakne težišča r_*
- Izračuna sile zraka F_z
- Izračuna navore zraka M_z
- Izračuna sile vzmeti F_v
- Izračuna navore vzmeti M_v
- Izračuna navore pregibov M_p
- Izračuna pospeške težišč ploskev \ddot{r}_*
- Izračuna nove hitrosti težišč ploskev V_*
- Izračuna nove vrtilne količine Γ
- Izračuna nove kotne hitrosti ω
- Vsako $1/60\text{ s}$ v simulaciji izvozi sliko
- Gre za dt naprej

Simulacijo bomo najprej preizkusili na preprostem primeru. Imamo 6 ploskev, ki so vezane v nekakšno postrani šesterno fizično nihalo (glej sliko).



Slika 11: Poševno nihalo

Pri tem je zgornji rob vrtljivo vpet. Za to je v program vključeno, da lahko na začetku vnesemo tudi katera oglšča bodo vpeta na svoje začetne položaje. Sile na vpeta oglšča računamo tako, kot sile med ploskvami. Vpeti rob bi lahko uporabili tudi, če bi nas zanimalo, kako se zaradi teže spremenijo koti med ploskvami letala pred spustom. Še ena možnost uporabe bi bila za simulirati, kako se spreminjajo koti med ploskvami ob metu papirnega letala. Pri zadnjem bi se pritrdišča premikala po krivulji meta, nato pa izginila.

Če poženemo simulacijo, vidimo, da se ob dovolj majhnem časovnem koraku dt prejšnje "nihalo" giblje tako, kot bi pričakovali. Če pa je dt prevelik, gibanje ploskev hitro podivja in razdalje med njimi grejo proti neskončno. Do tega verjetno pride, ker se oglšča vsakič premaknejo malo predaleč. Če na primer začnejo s hitrostjo $V(t)$ in se premaknejo za $V(t) dt$, V pa se manjša, je povprečna hitrost malo manjša kot $V(t)$. Vzmet je tako vsakič malo bolj raztegnjena in ploskve hitro grejo neskončno narazen. O tem lahko premislimo na najlažjem primeru (navadno vzmetno nihalo). Pri navadnem vzmetnem nihalu je število potrebnih korakov na sekundo, da simulacija ne divergira, razmerno \sqrt{k} (fiksno število korakov na nihaj $t_0 = 2\pi\sqrt{\frac{m}{k}}$). Če preizkusimo različne vrednosti k , vidimo, da se potrebno število korakov na sekundo res veča približno razmerno \sqrt{k} .

Težava je v tem, da za boljšo natančnost simulacije, potrebujemo večjo vrednost prožnostnega koeficienta k . Če pa izberemo večji k , moramo tudi zmanjšati časovni korak v simulaciji in jo s tem upočasnimo. En način, kako bi se dalo izboljšati simulacijo, da bi zadoščal že večji časovni korak dt , bi bil z uvedbo RK-metode višjega reda. To v simulacijo ni vključeno, so pa v naslednjih poglavjih opisani še trije različni načini popravkov.

3.11 POPRAVEK Z RAVNANJEM PLOSKEV

Ta popravek bi se ponovil le na vsakih nekaj časovnih korakov, zato simulacije praktično ne bi upočasnil. Popravek deluje tako: najprej poiščemo najmanj razmagnjeni oglišči, ki bi se mogli stikati. Nato premaknemo eno ploskev tako, da se ti dve oglišči stikata. Potem ploskev zavrtimo da se stikata še drugi dve skupni oglišči. To ponovimo za vse ploskve. Kotne hitrosti ploskev in hitrosti težišč popravimo tako, da za razliko kotnih hitrosti sosednjih ploskev vzamemo samo komponento, vzporedno pregibu.

Ta popravek je le možna ideja, saj v simulaciji ni dokončan. Izkaže se, da se programiranje tega popravka kljub temu, da zveni enostavno, zelo zaplete.

3.12 POPRAVEK Z OHRANITVIJO ENERGIJE

Ta popravek temelji na tem, da sproti računamo energije in delo zraka. Nato hitrosti težišč V_* in kotne hitrosti ω pomnožimo s takim faktorjem α , da velja energijski zakon. Če v enačbo zberemo prisotne oblike energije in delo dobimo:

$$W_{p1} + W_{pr1} + W_{M1} + W_{k1} + W_{kr1} = W_{p0} + W_{k0} + W_{kr0} - A \iff$$

$$W_{k1} + W_{kr1} = W_{p0} + W_{k0} + W_{kr0} - W_{p1} - W_{pr1} - W_{M1} - A$$

Kjer je:

- W_{p0} začetna potencialna energija
- W_{k0} začetna kinetična energija
- W_{kr0} začetna rotacijska kinetična energija
- A delo, ki ga opravi zrak
- W_{p1} potencialna energija
- W_{pr1} prožnostna energija vzmeti
- W_{M1} kotni ekvivalent prožnostne energije, v pregibih zaradi navora
- W_{k1} kinetična energija
- W_{kr1} rotacijska kinetična energija

Na levi strani prejšnje enačbe imamo $W_{k1} + W_{kr1}$, kar je na desni strani izraženo z drugimi energijami. Recimo, da bi v simulaciji dobili preveliko hitrost ali pa kotno hitrost. Potem bi bila številska vrednost, ki bi jo dobili z vstavljanjem podatkov, na levi strani večja kot na desni. Recimo, da vzamemo podatke na desni strani za prave. Da bi enačba držala, bi morali zmanjšati levo stran. To lahko naredimo tako, da hitrosti in kotne hitrosti pomnožimo s faktorjem α . Ker tako hitrost, kot tudi kotna hitrost v izrazu $W_{k1} + W_{kr1}$ nastopata, kot kvadrata, lahko iz izraza izpostavimo α^2 . Levo stran enačbe zamenjamo z $\alpha^2 (W_{k1} + W_{kr1})$ in dobimo

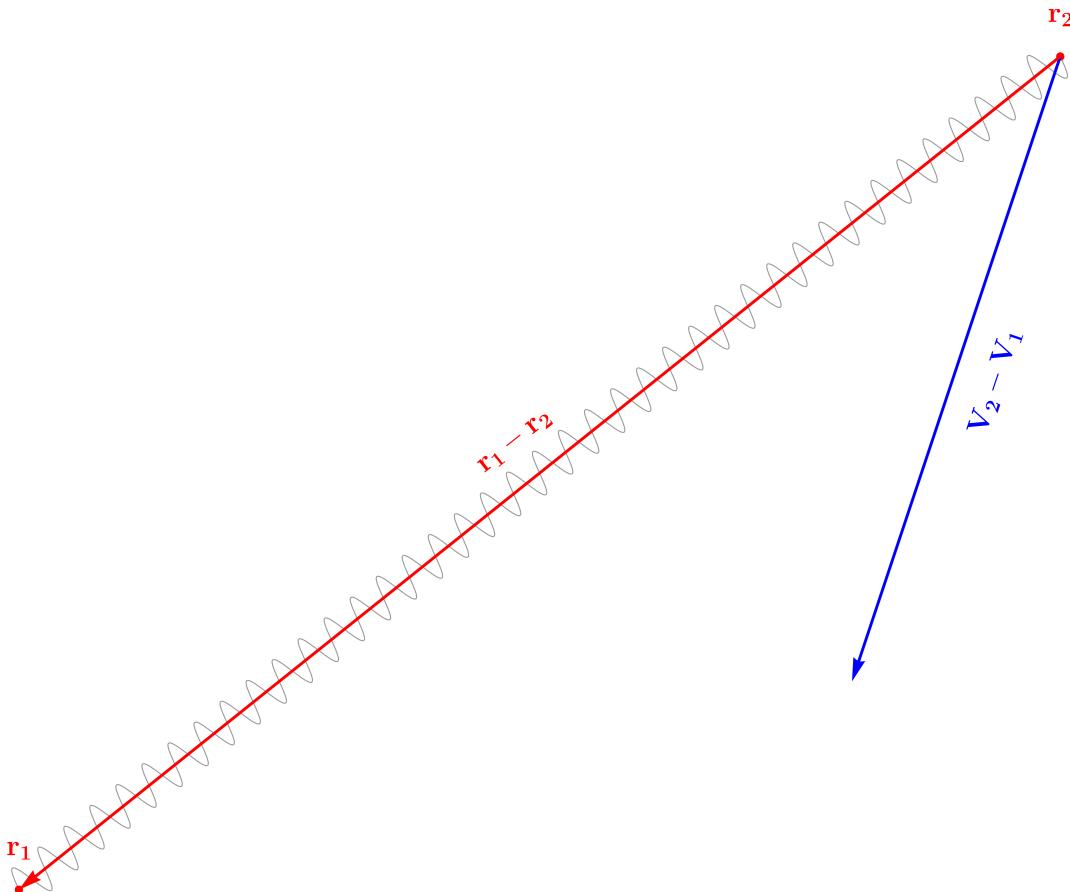
$$\alpha^2 (W_{k1} + W_{kr1}) = W_{p0} + W_{k0} + W_{kr0} - W_{p1} - W_{pr1} - W_{M1} - A \iff \alpha = \sqrt{\frac{W_{p0} + W_{k0} + W_{kr0} - W_{p1} - W_{pr1} - W_{M1} - A}{W_{k1} + W_{kr1}}}$$

S tem smo dobili izraz za faktor α , s katerim pomnožimo hitrosti in kotne hitrosti, da velja energijski zakon. Izrazi za posamezne od teh energij so splošno znani in so našteti spodaj:

- $W_p = \sum mgh = g \sum m\mathbf{r}_* \cdot (0, 0, 1)$
- $W_k = \frac{1}{2} \sum mV_*^2$
- $W_{kr} = \frac{1}{2} \sum \boldsymbol{\omega} \cdot J \cdot \boldsymbol{\omega}$
- $W_{pr} = \frac{k}{2} \sum \delta r^2$
- $W_M = u_p \sum \int_0^{\Delta\varphi_1} f(\Delta\varphi) d\Delta\varphi \implies \frac{D}{2} \sum \Delta\varphi^2$
- $dA = -dt \sum \mathbf{F}_z \cdot \mathbf{V}_* + \mathbf{M}_z \cdot \boldsymbol{\omega}$

3.13 POPRAVEK Z DUŠENJEM VZMETI

Pri temu popravku si zamislimo, da vzmeti ustvarjajo zavorno silo, razmerno hitrosti stiskanja oz. raztezanja vzmeti. Recimo, da je eno krajišče vzmeti \mathbf{r}_1 , drugo \mathbf{r}_2 , hitrost prvega krajišča \mathbf{V}_1 in drugega \mathbf{V}_2 (glej sliko).



Slika 12: Krčenje vzmeti

Če se postavimo v koordinatni sistem prvega krajišča, je hitrost drugega krajišča $\mathbf{V}_2 - \mathbf{V}_1$ (modri vektor na sliki). Hitrost krčenja vzmeti je enaka komponenti $\mathbf{V}_2 - \mathbf{V}_1$, ki kaže v smeri vzmeti:

$$V_{kr} = (\mathbf{V}_2 - \mathbf{V}_1) \cdot \frac{\mathbf{r}_1 - \mathbf{r}_2}{|\mathbf{r}_1 - \mathbf{r}_2|}$$

Zavorna sila dušenja vzmeti je zato enaka

$$\mathbf{F}_D = k_D \left((\mathbf{V}_2 - \mathbf{V}_1) \cdot \frac{\mathbf{r}_1 - \mathbf{r}_2}{|\mathbf{r}_1 - \mathbf{r}_2|} \right) \frac{\mathbf{r}_1 - \mathbf{r}_2}{|\mathbf{r}_1 - \mathbf{r}_2|} = \frac{k_D (\mathbf{V}_2 - \mathbf{V}_1) \cdot (\mathbf{r}_2 - \mathbf{r}_1)}{|\mathbf{r}_2 - \mathbf{r}_1|^2} (\mathbf{r}_2 - \mathbf{r}_1)$$

Celotna sila v zmeti \mathbf{F}_V , če vključimo dušenje je potem enaka

$$\mathbf{F}_V = \sum_i \left(k + \frac{k_D (\mathbf{V}_{i2} - \mathbf{V}_{i1}) \cdot (\mathbf{r}_{i2} - \mathbf{r}_{i1})}{|\mathbf{r}_{i2} - \mathbf{r}_{i1}|^2} \right) (\mathbf{r}_{i2} - \mathbf{r}_{i1})$$

To isto naredimo še za navor v zmeti \mathbf{M}_V :

$$\mathbf{M}_V = \sum_i \left(k + \frac{k_D (\mathbf{V}_{i2} - \mathbf{V}_{i1}) \cdot (\mathbf{r}_{i2} - \mathbf{r}_{i1})}{|\mathbf{r}_{i2} - \mathbf{r}_{i1}|^2} \right) (\mathbf{r}_{i1} - \mathbf{r}_*) \times (\mathbf{r}_{i2} - \mathbf{r}_{i1})$$

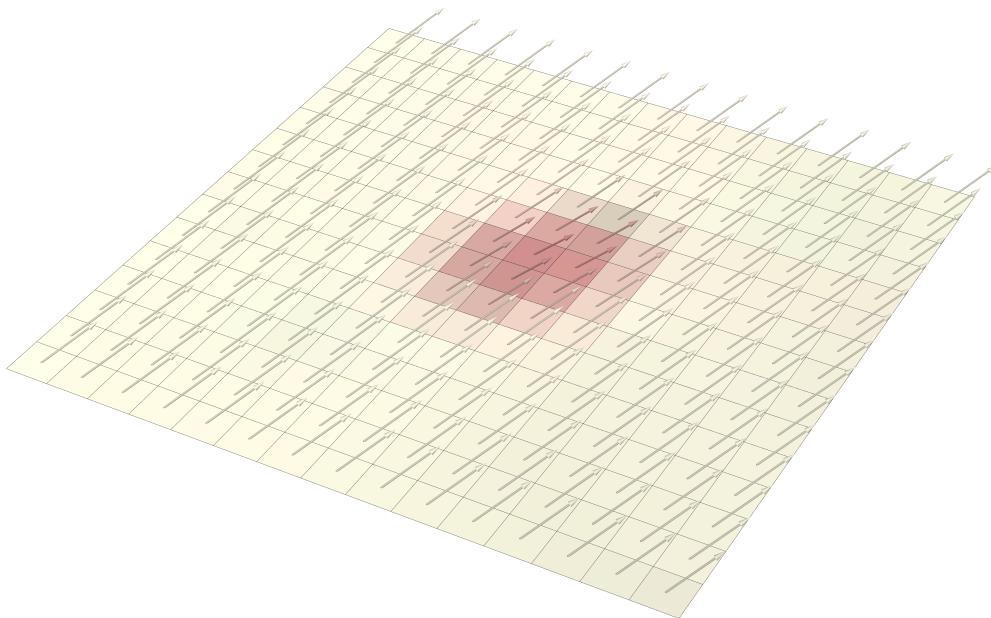
\mathbf{V}_{i1} in \mathbf{V}_{i2} pa poznamo že od prej:

$$\mathbf{V}_{i1} = \mathbf{V}_{*i1} + \boldsymbol{\omega}_1 \times (\mathbf{r}_{i1} - \mathbf{r}_{*i1})$$

$$\mathbf{V}_{i2} = \mathbf{V}_{*i2} + \boldsymbol{\omega}_2 \times (\mathbf{r}_{i2} - \mathbf{r}_{*i2})$$

4 SLEDENJE LETALOM IZ POSNETKA

Posnetek letala je sestavljen iz slik, slike pa iz mreže pikslov. Barva vsakega piksla je podana z vektorjem $\text{RGB} = (R, G, B)$, kjer so komponente deleži moči P svetlenja rdeče, modre in zelene luči v pikslu (glej sliko).



