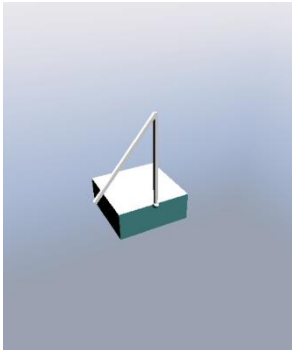
Dynamische systemen - Project 2 - Wiskunde 2.2

Pole-Placement

Nathan Isaac – 500899683
Dennis Philipoom – 500904361

Introductie

Voor de tweede opdracht in dynamische systemen hebben we een zogenaamd 'stacked pendulum' systeem moeten stabiliseren binnen de kortst mogelijke tijd. Het systeem is gesimuleerd met behulp van Python en de code die door de docent is verstrekt. Het systeem kan het best worden omschreven als twee stokken die verticaal op elkaar zijn gemonteerd, maar nog steeds afzonderlijk van elkaar kunnen bewegen. Het doel is om beide stokken zo te stabiliseren dat ze rechtop blijven staan. Ons streefdoel voor een snelle stabilisatie is vastgesteld op drie seconden als benchmark. Na het vaststellen van deze benchmark streven we ernaar om de stabilisatietijd te verkorten tot minder dan drie seconden. De onderzoeksvraag luidt als volgt:



Figuur 1: Ons toegewezen systeem in een nog niet gestabiliseerde staat

"In hoeverre beïnvloedt de snelheid van stabilisatie de kwadratische inputkosten?"

Voor het stabiliseren van het systeem hebben we gebruikgemaakt van de Pole-Placement controller. Bij deze controller voer je vier gewenste waarden in, waarvan de grootste waarde als de meest dominante wordt beschouwd. Het is belangrijk dat deze waarden altijd negatief zijn om stabiliteit te garanderen. Een andere essentiële eigenschap van deze controller is dat je geen twee keer dezelfde waarde kunt invoeren. Tot slot moeten we ervoor zorgen dat we numerieke stabiliteit behouden door de aanpassing van de tijdstap. Het belangrijke aspect hierbij is dat deze waarden alleen invloed hebben op de onderste stok; we hebben geen directe controle over de stabilisatie van de bovenste stok in dit systeem.

Analyse

Bewegingsvergelijking opstellen

De correcte bewegingsvergelijkingen hebben wij verkregen van de docent, we hebben geen vergelijkingen zelf opgesteld. De vier vergelijkingen die wij gebruikt hebben staan hieronder weergegeven.

1. $I = \frac{1}{12} * \rho * d^4 * l + \frac{1}{3} * \rho * d^2 * l^3$
2. $I_{cm} = \frac{1}{12} * \rho * d^2 * l(d^2 + l^2)$
3. $u + m_p * g * l * \sin(\theta) = \left(I + \frac{1}{2} * m_p * l^2\right)\ddot{\theta} + \frac{m_p * l^2}{4} * \cos(\varphi)\left(\ddot{\varphi} + \ddot{\theta}\right) - \frac{m_p * l^2}{4} * \sin(\varphi) * \left(\dot{\varphi} + \dot{\theta}\right)$
4. $\frac{m_p * g * l}{2} * \sin(\varphi + \theta) = \left(I_{cm} + \frac{1}{8} * m_p * l^2\right) * \left(\ddot{\varphi} + \ddot{\theta}\right) + \frac{m_p * l^2}{4} * \cos(\varphi) * \ddot{\theta} + \frac{m_p * l^2}{4} * \cos(\varphi) * \dot{\theta}^2$

State space representatie

Vervolgens hebben we de gegeven vergelijkingen omgezet naar een state-space representatie door de volgende stappen te volgen, die gedetailleerd zijn beschreven in de bijgevoegde bijlage:

1. Als eerste hebben we gebruik gemaakt van de raaklijn benadering zodat
   - $sin(x)$ gelijk gesteld kan worden aan x
   - $cos(x)$ gelijk gesteld kan worden aan 1
   - $\ddot{x}^2$gelijk gesteld kan worden aan 0
   - $x\dot{x}$ gelijk gesteld kan worden aan 0
2. Daarna hebben we gezorgd dat alle variabelen geïsoleerd waren
3. Vervolgens hebben we de formules in een matrix gezet