Slika 13: Vektorji RGB

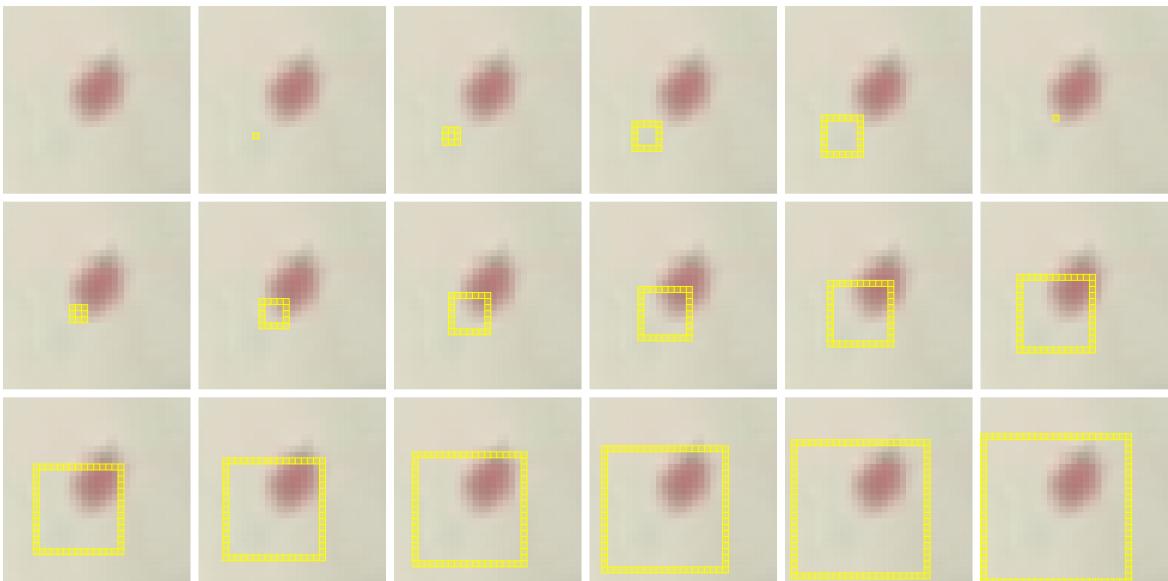
Na vsako ploskev papirnega letala narišemo tri barvne pike. Program je napisan tako, da najprej na prvi sliki pritisnemo barvno piko. Od tod program dobi barvo pike RGB_0 . Za sredino pike na sliki vzamemo kar povprečje koordinat pikslov te pike. Program mora najprej ugotoviti kateri piksli so del te pike. To naredimo tako, da za vsak piksel pregledamo, ali je njegova barva dovolj podobna barvi, izbrani na začetku. Zato si je treba izbrati smiseln kriterij za določanje, kdaj sta si barvi dovolj podobni. Lahko bi vzeli velikost razlike barvnih vrednosti $|\text{RGB}_1 - \text{RGB}_2|$, ampak to ne bi delovalo dobro. Letalo se giblje, pri čemer se spreminja kot ploskve glede na kamero. Zaradi tega se spreminja tudi gostota svetlobnega toka, ki se od pike odbije proti kamери. To se na RGB kaže kot daljšanje oziroma krašanje vektorja RGB, ki pa še vedno kaže v isto smer. Za določitev če sta si barvi dovolj podobni, bi bil bolj smiseln kriterij: ali imata njuna RGB vektorja dovolj podobno smer. To pa lahko ugotovimo s skalarnim produktom smernih vektorjev. Če sta si barvi dovolj podobni, bo ta dovolj blizu 1:

$$\frac{\text{RGB}_1}{\text{RGB}_1} \cdot \frac{\text{RGB}_2}{\text{RGB}_2} > 1 - \epsilon$$

Če za ϵ vzamemo premajhno vrednost, potem program ne bo našel vseh piksov barvne pike, saj se med seboj za malenkost razlikujejo. Če izberemo preveliko vrednost, pa bo program štel tudi piksele, ki ne spadajo v to piko. Izkaže se, da ponavadi dobro delujejo vrednosti ϵ okoli 0,0005.

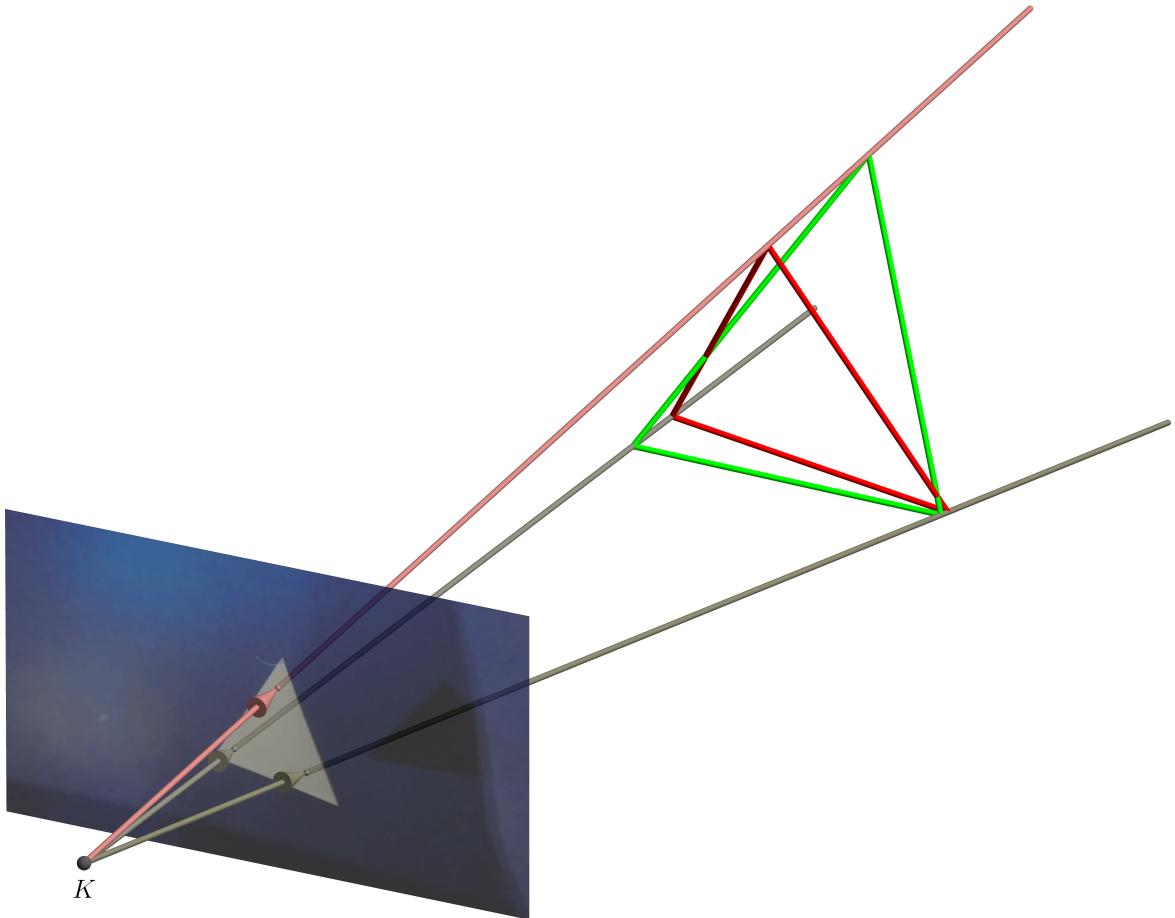
Če opisani algoritem za iskanje sredine pike sprogramiramo, se izkaže, da program deluje prepočasi. Za vsako sliko potrebuje okoli 10s. Program je počasen, saj pregleduje vse piksele na sliki. Lahko ga pospešimo tako, da pregleda samo piksele v okolini pike. Za to uporabimo funkcijo FindFit, vgrajeno v Mathematico. Ta poišče vrednosti parametrov v podani funkciji, pri katerih je vsota odstopanj točk najmanjša, oziroma, se graf funkcije najbolje prilega točkam. To funkcijo uporabimo tako, da gibanje sredine pike na sliki razdelimo na navpično in vodoravno komponento. Za vsakega od teh s funkcijo FindFit poiščemo polinom čez zadnjih nekaj točk (t, x) oziroma (t, y). Izkaže se, da dobro deluje polinom 5. stopnje skozi zadnjih 10 točk. S tem lahko ugibamo, kje približno bo sredina barvne pike na naslednji sliki: $(p_1(t + dt), p_2(t + dt))$

Ko imamo začetno točko pregledovanja piksov, jo najprej zaokrožimo, da dobimo začetni piksel. Ta je lahko že v barvni piki, lahko pa ni. Od tega začnemo navzven v koncentričnih kvadratnih zankah pregledovati piksele. Če bi kvadratna zanka slučajno segala čez rob slike, vzamemo pravokotno zanko, s problematičnim robom na robu slike. Tako iz začetnega piksla navzven skeniramo piksele, dokler ne trčimo v piksele prave barve. Od teh enega izberemo. Potem od izbranega spet navzven pregledujemo piksele. To delamo, dokler je na trenutni zanki piksov vsaj 1 piksel prave barve. Ko na zanki ni več piksov prave barve, je celotna pika znotraj zanke. Zato vemo, da smo našli vse piksele pike. Algoritem je prikazan na naslednji sliki.



Slika 14: Algoritem za pregledovanje piksov

Če na tak način na vsaki ploskvi sledimo trem barvnim pikam, lahko dobimo gibanje letala v prostoru. Poznati moramo začetne koordinate pik v simulaciji $\mathbf{R}_{0,1}$, $\mathbf{R}_{0,2}$, $\mathbf{R}_{0,3}$. Te Točke tvorijo trikotnik. Izračunamo lahko dolžine stranic trikotnika $|\mathbf{R}_{0,3} - \mathbf{R}_{0,2}|$, $|\mathbf{R}_{0,1} - \mathbf{R}_{0,3}|$, $|\mathbf{R}_{0,2} - \mathbf{R}_{0,1}|$. Predstavljamo si lahko, da se slika kamere nahaja ΔY_0 pred kamero in je široka ΔX_0 (glej naslednjo sliko). Podatka ΔX_0 in ΔY_0 bomo izmerili v poglavju snemalni kot kamere.



Slika 15: Sledenje gibanju letala v prostoru

Vektor do levega spodnjega vogala slike je enak

$$\left(-\frac{\Delta X_0}{2}, \Delta Y_0, -\frac{\Delta X_0 L_y}{2L_x} \right)$$

Kjer je $L_x \times L_y$ ločljivost slike.

Ker je slika široka ΔX_0 , je širina piksla enaka $\Delta X_0 / L_x$. S tem lahko izrazimo lego sledenih barvnih pik na sliki v 3D. Recimo, da je sredinski piksel sledene pike na sliki p_x -ti piksel od leve proti desni in p_y -ti piksel od dna proti vrhu. Potem bo lega tega piksla na sliki v prostoru enaka

$$\left(-\frac{\Delta X_0}{2} + p_x \frac{\Delta X_0}{L_x}, \Delta Y_0, -\frac{\Delta X_0 L_y}{2L_x} + p_y \frac{\Delta X_0}{L_x} \right)$$

Na prejšnji sliki je so lege sredinskih pikov sledenih pik prikazane z vektorji. Recimo, da smo po prejšnji enačbi dobili, da so ti vektorji \mathbf{r}_1 , \mathbf{r}_2 in \mathbf{r}_3 .

Zdaj pa si zamislimo, kje v prostoru so ploskve letala in pike na njih. Pike gotovo ležijo nekje na premicah, narisanih na prejšnji sliki. Vemo pa tudi, da tvorijo trikotnik z znanimi dolžinami stranic. To lahko zapišemo s sistemom enačb:

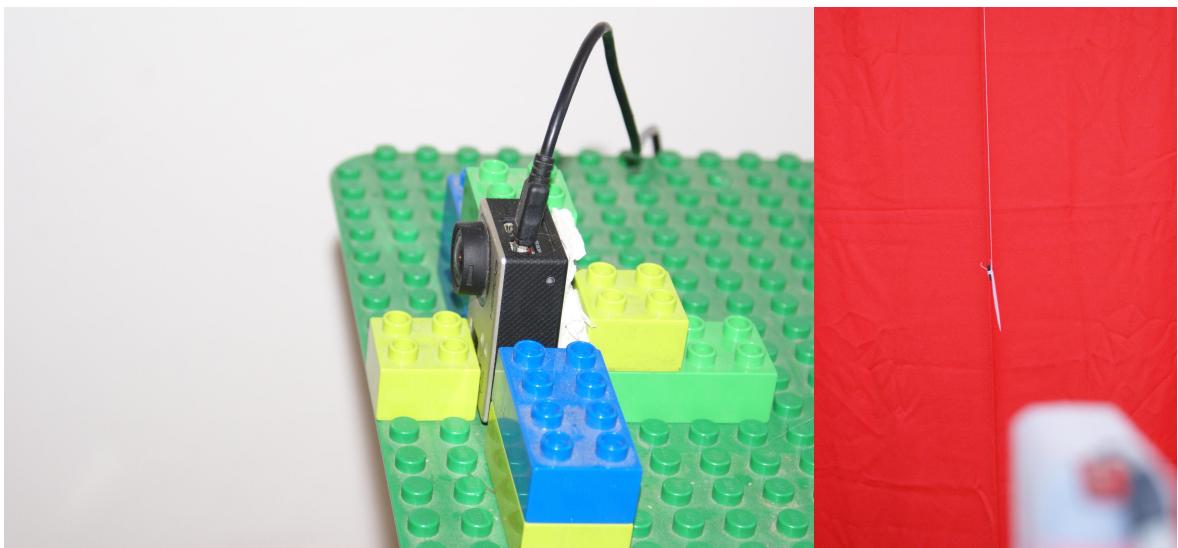
$$|\mathbf{R}_{0,3} - \mathbf{R}_{0,2}| = |c_3\mathbf{r}_3 - c_2\mathbf{r}_2|$$

$$|\mathbf{R}_{0,1} - \mathbf{R}_{0,3}| = |c_1\mathbf{r}_1 - c_3\mathbf{r}_3|$$

$$|\mathbf{R}_{0,2} - \mathbf{R}_{0,1}| = |c_2\mathbf{r}_2 - c_1\mathbf{r}_1|$$

Iz zgornjega sistema dobimo parametre c_1 , c_2 in c_3 in s tem lege sledenih pik v prostoru $c_1\mathbf{r}_1$, $c_2\mathbf{r}_2$ in $c_3\mathbf{r}_3$. Dobimo 4 komplete rešitev. 2 od teh sta za kamero in nas zato ne zanimata, 2 pa sta pred njo. Na sliki sta označena z dvema trikotnikoma. To je tako kot znana optična iluzija sence lutke, za katero se ne da določiti v katero smer se vrta. Zato moramo v program ročno vnesti še katera rešitev je prava.

Na ta način lahko sledimo gibanju vsake ploskve papirnega letala. Metoda dobro deluje na preprostem primeru, ko je letalo snemano zelo od blizu in dobro osvetljeno. Sicer se izkaže, da je ločljivost kamere premajhna in stvar ne deluje. Pri nižjem številu slik na sekundo in višji ločljivosti tudi ne deluje, saj kamera zajema svetlobo dlje časa in so pike zamazane. Vseeno pa lahko sledimo celotnim letalom, kot točkastim telesom, če se ti ne gibajo v y -smeri. Za barvno piko vzamemo kar celotno letalo. Potem moramo poznati še ΔY razdaljo letala od kamere. To naredimo tako, da iz mesta spuščanja papirnega letala obesimo kocko. Nato laserski merilnik postavimo na mesto kamere in izmerimo razdaljo do kocke. Dobimo $\Delta Y = 1,25 \text{ m}$ (glej sliko).



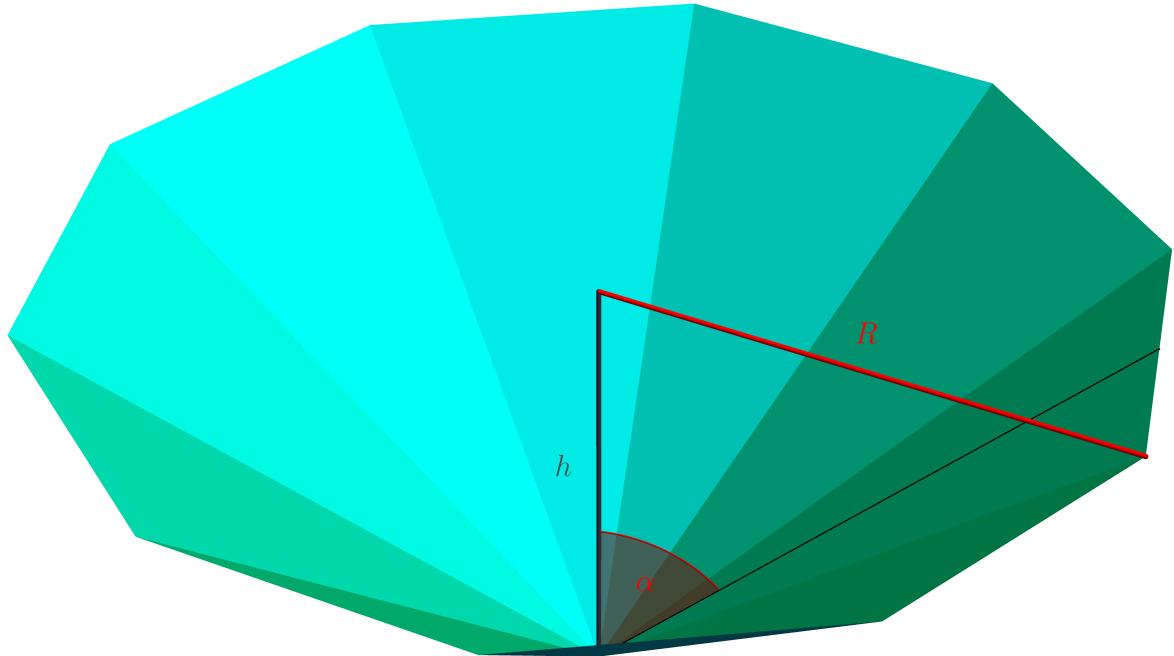
Slika 16: Kamera in merjenje razdalje ΔY

Potem po Talesovem izreku dobimo $(\Delta X, \Delta Z)$ koordinate papirnega letala:

$$(\Delta X, \Delta Z) = \frac{\Delta Y}{\Delta Y_0} \left(-\frac{\Delta X_0}{2} + p_x \frac{\Delta X_0}{L_x}, -\frac{\Delta X_0 L_y}{2 L_x} + p_y \frac{\Delta X_0}{L_x} \right)$$

5 PIRAMIDNO LETALO

V tem poglavju bomo obravnavali primer papirnega letala, ki je dovolj enostavno, da lahko pod predpostavkami iz modela dobimo analitične rešitve za njegovo gibanje. Potem lahko tudi ugotovimo približni red natančnosti modela. Piramidno letalo definirajmo, kot stranske ploskve pravilne pokončne s -strane piramide (glej sliko).



Slika 17: Piramidno letalo

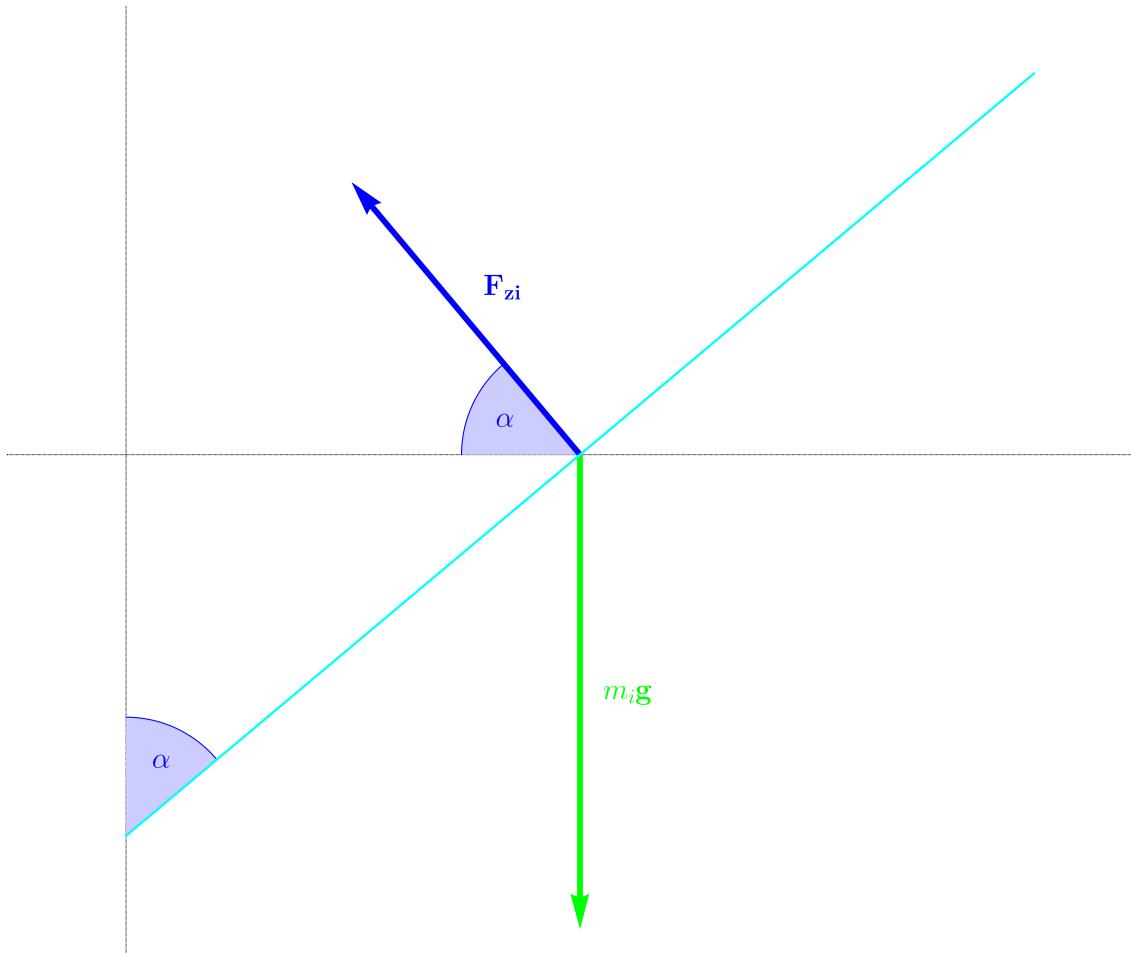
Najprej z R in h lahko izrazimo kot α . Zgornje, desno krajišče poševne črne črte na zgornji sliki je na sredini ene izmed stranic s -kotnika. Recimo, da je koordinatni sistem tak, da je s -kotnik na ravnini osi x in y in je eno oglišče $R(1, 0, 0)$. Naslednje oglišče je potem $R(\cos(\frac{2\pi}{s}), \sin(\frac{2\pi}{s}), 0)$. Povprečje teh dveh je na $\frac{R}{2}(1 + \cos(\frac{2\pi}{s}), \sin(\frac{2\pi}{s}), 0)$. S tem lahko izrazimo kot α :

$$\alpha = \arctan \left(\frac{\frac{R}{2} \sqrt{(\cos(\frac{2\pi}{s}))^2 + (\sin(\frac{2\pi}{s}))^2}}{h} \right) = \arctan \left(\frac{R \cos(\frac{\pi}{s})}{h} \right)$$

Ker bomo piramidna letala izrezovali iz papirja, moramo za izris mrež piramidnih letal poznati še polmer očrtanega kroga mreži R_m in kote ob vrhu trikotnikov φ_m . R_m je po Pitagorovem izreku $\sqrt{R^2 + h^2}$. φ_p pa lahko izračunamo s skalarnim produktom:

$$\varphi_p = \arccos \left(\frac{(R, 0, h) \cdot (R \cos(\frac{2\pi}{n}), R \sin(\frac{2\pi}{n}), h)}{R_m^2} \right) = \arccos \left(\frac{R^2 \cos(\frac{2\pi}{n}) + h^2}{R^2 + h^2} \right)$$

Izpeljimo izraz za gibanje piramidnega papirnega letala z danim kotom α . Slika prikazuje silo zraka \mathbf{F}_{zi} in težo $m_i \mathbf{g}$ na i -to ploskev papirnega letala.



Slika 18: Sile na piramidno letalo

Recimo, da bomo letalo spuščali samo obrnjeno z vrhom piramide navzdol. Potem je letalo simetrično in lahko vodoravne komponente \mathbf{F}_{zi} zložimo v s -kotnik in je njihova vsota $(0, 0, 0)$ oziroma se izničijo. Zato lahko za gibanje celotnega letala upoštevamo samo navpične komponente \mathbf{F}_{zi} . Po 2. Newtonovem zakonu zato velja:

$$\ddot{z} = g - \frac{F_z \sin(\alpha)}{m} = g - \frac{\rho S (V \sin(\alpha))^2 \sin(\alpha)}{\rho_p S} = g - \frac{\rho (\sin(\alpha))^3}{\rho_p} \dot{z}^2 = g - \beta \dot{z}^2$$

Če to enačbo vnesemo v Wolfram Mathematico, dobimo rešitev:

$$z = \frac{c_2 + \ln(\cosh(\sqrt{g\beta}(t + c_1)))}{\beta};$$

$$c_1 = \beta x_0 - \ln\left(\sqrt{\frac{g}{g - \beta V_0^2}}\right); c_2 = \frac{\operatorname{arccosh}\left(\sqrt{\frac{g}{g - \beta V_0^2}}\right)}{\sqrt{g\beta}}$$

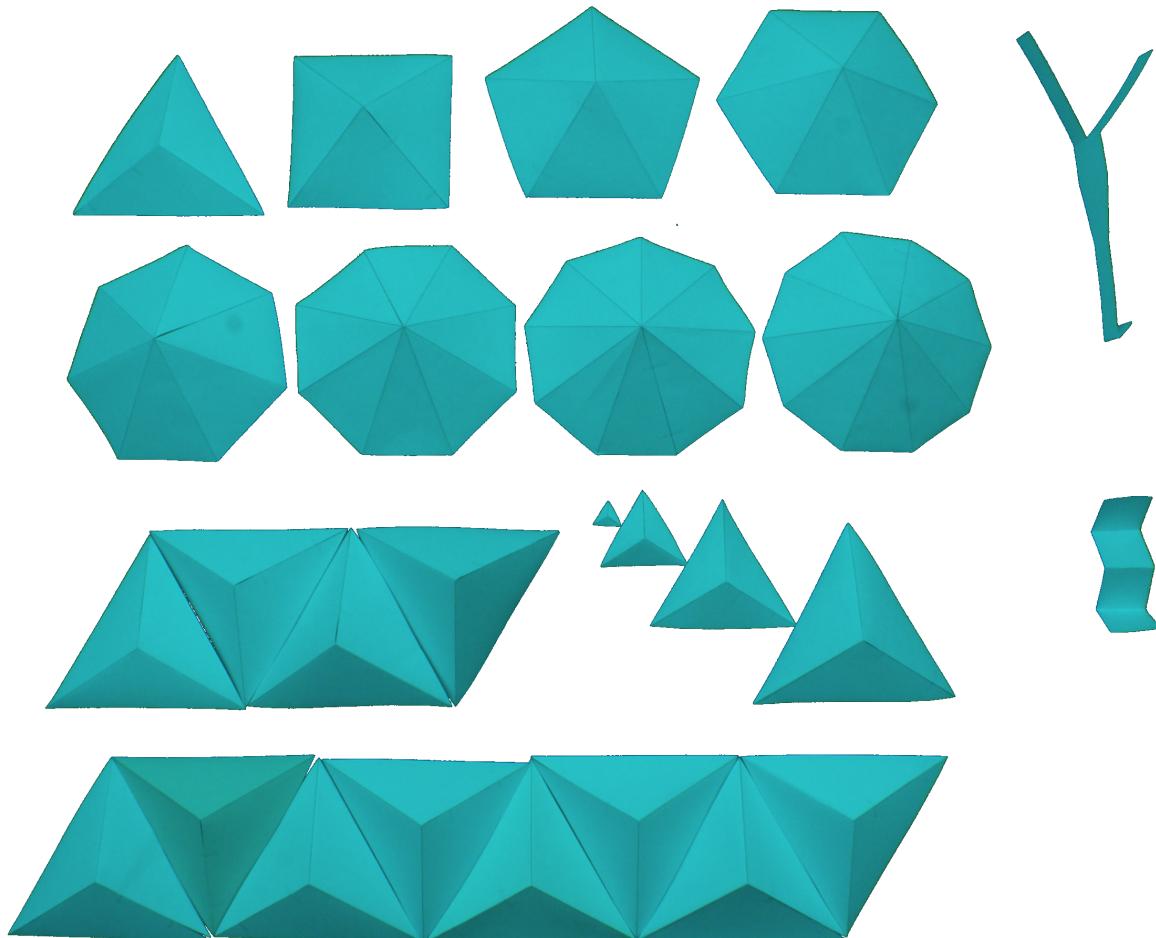
Lahko pa izračunamo tudi terminalno hitrost, če v prejšnjo diferencialno enačbo vstavimo $\ddot{z} = 0$ ali pa izračunamo neskončno limito rešitve. Dobimo:

$$V_{max} = \sqrt{\frac{g}{\beta}}$$

Iz predpostavk modela smo izpeljali enačbe, iz katerih sledi, da je gibanje piramidnega letala pri stalnem zraku, z nespremenljivo gostoto odvisno samo od kota α in ploščinske gostote papirja. Kasneje bom z eksperimentom preveril, ali to drži in s tem preveril eno izmed napovedi modela.

6 EKSPERIMENTALNI DEL

V eksperimentalnem delu raziskovalne naloge so analizirane 3 vrste papirnih letal. Na sliki so prikazana ta letala, izdelana iz papirja.



Slika 19: Izrezana letala

Desno zgoraj imamo helikopter letalo, ki je primer letala, katerega let model dobro opisuje. Pod njim je W letalo. To je primer letala, katerega gibanje model ne napove niti približno. Vse ostalo so Piramidna letala, pri katerih je preverjena natančnost modela in njegovih napovedi. V zgornjih dveh vrstah se letala razlikujejo samo po številu stranic s . V naslednji vrstici levo se razlikujejo samo po kotu α . Desno v isti vrstici se razlikujejo samo po polmeru R . V spodnji vrstici se piramidna letala razlikujejo le po ploščinski gostoti papirja ρ_p , iz katerega so izdelana.

Let letal je bil nato posnet s kamero in analiziran. Da so letala spuščena iz iste točke, iz kock sestavimo mehanizem za spuščanje (glej naslednjo sliko).



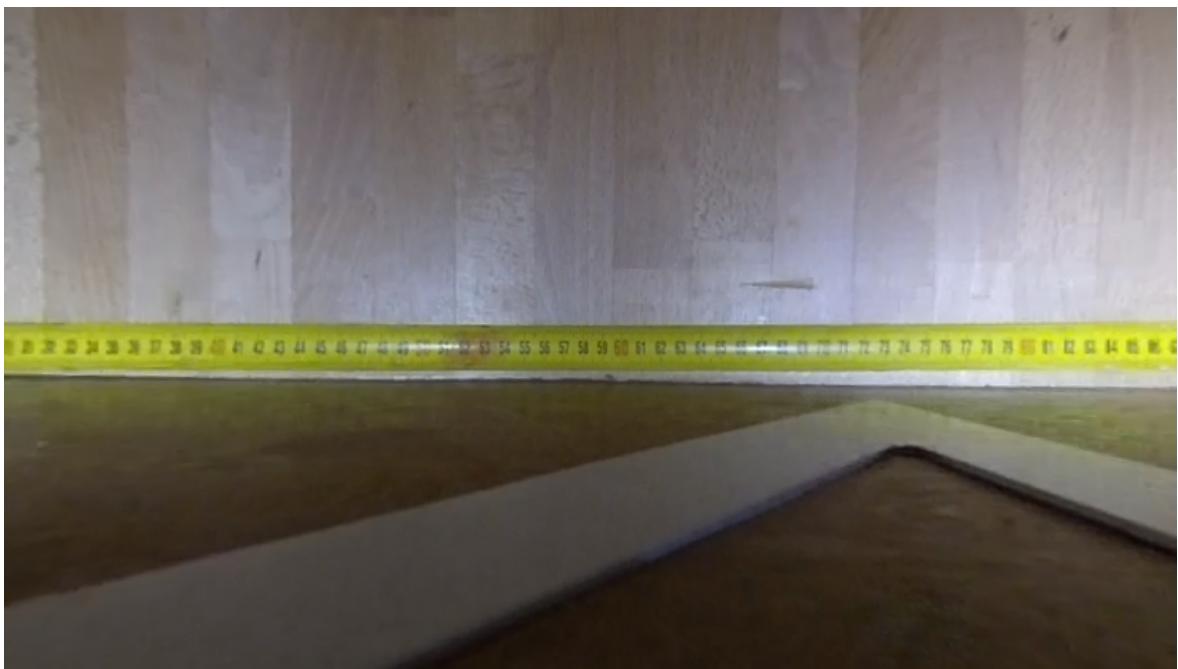
Slika 20: Mehanizem za spuščanje papirnih letal

Mehanizem sprožimo tako, da v levo povlečemo vrvico prožilca. Potem se zaradi navora elastike desni del mehanizma hitro umakne dol in letalo se začne gibati. Pod mesto spuščanja letala obesimo enobarvno ozadje. Poskrbeti moramo tudi za dobro osvetlitev. Osvetlitev z lučmi na izmenični tok za eksperiment ni dobra, saj kamera posname 240 slik v sekundi. Slike nastajajo dovolj malo časa, da je pri nekaterih svetlobnini tok luči ves čas zajemanja majhen. Te slike so zato zrnate in neuporabne za obdelavo. Zato je eksperiment najlažje delati kar pri dnevni svetlobi. Zadnji pomemben korak priprave eksperimenta je poskrbeti, da je kamera usmerjena naravnost proti ravnini spuščanja papirnih letal. To najlažje naredimo z ravnanjem na črte parketa, ki so si vzporedne. Želimo pa tudi, da bo mesto spuščanja letala na sredini slike. To uredimo hkrati z merjenjem razdalje do mesta spuščanja ΔY (opisano na koncu poglavja 4).