Deze stappen leiden tot de volgende matrices:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{l^3 * {m_p}^2 * (-g+1) + 8 * I_{cm} * l * m_p}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m_p} * I & 0 & \frac{-g * l^3 * {m_p}^2}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m_p} * I & 0 \\ 0 & 0 & 0 & 1 \\ \frac{3 * l^3 * {m_p}^2 * (g-1) - 2 * I_{cm}(4 * l * m_p + g)}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m_p} * I & 0 & \frac{3 * g * l^3 * {m_p}^2 + 4 * g * l * m_p}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m_p} * I & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ \frac{8 * I_{cm} + l^2 * m}{8 * I_{cm} + 4 * I_{cm} * l^2 * m + l^2 * m} * I \\ 0 \\ \frac{-8 * I_{cm} - 3 * l^2 * m}{8 * I_{cm} + 4 * I_{cm} * l^2 * m + l^2 * m} * I \end{pmatrix}$$

Vervolgens hebben we de constanten bepaald:

- De lengte van de slingers zijn gegeven en zijn elk 0,6 meter
- De massa van de slingers is bepaald doormiddel van de formule:

$$m_p = lengte * breedte * hoogte * \rho_{hout}$$

  Waar de breedte, hoogte en de dichtheid van hout gegeven zijn en 0,02 meter en 700 kg/m^3 zijn. Hieruit volgt dan een massa van 8,4 kilogram.

- I is een gegeven formule die als volgt gaat:
  - $I = \frac{1}{12} * \rho_{hout} * breedte^4 * lengte + \frac{1}{3} * \rho_{hout} * breedte^2 * lengte^3$
- I_{cm} is ook een gegeven formule die gaat als volgt:

$$I_{cm} = \frac{1}{12} * \rho_{hout} * breedte^2 * lengte * (breedte^2 + lengte^2)$$

Systeemeigenschappen

Met deze matrix- en constanten hebben we gekeken naar het gevraagde systeemeigenschap van stabiliteit. Deze is bepaald door in de code de eigenwaardes van matrix 1 te bepalen. Deze matrix is te vinden in de bijlage met de uitwerkingen. De gevonden eigenwaardes zijn Eigenwaarden: -8,72465973; -3,97362688; 8,72465973; 3,97362688. Hieruit kun je bepalen dat de ons systeem aan het begin instabiel is. Hieruit is uit te halen dat er een controller gemaakt moet worden die kracht zet om het stabiel te maken.

Oplossing

Voor het opereren van de Pole-Placement controller moeten 4 verschillende negatieve waardes gebruikt worden. Daarom zijn wij begonnen met de waardes: [-1, -2, -3, -4]. We hebben alle parameters met een vaste stapgrootte verlaagt en genoteerd wat de kwadratische kosten waren voor dat systeem. We hebben de stapgrootte gevarieerd tussen de 0,1 en de 2,0 afhankelijk hoeveel verandering er werd waargenomen. Toen de slingers stabiel rechtop stonden binnen de benchmark van 3s zijn wij de kwadratische kosten gaan optimaliseren. Toen er bij meting 6 *(zie tabel 1)* voor het eerst een stijging in de kwadratische kosten werden geobserveerd zijn wij met kleinere stappen gaan variëren rond de parameters van meting 5. We hebben hier zowel boven als onder de parameters van meting 5 met gevarieerd totdat de kosten niet meer daalden.

Meting [#]	pole #1	pole #2	pole #3	pole #4	kosten u	t [s]	kos-ten/tijd	Verschil met me-ting 0
0	-1	-2	-3	-4	465	>10	-	-
1	-1,5	-2,5	-3,5	-4,5	388	>10	-	-
2	-2	-3	-4	-5	0,084467	>10	-	-
3	-2,5	-3,5	-4,5	-5,5	0,015445	1,730	0,008928	1,5
4	-3,5	-4,5	-5,5	-6,5	0,006558	0,372	0,017629	2,5
5	-4,5	-5,5	-6,5	-7,5	0,00579	0,274	0,02113	3,5
6	-5,5	-6,5	-7,5	-8,5	0,006159	0,214	0,028779	4,5
7	-4,6	-5,6	-6,6	-7,6	0,005797	0,268	0,02163	3,6
8	-4,4	-5,4	-6,4	-7,4	0,005793	0,282	0,020541	3,4