6.1 SNEMALNI KOT KAMERE

Za snemanje papirnih letal je bila uporabljena kamera, ki posname 240 slik v sekundi. Posnetek je v osnovi ukrivljen, zato v programu Premiere pro nastavimo odpravljanje ukrivljenosti posnetka in izvozimo slike png. Z odpravljanjem ukrivljenosti izgubimo nekaj robov posnetka. Po odpravljanju ukrivljenosti je ločljivost posnetka 848×480 pikslov.

Naslednji korak je, da določimo zorni kot posnetka. Najprej postavimo kamero vzporedno merilnemu traku. To najlažje naredimo tako, da oboje postavimo na meje med ploščami parketa, ki so si vzporedne. Na isto črto, kjer je kamera postavimo še laserski merilnik razdalje. Da sta na natančno isti črti naredimo tako, da ju postavimo za isto ravnilo (glej sliko).



Slika 21: Merjenje snemalnega kota kamere θ_x

Nato z laserskim merilnikom izmerimo razdaljo do merilnega traku $\Delta Y_0 = 39,1\text{ cm}$. Potem iz merilnega traku odčitamo številke na robovih posnetka in dobimo $\Delta X_0 = 57\text{ cm}$. Zdaj lahko izračunamo snemalni kot kamere:

$$\theta_x = 2 \arctan \left(\frac{\frac{\Delta X_0}{2}}{\Delta Y_0} \right) = 1,26 = 72^\circ$$

6.2 REZULTATI ZA PIRAMIDNA LETALA

Pri analizi posnetkov leta piramidnih letal se zgornjega dela leta ni dalo dobro analizirati zaradi prožilca in bele stene v ozadju. Zato se bomo osredotočili samo na terminalne hitrosti V_{max} teh letal. Za osnovo vzemimo piramidno letalo s podatki $n = 3$, $R = 7,0 \text{ cm}$, $\alpha = 50^\circ$ in $\rho_p = 75 \frac{\text{g}}{\text{m}^2}$. Potem spremenjamo po eno spremenljivko naenkrat (opisano pri sliki 19).

Ko posnamemo papirna letala in dobimo njihove koordinate, s funkcijo FindFit v Mathematici poiščemo strmino premic skozi zadnjih nekaj $z(t)$ meritev. To je izmerjena terminalna hitrost letal. Izmerjene vrednosti terminalnih hitrosti so zbrane v spodnjih tabelah.

| $R [\text{cm}]$ | $V_{max} [\frac{\text{m}}{\text{s}}]$ |
|-----------------|---------------------------------------|
| 1,0 | 0,95 |
| 3,0 | 1,03 |
| 5,0 | 0,89 |
| 7,0 | 0,94 |

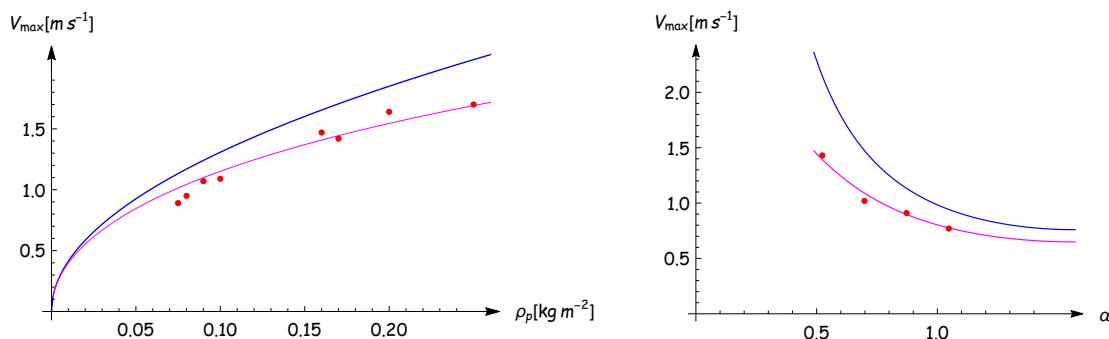
| n | $V_{max} [\frac{\text{m}}{\text{s}}]$ |
|-----|---------------------------------------|
| 3 | 0,90 |
| 4 | 1,02 |
| 5 | 1,03 |
| 6 | 0,97 |
| 7 | 0,96 |
| 8 | 0,95 |
| 9 | 1,03 |
| 10 | 1,03 |

| $\rho_p [\frac{\text{g}}{\text{m}^2}]$ | $V_{max} [\frac{\text{m}}{\text{s}}]$ |
|--|---------------------------------------|
| 75 | 0,89 |
| 80 | 0,95 |
| 90 | 1,07 |
| 100 | 1,09 |
| 160 | 1,47 |
| 170 | 1,42 |
| 200 | 1,64 |
| 250 | 1,70 |

| α | $V_{max} [\frac{\text{m}}{\text{s}}]$ |
|------------|---------------------------------------|
| 30° | 1,43 |
| 40° | 1,02 |
| 50° | 0,91 |
| 60° | 0,77 |

Če primerjamo meritve piramidnega letala s podatki, ki smo jih vzeli za osnovo, lahko ocenimo, da so merske napake velikostnega reda nekaj stotink $\frac{\text{m}}{\text{s}}$. V okviru te natančnosti meritve potrjujejo napoved modela, da V_{max} ni odvisna od R in n .

Zdaj pa meritve primerjajmo še z modelom. Na naslednjih grafih so z rdečimi pikami prikazane meritve iz desnih dveh tabel. Modri krivulji predstavljata napoved modela



Slika 22: V_{max} meritve piramidno letalo

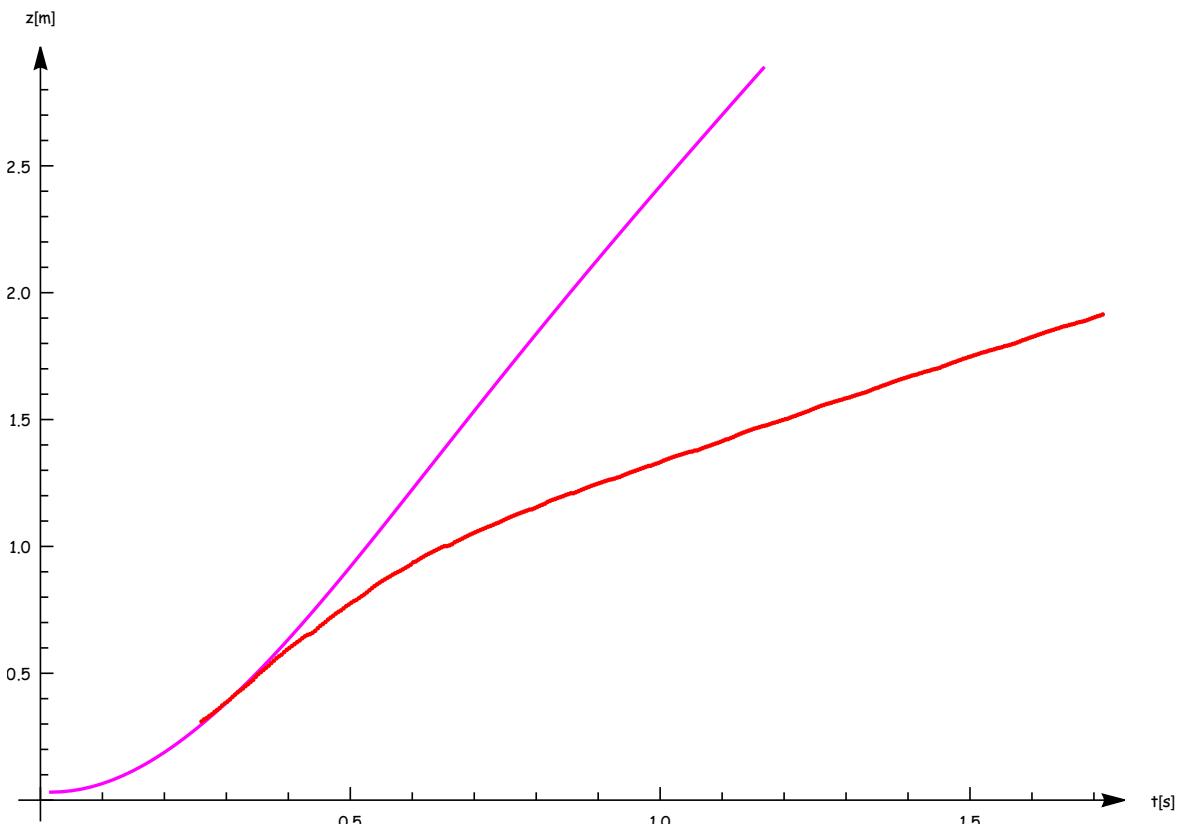
Na levem grafu lahko vidimo, da model za vse manjše kote α napove vse bolj prevelike vrednosti. Odstopanje modela je reda $0,2 \frac{m}{s}$. Model temelji na približku stoječega zraka okoli letala in sile curka. V enačbo za terminalno hitrost dodamo še zavorni člen, linearen s hitrostjo. Ta predstavlja približek za viskozne sile. S FindFit poiščemo vrednost linearnega koeficiente in dobimo boljše prileganje, ki ga prikazujeta magenta črti. Opazimo lahko tudi, da je pri $160 \frac{g}{m^2}$ papirju terminalna hitrost višja, kot pri naslednjem težjem papirju. Ker že vemo, da viskozne sile zraka niso zanemarljive, lahko to pojasnimo z dejstvom, da je bil $160 \frac{g}{m^2}$ papir fotografski in manj hrapav od ostalih.

Preverimo lahko še lastnost, piramidnih letal, da se vedno obrnejo z vrhom navzdol. To dobimo tudi v simulaciji.

Simuliramo lahko tudi preprost primer ukrivljene ploskve, stožec. Tega dobimo v limiti $\lim_{n \rightarrow \infty}$. Ko vzamemo za n veliko število ploskev pa nastane težava. Ker je v spodnjem oglišču povezana vsaka ploskev z vsako drugo, računska kompleksnost narašča kot $\mathcal{O}(n^2)$. Problem lahko rešimo tako, da namesto trikotnikov vzamemo enakokrake trapeze, ki segajo skoraj do vrha piramide.

6.3 REZULTATI ZA HELIKOPTER LETALO

Za merjene helikopterje je bil uporabljen $160 \frac{g}{m^2}$ papir, ki je dovolj čvrst, da ploskve med letom ostanejo ravne. Gibanje helikopter letala lahko opišemo tako: najprej začne padati s pospeškom g . Potem, ko dobi nekaj hitrosti, grejo krila zaradi sile zraka bolj skupaj. Hkrati se veča tudi kotna hitrost vrtenja tega letala okoli navpične osi. Krili gresta nato bolj narazen, zaradi povečane centripetalne sile nanju. Ker gresta bolj narazen, se padanje helikopterja upočasni. V limiti se hitrost in kotna hitrost letala ne spremunjata več. Enak potek faz leta dobimo tudi v simulaciji. Na spodnjem grafu je prikazana višina helikopterja v odvisnosti od časa $z(t)$. Magenta črta je iz simulacije, rdeča pa predstavlja meritve.

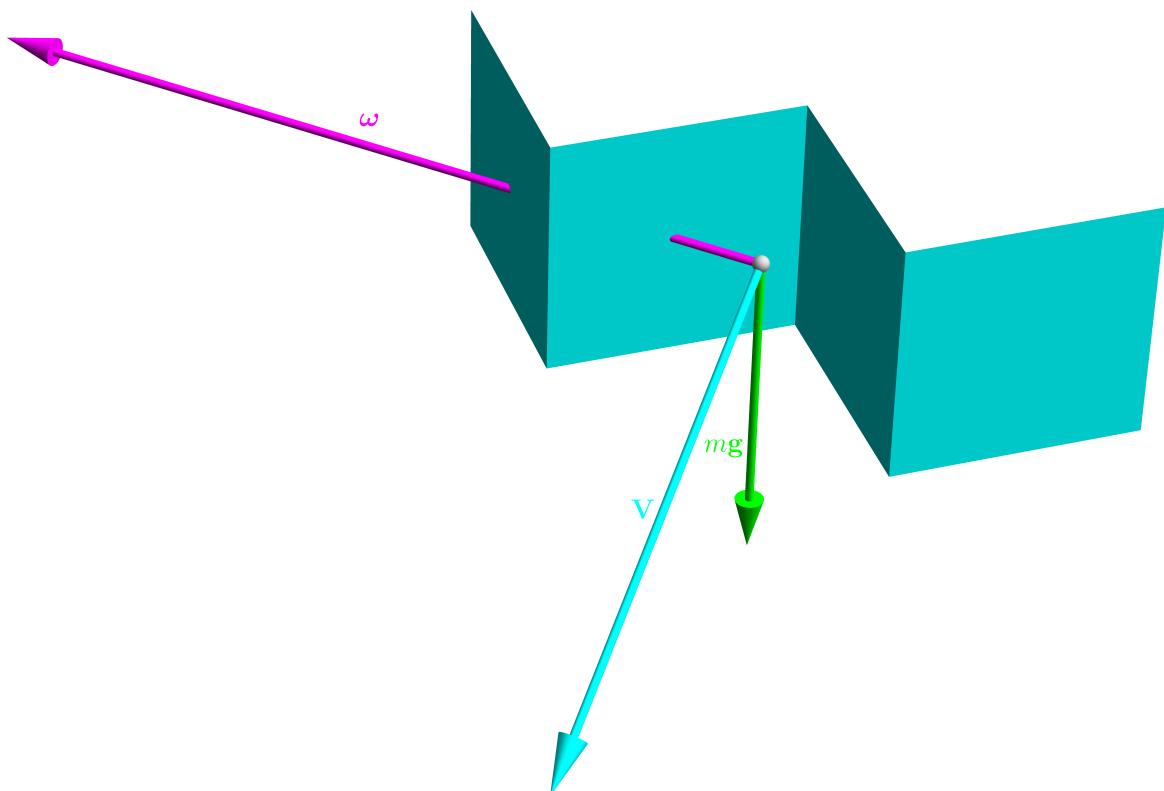


Slika 23: $z(t)$ meritve helikopter letalo

Opazimo lahko, da je simulacija na začetku precej natančna, saj so sile zraka še majhne in je gibanje precej blizu prostemu padu. V nadaljevanju se v simulaciji padanje upočasni, ampak veliko manj, kot v resnici. Še ena lastnost helikopterja, ki jo dobimo tudi v simulaciji, je stabilnejši let, če je letalo spodaj obteženo. To v resnici naredimo tako, da spodno majhno ploskev večkrat zapognemo navzgor (5-krat konkretno na primeru testiranega helikopterja). Kot omenjeno, v simulaciji več plasti papirja ne deluje pravilno. Zato namesto tega samo pomnožimo maso spodnje ploskve s 5.

6.4 REZULTATI ZA W LETALO

W letalo se giblje na zelo zanimiv način. Neglede na to kako izpustimo to letalo, se po nekaj časa gibanje ustali tako, da se težišče letala giblje po premici, letalo pa se obrača s kotno hitrostjo v vodoravni ravnnini in pravokotno na premico gibanja težišča (glej sliko).



Slika 24: W letalo

Letalo se v simulaciji ne začne obračati, temveč samo naravnost pada. Ker model ne deluje, lahko sklepamo, da je za opis gibanja tega letala ključno gibanje zraka okoli letala. V praksi se izkaže, da enako gibanje dobimo že pri podolgovatem papirju v obliki enojnega V in drugih sorodnih oblikah, a je pri W obliki v primerjav z ostalimi zelo hiter prehod na končno gibanje. W letalo je dober primer enostavnega papirnega letala, ki ga s tem modelom ne moremo opisati.

7 RAZPRAVA IN ZAKLJUČEK

Ko sem se lotil raziskovanja in si ustvaril pregled nad obsežnostjo primerov, ki so me zanimali, sem si moral postaviti izvedljive cilje. To so bili:

- Simulacija papirnih letal z zgornjimi predpostavkami
- Sprogramirano sledenje gibanju papirnih letal iz posnetka
- Eksperimentalno preverjanje modela

Cilje sem izpolnil. Naloga je izvirna in model je preverjen na treh vrstah papirnih letal. Naloga ponuja še mnogo možnosti za nadaljnje raziskovanje:

- Uvedba RK metode v simulaciji
- Uvedba različnih k -jev vzmeti (kjer so prisotne večje sile, večjih). Potreben bi bil algoritmom, ki bi ocenil kolikšen k rabimo med dvema ogliščema, da bo med njima odmik ves čas manjši od vnesene omejitve.
- Ugotoviti kako se navor pregibov spreminja s spremembo kota.
- Uporaba funkcije NDSolve za reševanje Navier-Stoksovih enačb namesto približka s silo curka
- Simulacija za papirna letala iz več plasti papirja
- Simuliranje ukrivljanja papirja
- Izboljšanje obdelave posnetka
- GUI za origami pregibanje namesto ročne podaje koordinat oglišč ploskev letala

8 VIRI IN LITERATURA

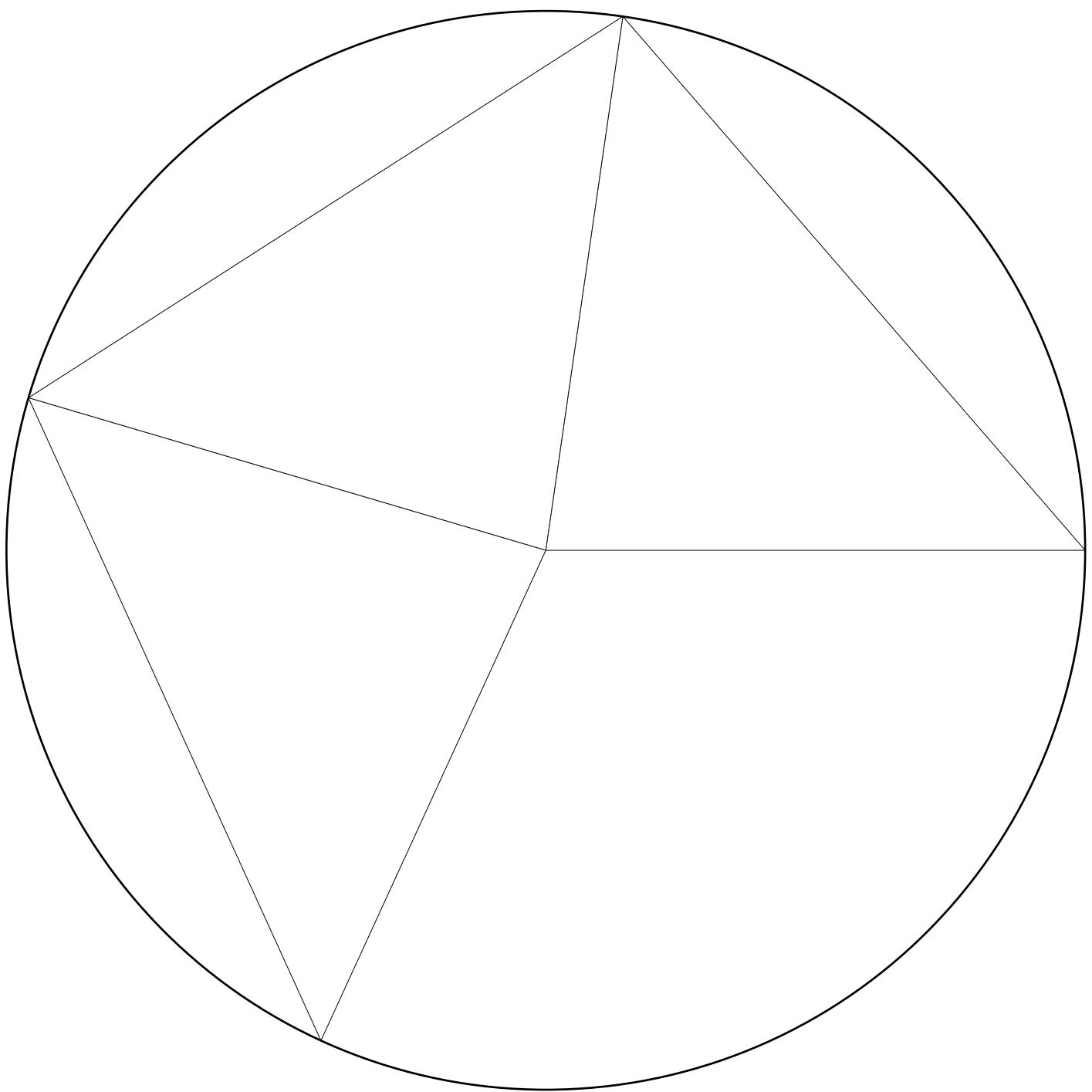
- [1] 3Blue1Brown. *Change of basis - Essence of linear algebra, chapter 13.* Youtube. 2016. URL: <https://www.youtube.com/watch?v=P2LTAU01TdA>.
- [2] Stand-up Maths. *Ellipsoids and The Bizarre Behaviour of Rotating Bodies.* Youtube. 2020. URL: <https://www.youtube.com/watch?v=151LcwHOW7s>.
- [3] rubinhlndau. *8.5 ODE Algorithms.* Youtube. 2020. URL: https://www.youtube.com/watch?v=Yr_ZSnu6XbY.
- [4] The Coding Train. *Coding Challenge nr. 93: Double Pendulum.* Youtube. 2018. URL: https://www.youtube.com/watch?v=uWzPe_S-RVE.
- [5] WIRED. *Aerodynamics Explained by a World Record Paper Airplane Designer - Level Up - WIRED.* Youtube. 2020. URL: https://www.youtube.com/watch?v=3KqjRPV9_PY.