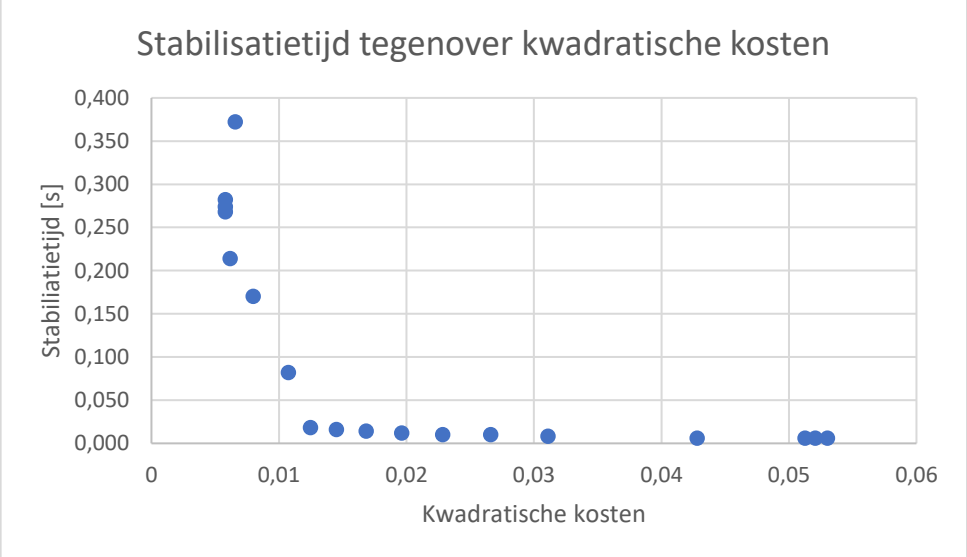
Tabel 1: Input Pole-Placement controller met als doel systeem in evenwichtstoestand binnen 3s met minimale kwadratische kosten

Vervolgens was het doel om het systeem zo snel mogelijk te stabiliseren. Hiervoor hebben wij de parameters geleidelijk veranderd en vooral rekening gehouden met de daling van de stabilisatietijd. Bij meting 19 stabiliseert het systeem sneller dan de tijdstap waarmee we het systeem simuleren. We weten daardoor de simulatietijd niet zeker. Om de laatste meetbare waarde op te zoeken worden zoeken we de parameters van meting 18 weer op en verlagen we alle parameters met 0,1. Deze stap herhalen we totdat de stabilisatietijd van het systeem weer onzeker is.

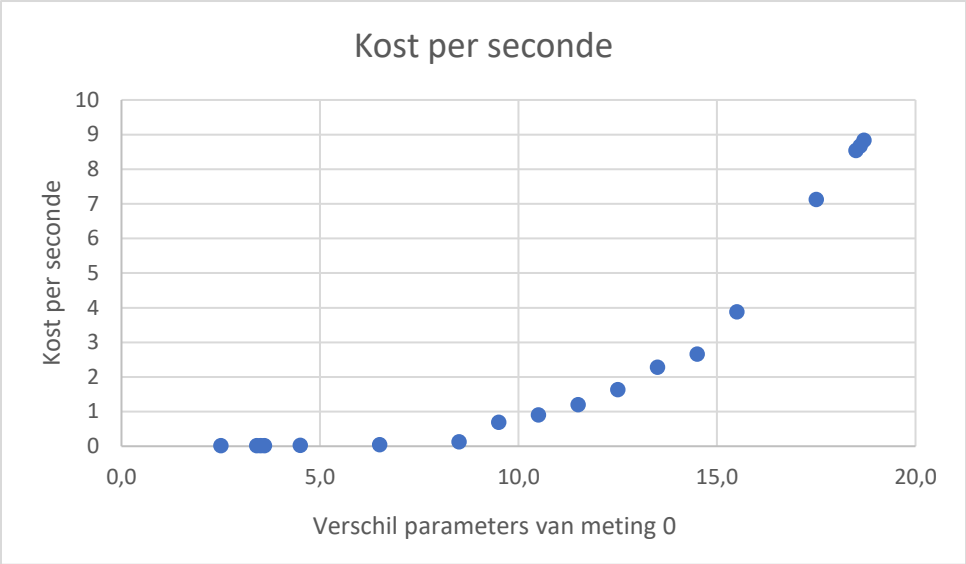
Meting [#]	pole #1	pole #2	pole #3	pole #4	kosten u	t [s]	kosten/tijd	Verschil met meting 0
9	-7,5	-8,5	-9,5	-10,5	0,007985	0,170	0,046969	6,5
10	-9,5	-10,5	-11,5	-12,5	0,010736	0,082	0,13093	8,5
11	-10,5	-11,5	-12,5	-13,5	0,012475	0,018	0,693037	9,5
12	-11,5	-12,5	-13,5	-14,5	0,014505	0,016	0,906565	10,5
13	-12,5	-13,5	-14,5	-15,5	0,016848	0,014	1,203461	11,5
14	-13,5	-14,5	-15,5	-16,5	0,019613	0,012	1,6344	12,5
15	-14,5	-15,5	-16,5	-17,5	0,022826	0,010	2,282555	13,5
16	-15,5	-16,5	-17,5	-18,5	0,026607	0,010	2,66071	14,5
17	-16,5	-17,5	-18,5	-19,5	0,031101	0,008	3,887572	15,5
18	-18,5	-19,5	-20,5	-21,5	0,042804	0,006	7,13396	17,5
19	-20,5	-21,5	-22,5	-23,5	0,061134	<0,002	-	-
20	-19,5	-20,5	-21,5	-22,5	0,051262	0,006	8,543631	18,5
21	-19,6	-20,6	-21,6	-22,6	0,052064	0,006	8,677318	18,6
22	-19,7	-20,7	-21,7	-22,7	0,05302	0,006	8,836632	18,7
23	-19,8	-20,8	-21,8	-22,8	0,053989	<0,002	-	-