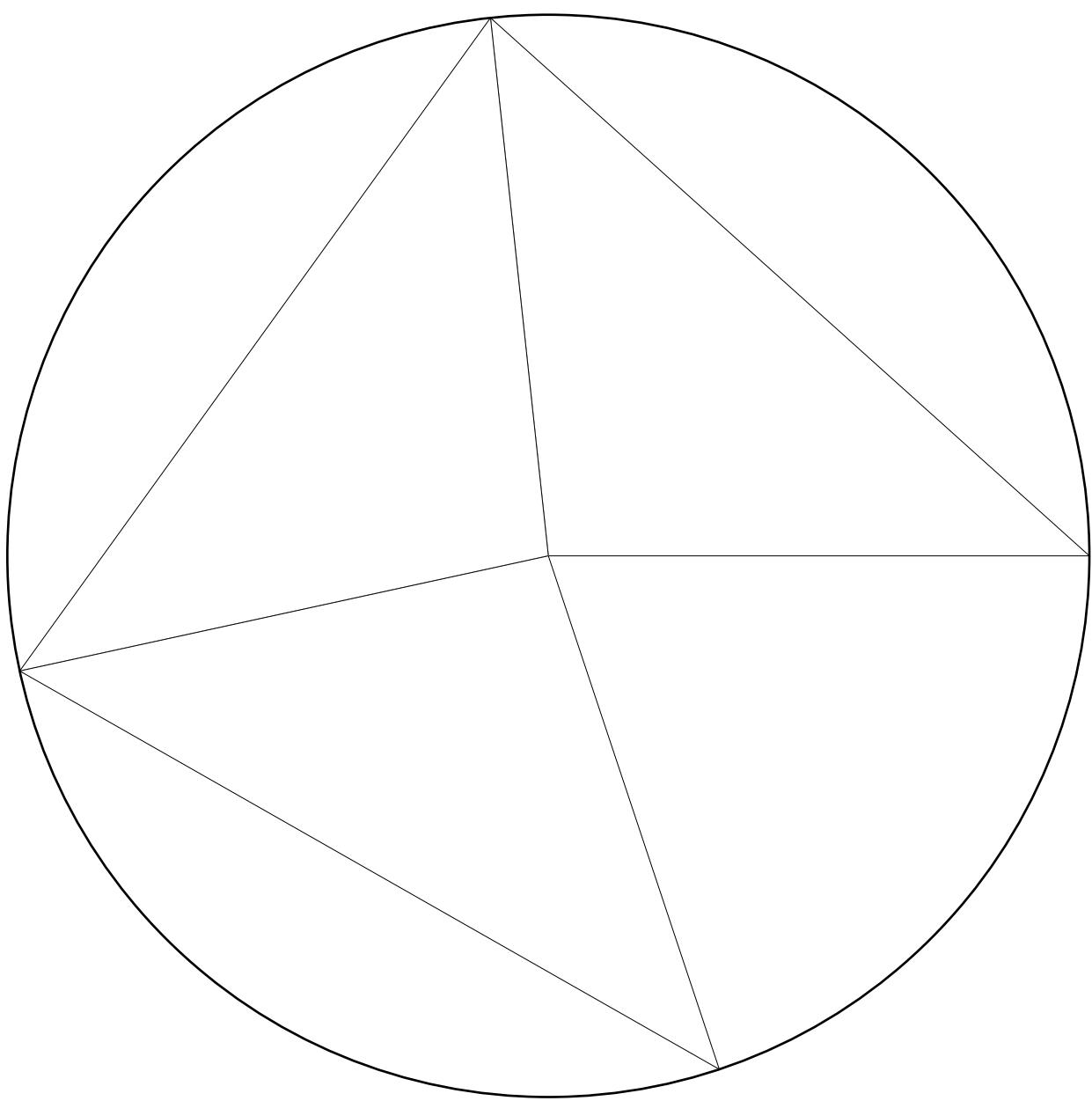
9 PRILOGE

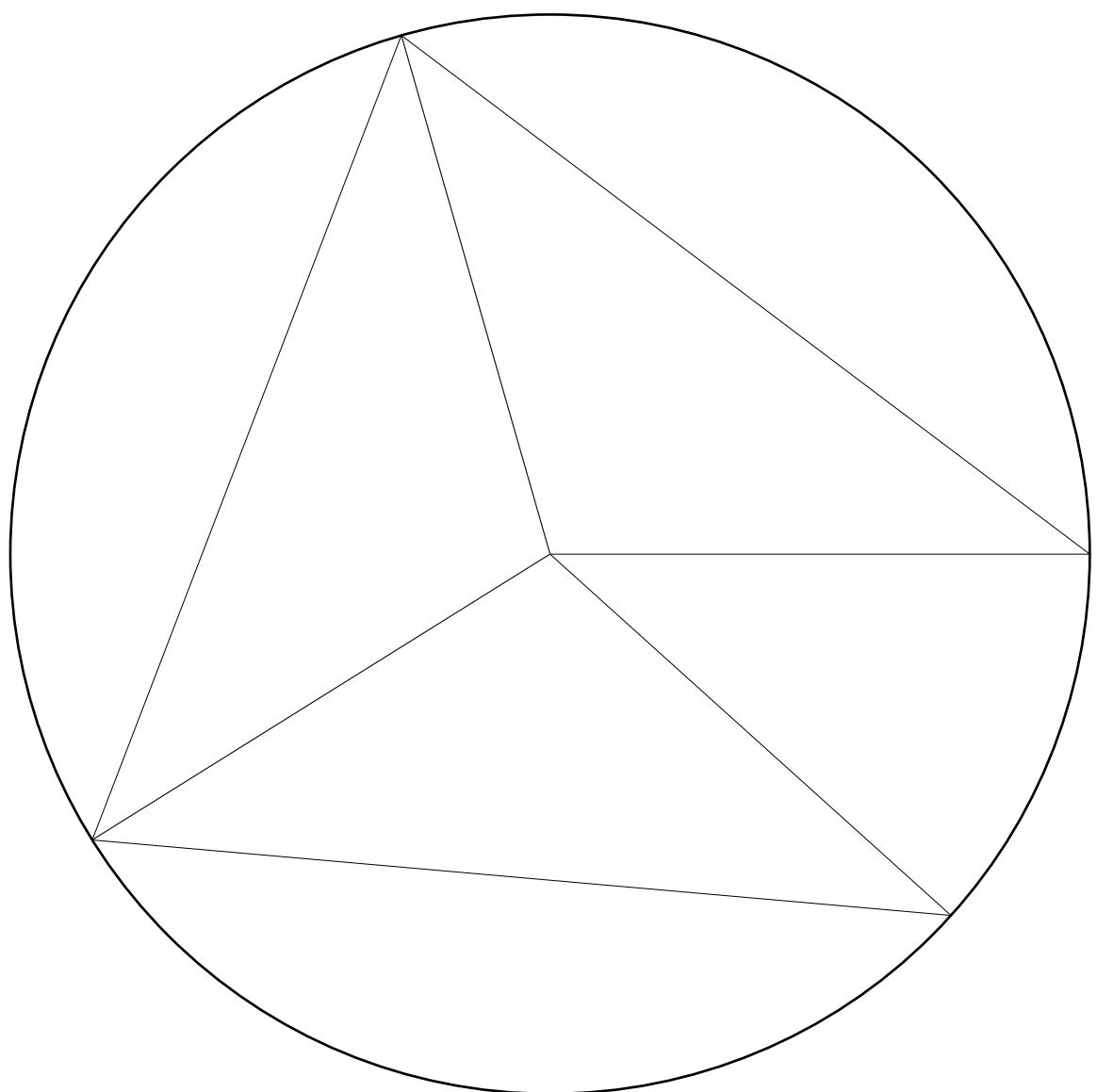
9.1 POENOSTAVLJENA ENAČBA ZA VZTRAJNOSTNI MOMENT TRIKOTNIKA V PROSTORU

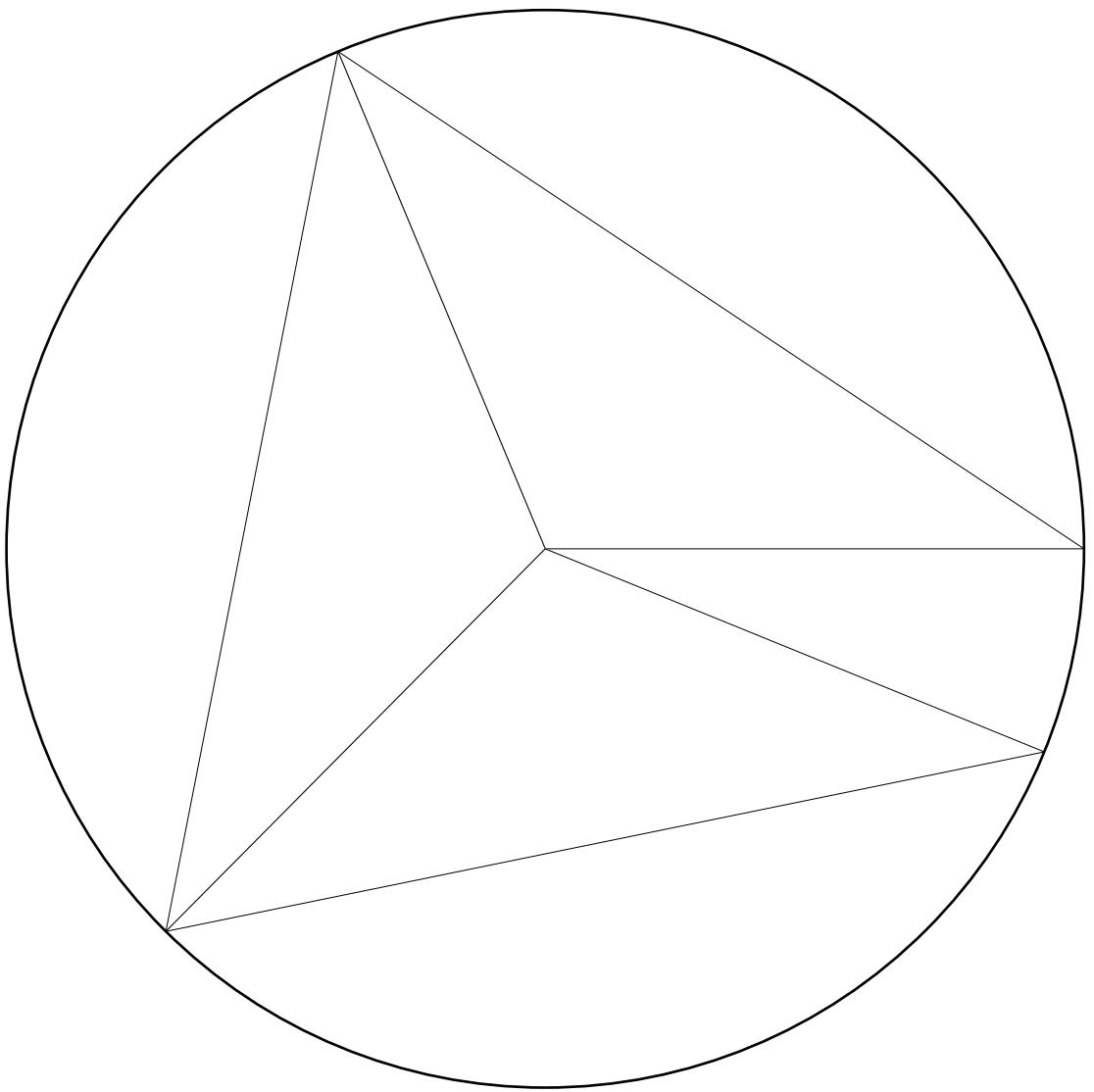
```
J[trikotnik_, ρ2_, tež_] := {
  {{r1x r1y r1z
    r2x r2y r2z
    r3x r3y r3z}} = {r1, r2, r3} = trikotnik;
  B1 = (1/24) ρ2 ((\sqrt{r1yr2x - r1xr2y - r1yr3x + r2yr3x + r1xr3y - r2xr3y})^2 +
    (r1z r2x - r1x r2z - r1z r3x + r2z r3x + r1x r3z - r2x r3z)^2 +
    (r1z r2y - r1y r2z - r1z r3y + r2z r3y + r1y r3z - r2y r3z)^2);
  {B2x, B2y, B2z} = r1 -
  tež; B1
  {{2(6B2y^2 + 6B2z^2 - 8B2yr1y + 3r1y^2 - 8B2zr1z + 3r1z^2 + 4B2yr2y - 3r1yr2y + r2y^2 +
    4B2zr2z - 3r1zr2z + r2z^2 + 4B2yr3y - 3r1yr3y + r2yr3y + r3y^2 + 4B2zr3z - 3r1zr3z +
    r2zr3z + r3z^2), -12B2xB2y + 8B2yr1x + 8B2xr1y - 6r1xr1y -
    4B2yr2x + 3r1yr2x - 4B2xr2y + 3r1xr2y - 2r2xr2y - 4B2yr3x + 3r1yr3x - r2yr3x -
    4B2xr3y + 3r1xr3y - r2xr3y - 2r3xr3y, -12B2xB2z + 8B2zr1x + 8B2xr1z - 6r1xr1z -
    4B2zr2x + 3r1zr2x - 4B2xr2z + 3r1xr2z - 2r2xr2z - 4B2zr3x + 3r1zr3x - r2zr3x -
    4B2xr3z + 3r1xr3z - r2xr3z - 2r3xr3z},
  {-12B2xB2y + 8B2yr1x + 8B2xr1y - 6r1xr1y - 4B2yr2x + 3r1yr2x -
    4B2xr2y + 3r1xr2y - 2r2xr2y - 4B2yr3x + 3r1yr3x - r2yr3x - 4B2xr3y + 3r1xr3y -
    r2xr3y - 2r3xr3y, 2(6B2x^2 + 6B2z^2 - 8B2xr1x + 3r1x^2 -
    8B2zr1z + 3r1z^2 + 4B2xr2x - 3r1xr2x + r2x^2 + 4B2zr2z - 3r1zr2z + r2z^2 + 4B2xr3x -
    3r1xr3x + r2xr3x + r3x^2 + 4B2zr3z - 3r1zr3z + r2zr3z + r3z^2)},
  -12B2yB2z + 8B2zr1y + 8B2yr1z - 6r1yr1z - 4B2zr2y + 3r1zr2y -
  4B2yr2z + 3r1yr2z - 2r2yr2z - 4B2zr3y + 3r1zr3y -
  r2zr3y - 4B2zr3z + 3r1yr3z - r2yr3z - 2r3yr3z},
  {-12B2xB2z + 8B2zr1x + 8B2xr1z - 6r1xr1z - 4B2zr2x + 3r1zr2x -
    4B2xr2z + 3r1xr2z - 2r2xr2z - 4B2zr3x + 3r1zr3x - r2zr3x - 4B2xr3z + 3r1xr3z -
    r2xr3z - 2r3xr3z, -12B2yB2z + 8B2zr1y + 8B2yr1z -
    6r1yr1z - 4B2zr2y + 3r1zr2y - 4B2yr2z + 3r1yr2z - 2r2yr2z -
    4B2zr3y + 3r1zr3y - r2zr3y - 4B2zr3z + 3r1yr3z - r2yr3z - 2r3yr3z,
  2(6B2x^2 + 6B2y^2 - 8B2xr1x + 3r1x^2 - 8B2yr1y + 3r1y^2 + 4B2xr2x -
    3r1xr2x + r2x^2 + 4B2yr2y - 3r1yr2y + r2y^2 + 4B2xr3x -
    3r1xr3x + r2xr3x + r3x^2 + 4B2zr3y - 3r1zr3y + r2zr3y + r3z^2)}}
}; [[1]];
```

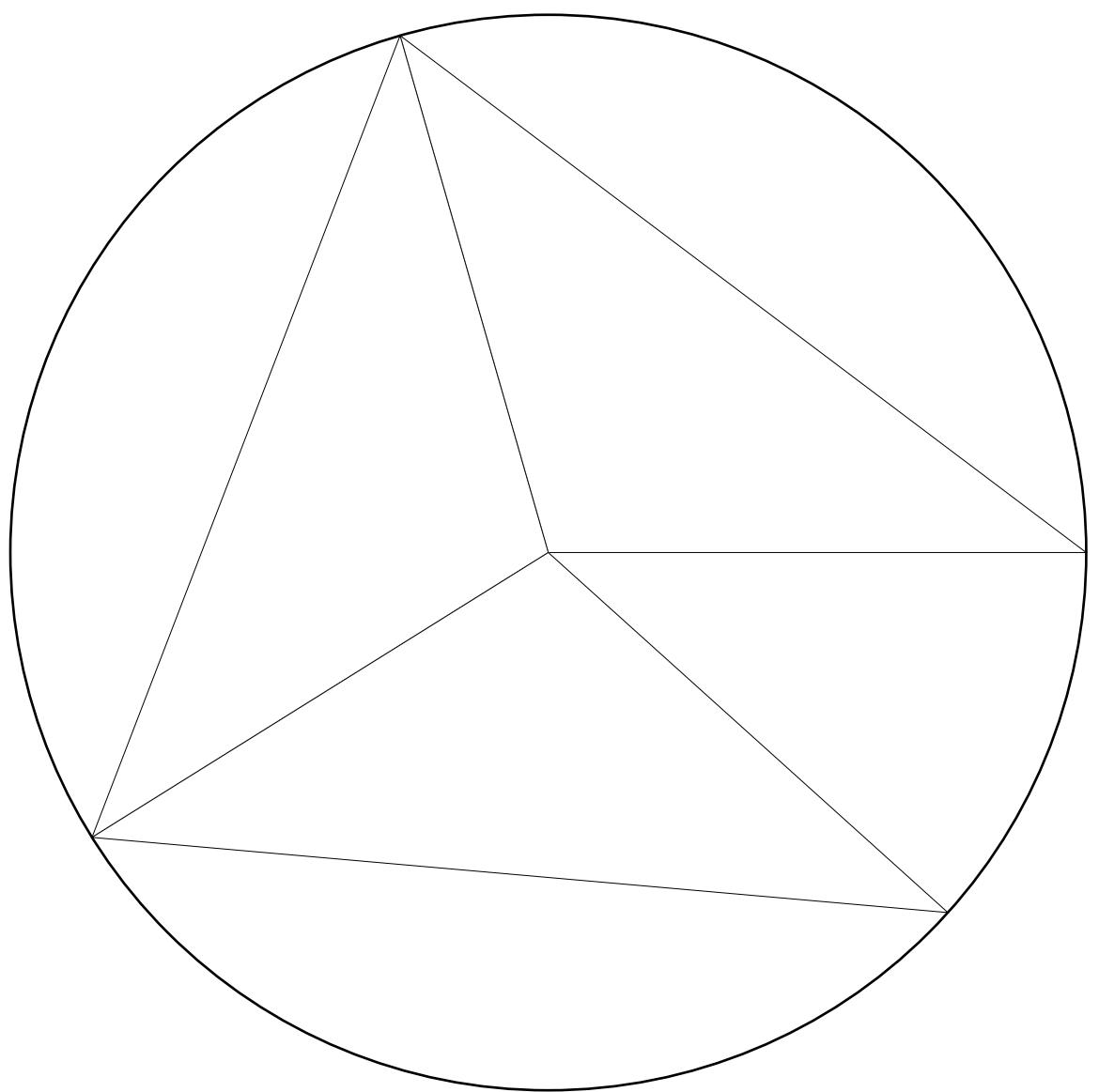
9.2 MREŽE LETAL

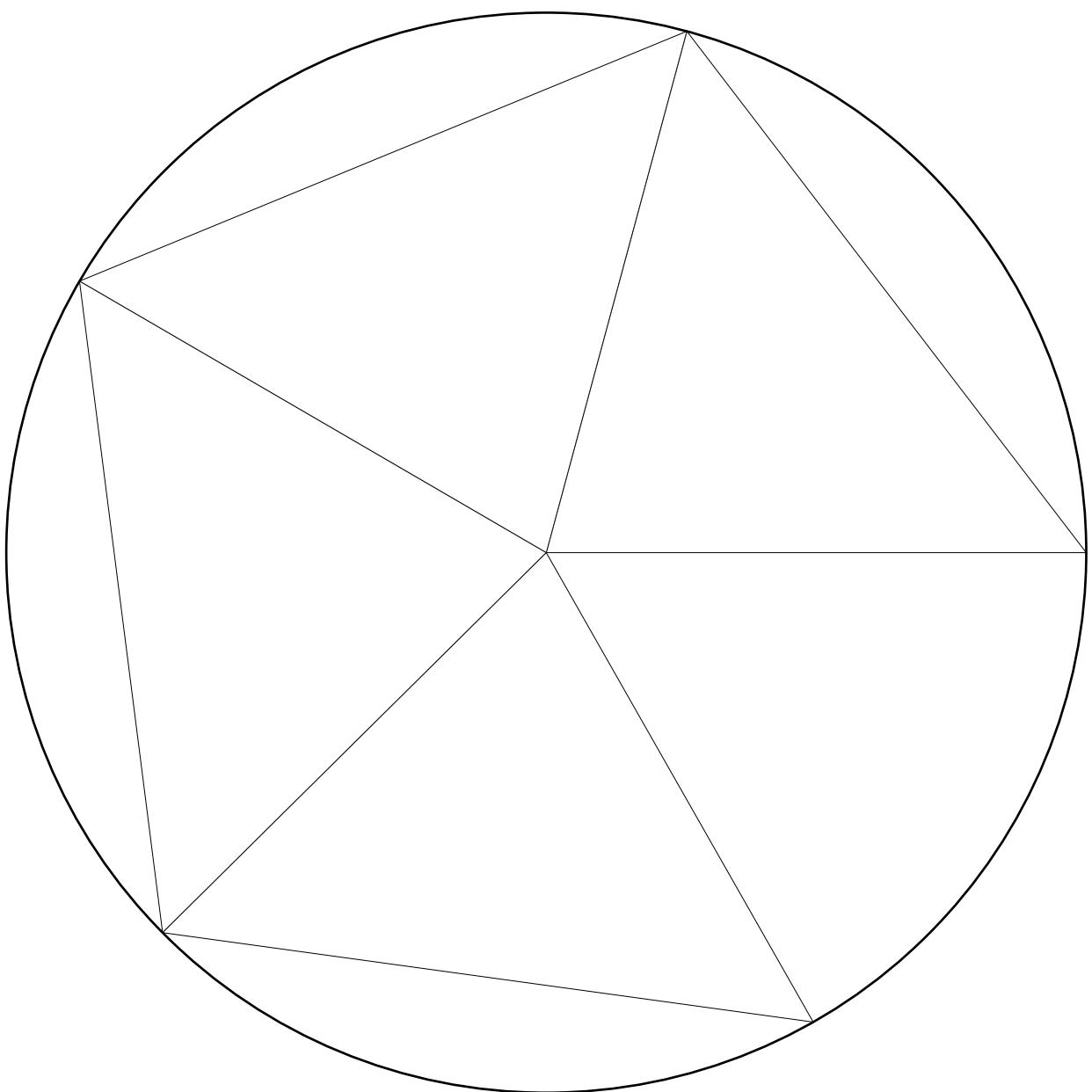


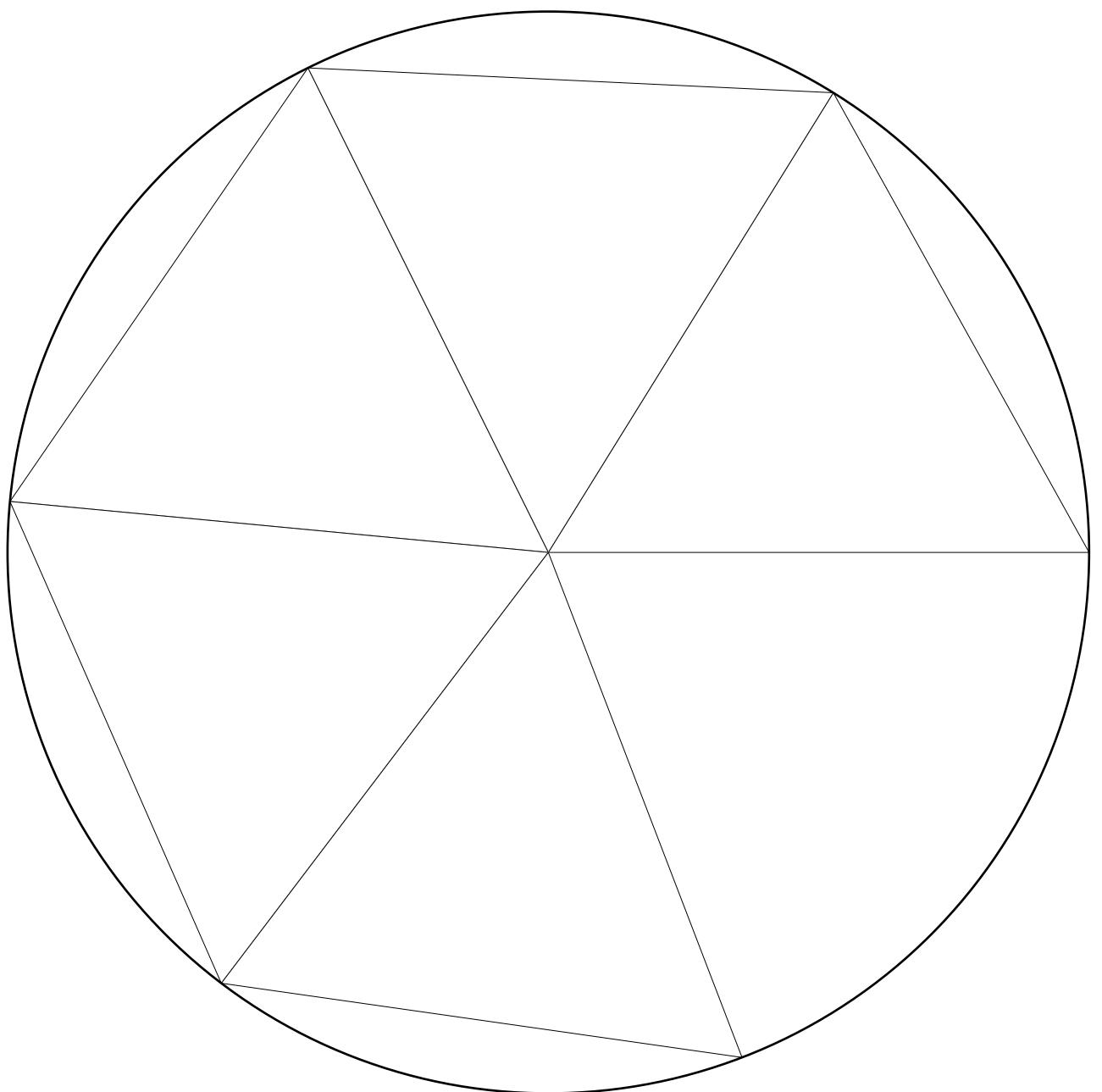


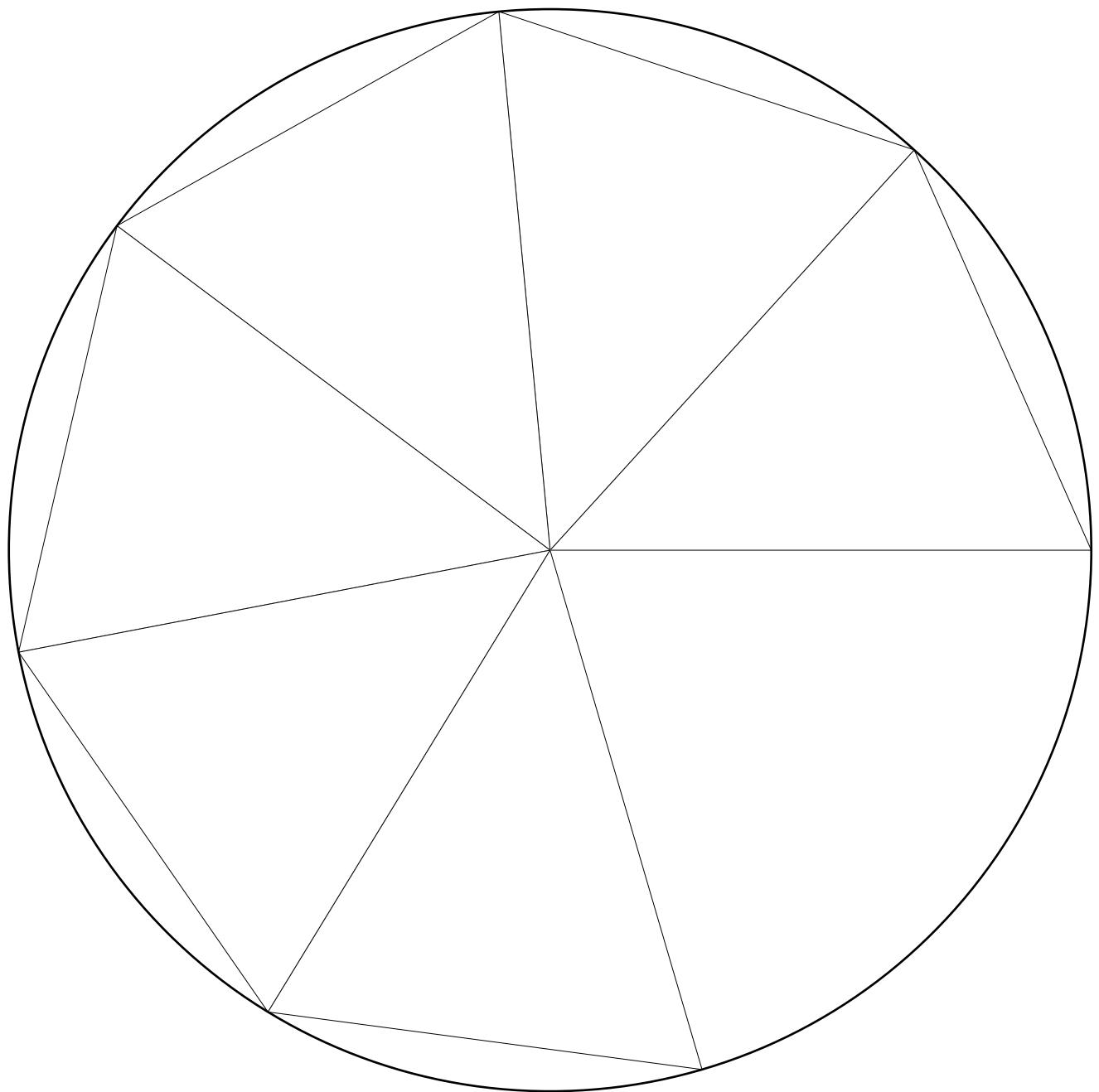


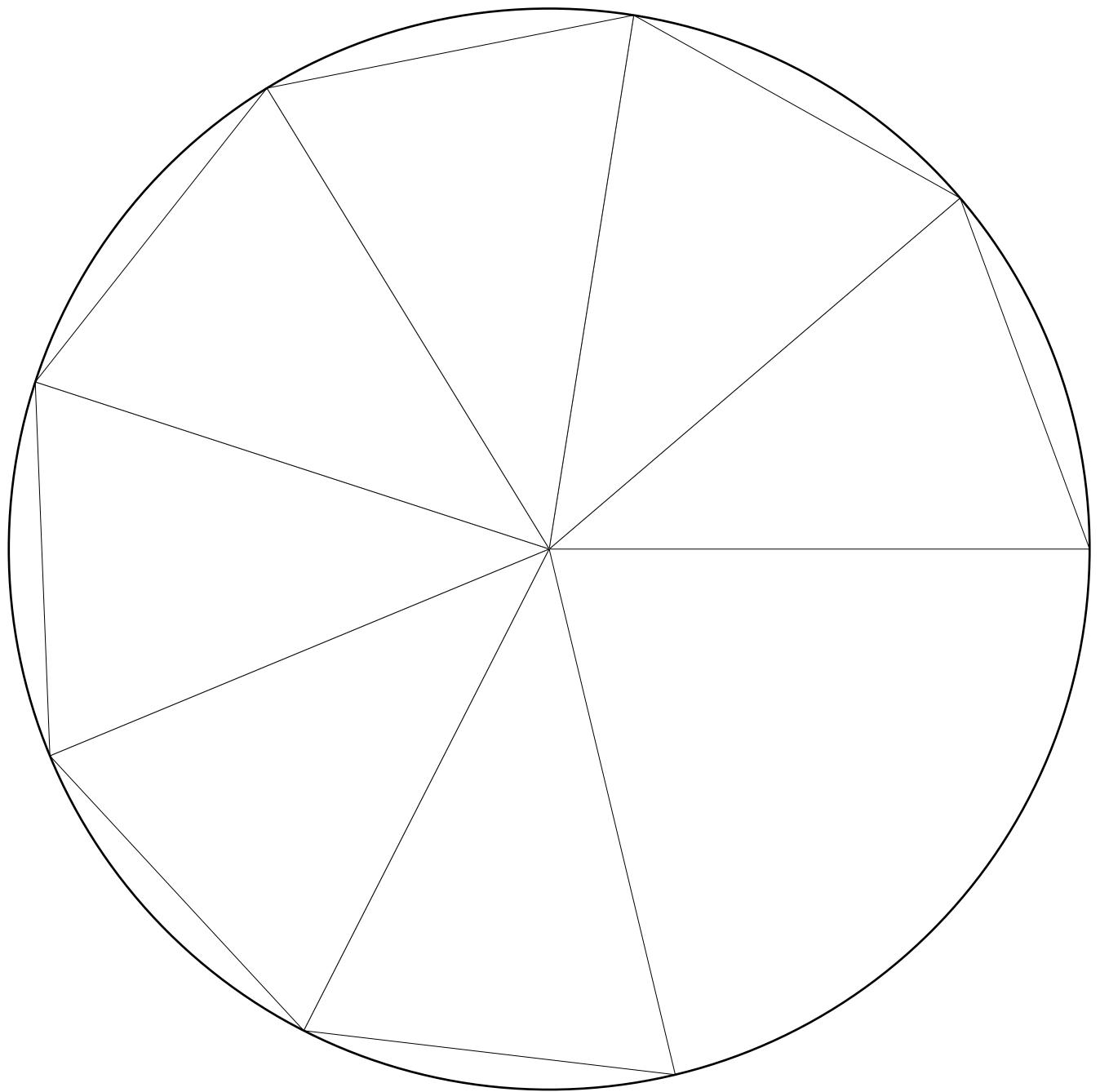


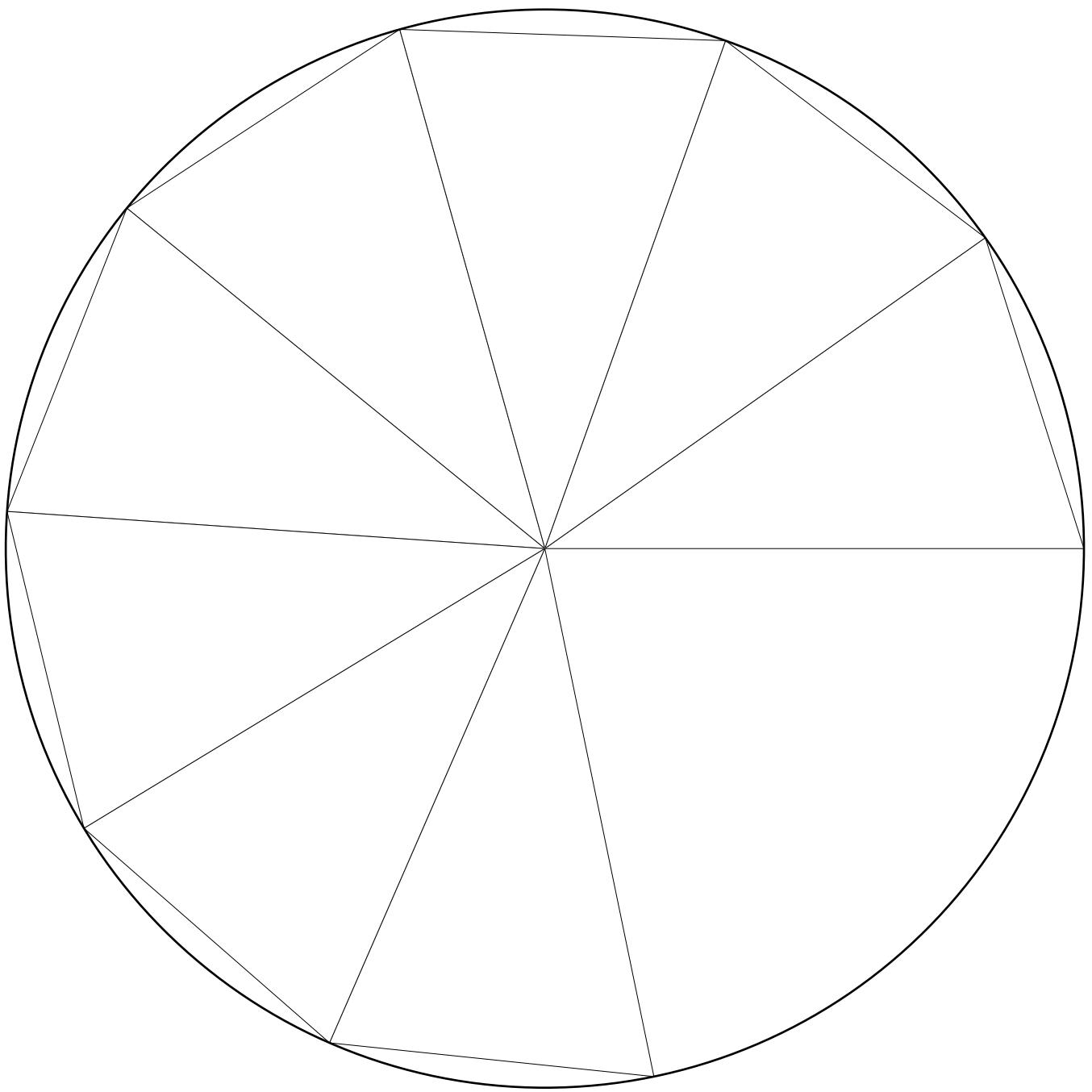


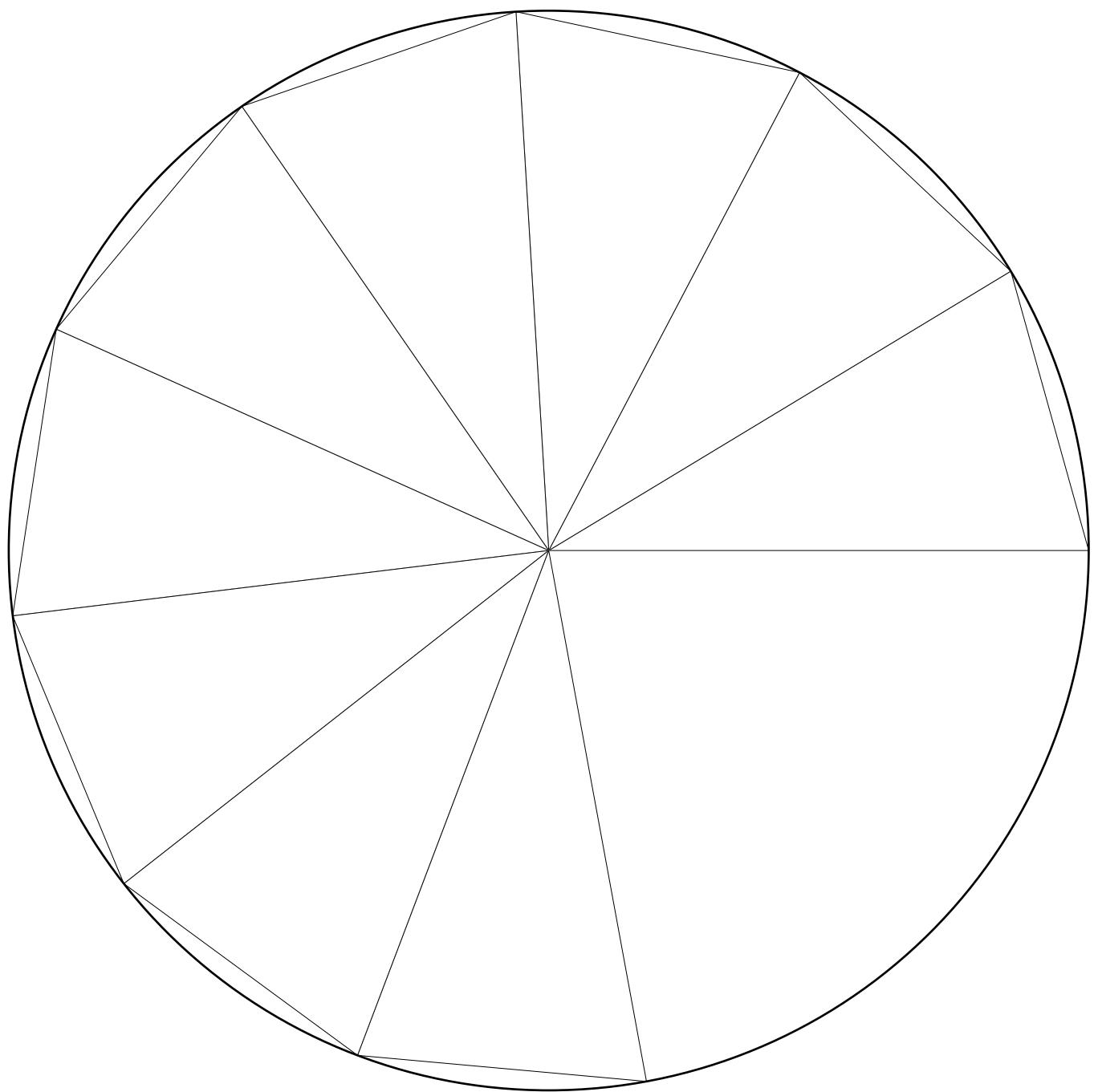


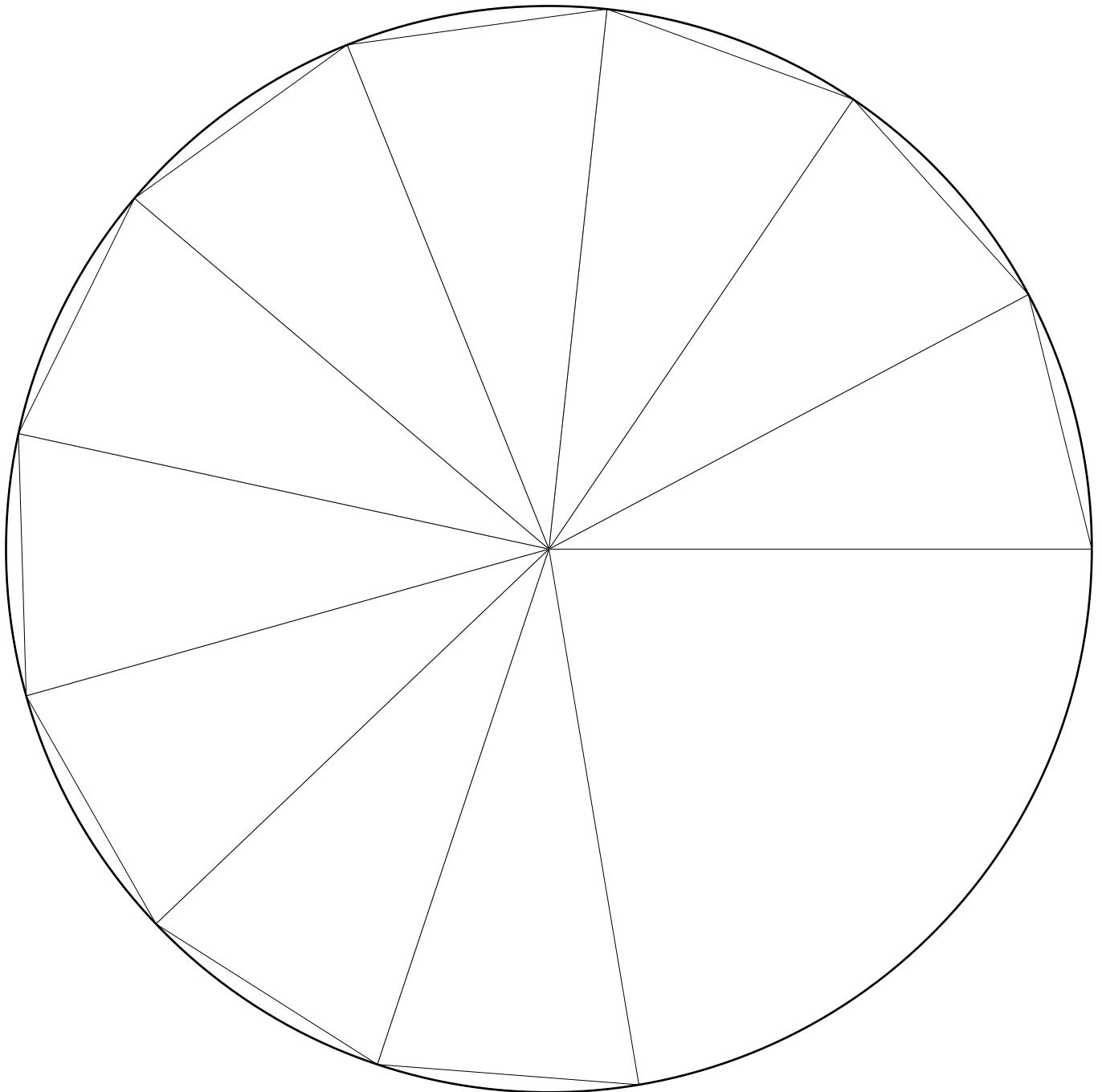


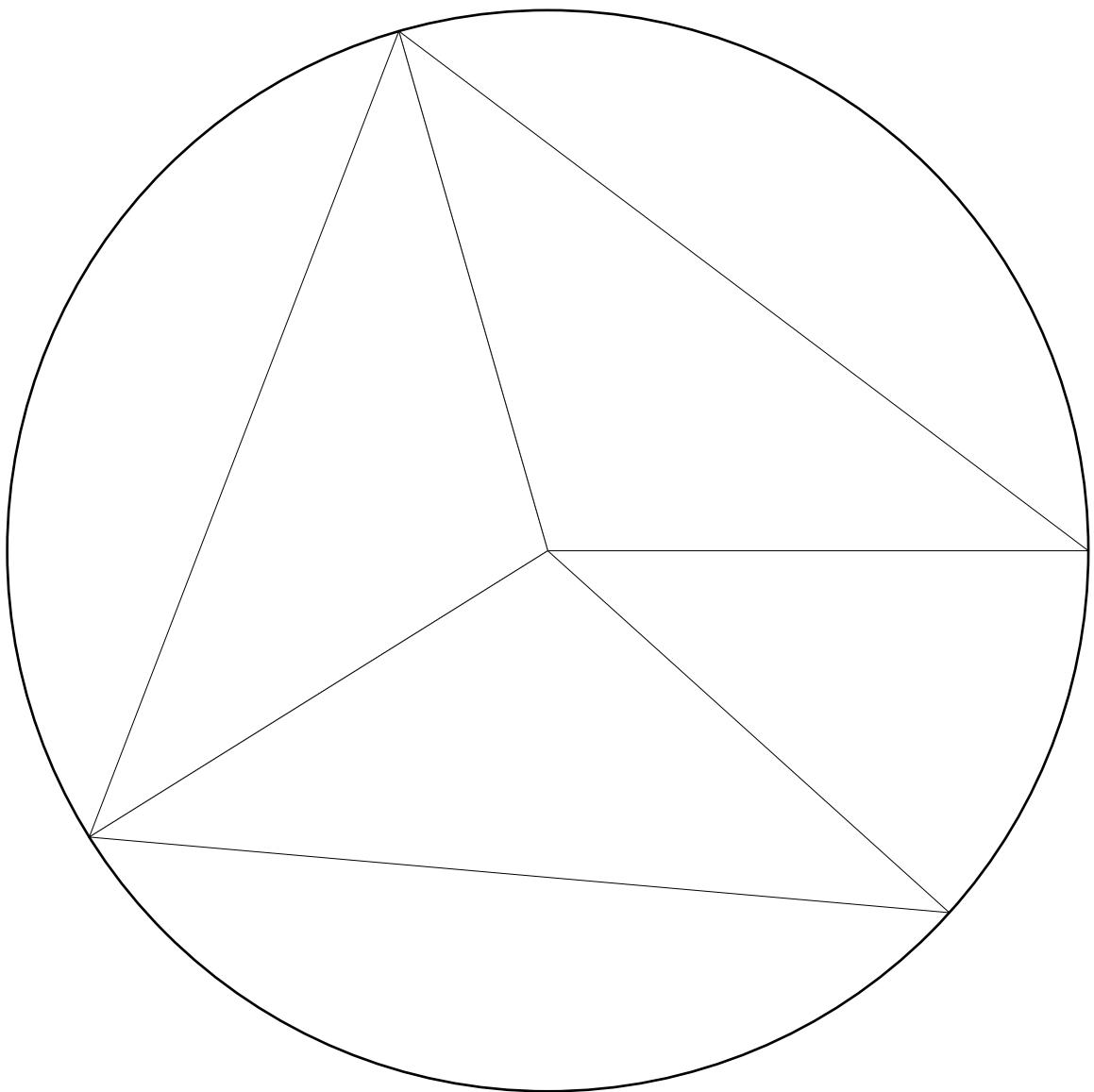
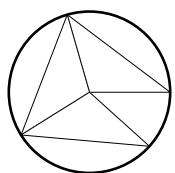
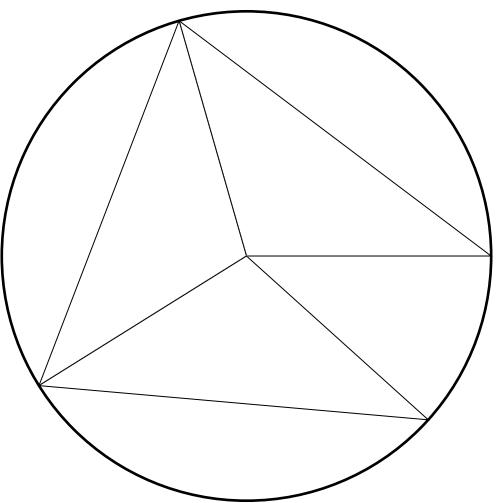
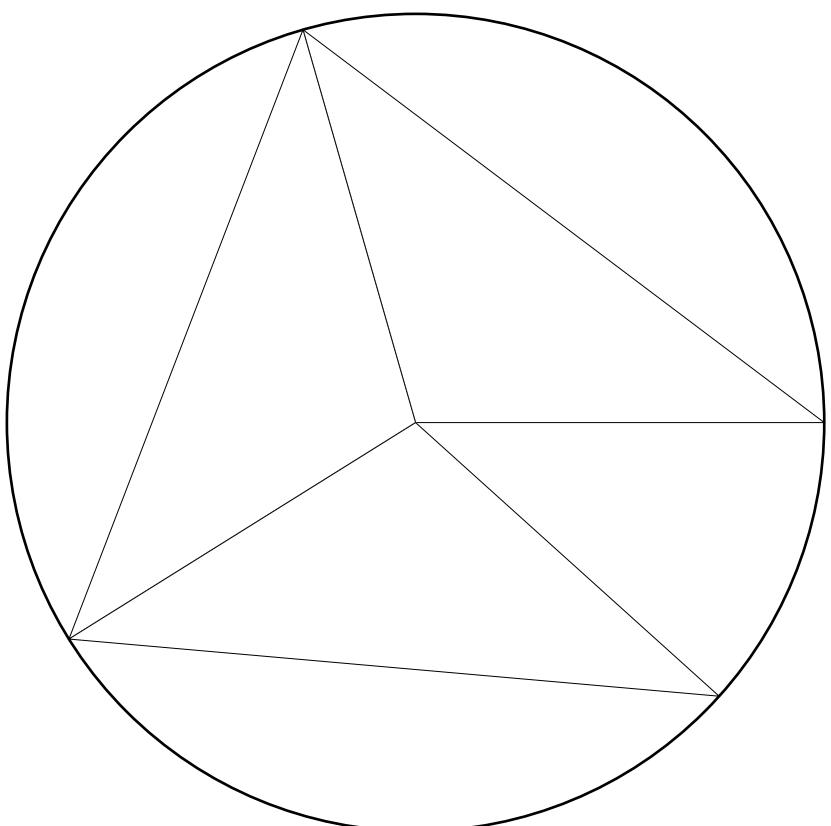


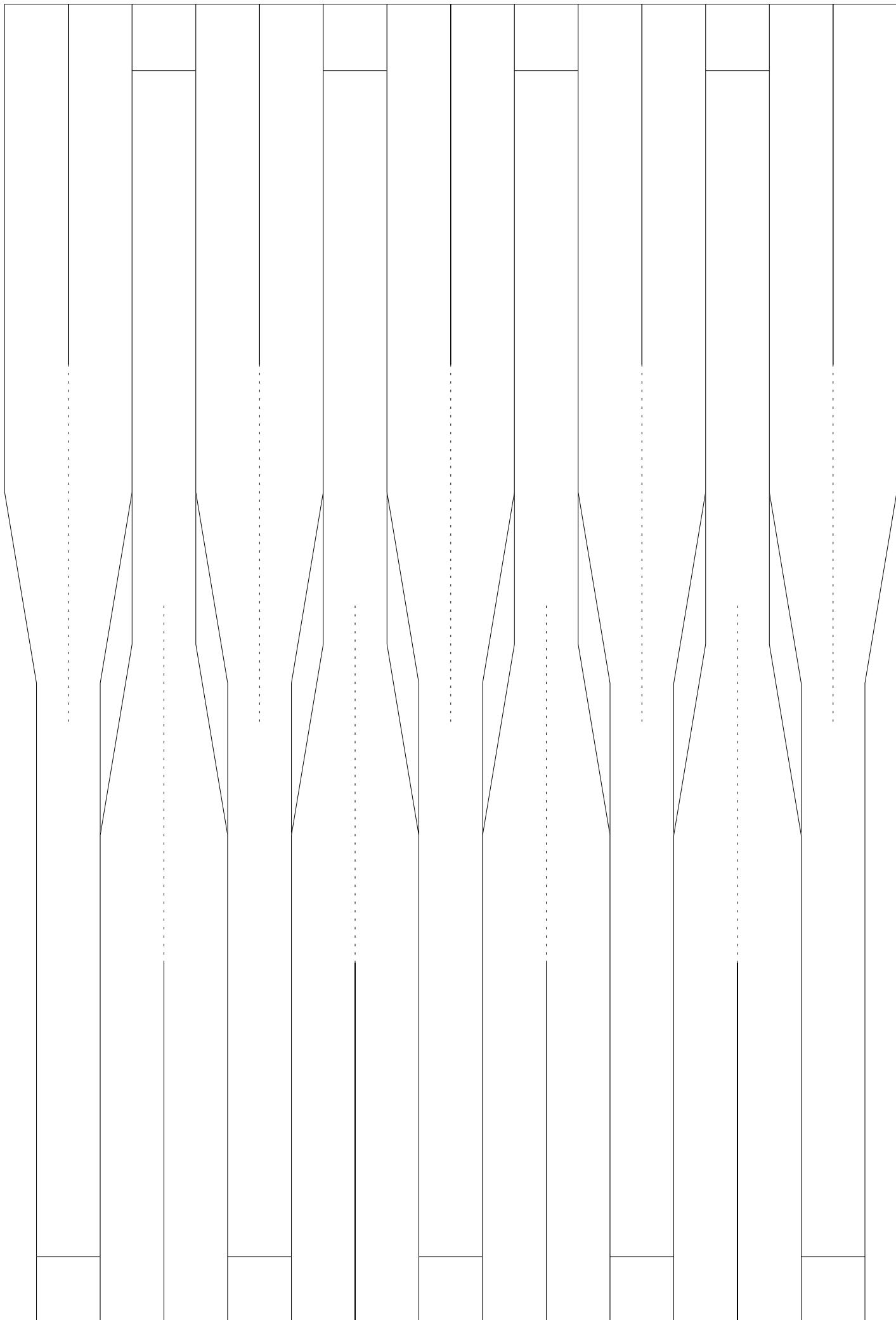


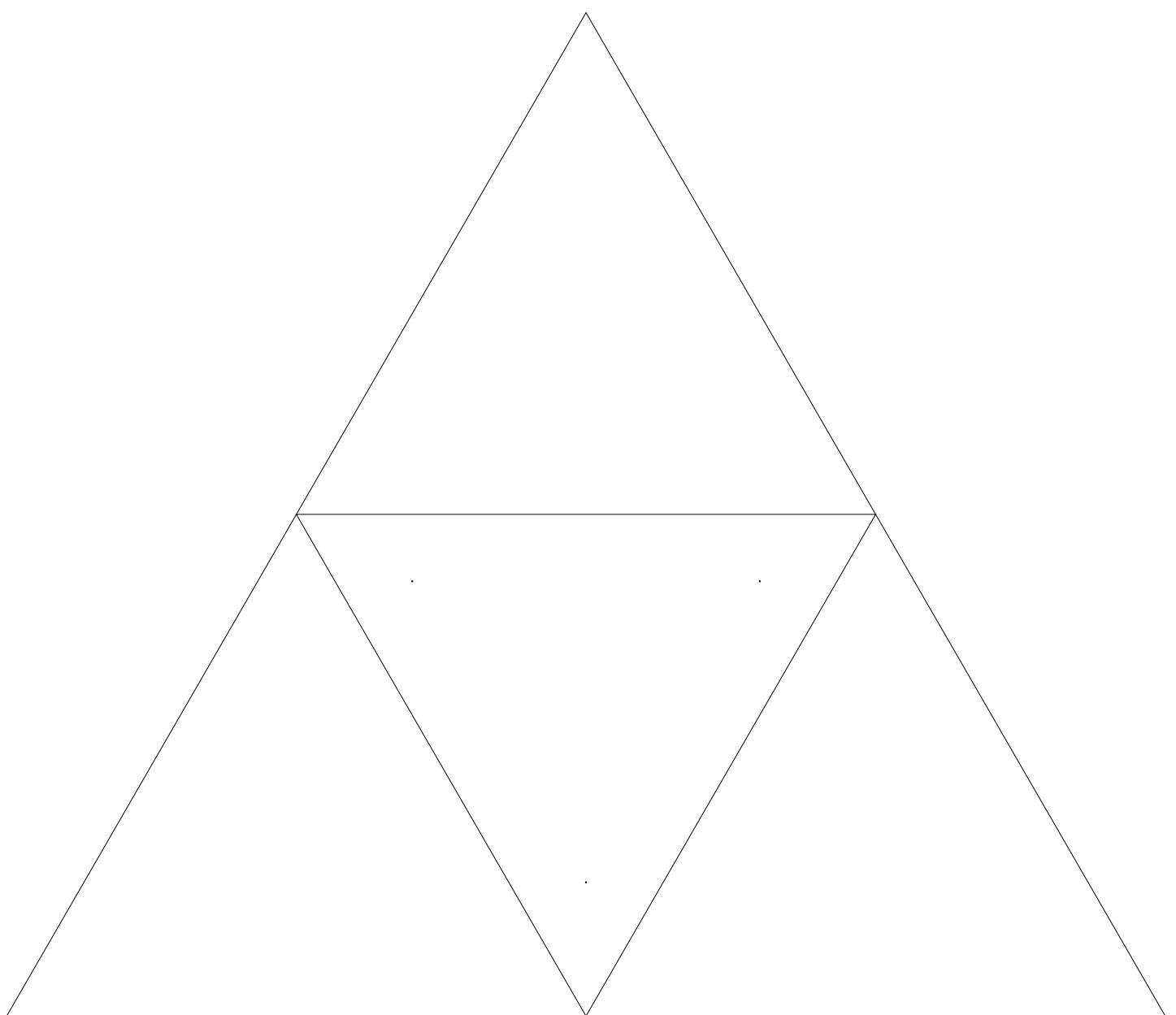












9.3 PROGRAMI IN OSTALE PRILOGE

Če bi bili programi priloženi k raziskovalni nalogi, bi jih bilo več strani, kot vseh ostalih. Iz praktičnosti so programi naloženi na spletni strani Github, na spodnji povezavi:

<https://bit.ly/3a4x5Wn>