Tabel 2: Input Pole-Placement controller met als doel stabilisatieperiode te minimaliseren

Bij het plotten van de stabilisatietijd tegenover de kwadratische kosten visualiseren wij metingen 0, 1, 2, 19 en 23 niet. Deze metingen komen niet in de grafiek, omdat deze metingen geen concrete stabilisatietijd hebben. Meting 3 word niet weergegeven in figuur 2, omdat deze meting een hogere stabilisatietijd en kwadratische kosten heeft dan de rest van de metingen. Door meting 3 wordt de visualisatie van de andere punten minder duidelijk.



figuur 2: Stabilisatietijd van metingen uit tabel 1 en 2 tegenover de kwadratische kosten. Exclusief metingen 0, 1, 2, 3, 19 en 23



Figuur 3: Kosten per seconde tegenover de afwijking van de parameters van de eerste meting ([-1, -2, -3, -4])

Conclusie

Als wij stabilisatieperiode tegenover de kwadratische kosten plotten word het duidelijk dat de kwadratische kosten snel stijgen als de stabilisatietijd de nul benaderd. Wij verwachten dat deze trend doorzet bij kleinere stabilisatieperiodes. Dat betekent dat als men het systeem nog sneller wil maken er een stuk meer energie in gestopt moet worden om dit te bereiken. De kost per seconde stijgt steeds sneller naar mate de parameters kleiner worden en het systeem ook meer energie verbruikt. De kracht die het systeem verricht zorgt voor minder resultaat naarmate we de parameters meer verhogen.

Volgens onze data nemen de stijgen de kosten snel na de stabilisatietijd rond de 1 seconde. Wij denken dat de waardes met een stabilisatieperiode van 1 seconde het meest optimaal zijn. Omdat er dan het snelst word gestabiliseerd over de laagste kosten. We halen het meest uit de kosten die we gebruiken.

We hadden onze tijdstap kleiner kunnen maken om te kijken hoe de kwadratische kosten zouden reageren op nog snellere stabilisatietijden. Ook hadden we het verschil tussen de ingevulde parameters kunnen variëren voor een meer gespreide meting.

Advies

Wij verwachtten dat het alleen maar meer kracht vereist om het systeem sneller te stabiliseren. Wij adviseren om niet alleen te letten op hoe snel het systeem een evenwichtstoestand bereikt, maar ook te kijken efficientie van het systeem. Wij verwachten dat een zo efficiënt mogelijk systeem nastreven wel belangrijker is dan een systeem met lage stabilisatietijden. Wij denken dat de kosten per seconde dat het systeem instabiel veel relevatie heeft voor de optimalisatie van het systeem.

Bijlage

Linearisering

De eerste formule die lineair gemaakt moet worden is:

$$u + m_p * g * l * \sin(\theta) = \left(I + \frac{1}{2} * m_p * l^2\right)\ddot{\theta} + \frac{m_p * l^2}{4} * \cos(\varphi)\left(\ddot{\varphi} + \ddot{\theta}\right) - \frac{m_p * l^2}{4} * \sin(\varphi) * \left(\dot{\varphi} + \dot{\theta}\right)$$

Als eerste gebruik je de kleine hoekbenadering waardoor $sin(x) = x, cos(x) = 1$ en $x * \dot{x} = 0$:

$$u + m_p * g * l * \theta = \left(I + \frac{1}{2} * m_p * l^2\right)\ddot{\theta} + \frac{m_p * l^2}{4} * \left(\ddot{\varphi} + \ddot{\theta}\right)$$

Dan werk je de haakjes weg van de tweede orde differentiaalvergelijkingen en kom je op:

$$u + m_p * g * l * \theta = \left(I + \frac{1}{2} * m_p * l^2\right)\ddot{\theta} + \frac{m_p * l^2}{4} * \ddot{\varphi} + \frac{m_p * l^2}{4} * \ddot{\theta}$$

Als je dit dan vervolgens weer binnen haakjes zet kom je op:

$$u + m_p * g * l * \theta = \left(I + \frac{1}{2} * m_p * l^2 + \frac{m_p * l^2}{4}\right)\ddot{\theta} + \left(\frac{m_p * l^2}{4}\right) * \ddot{\varphi}$$

Dan als laatste isoleer je alle termen van de tweede orde en kom je op:

$$\left(I + \frac{1}{2} * m_p * l^2 + \frac{m_p * l^2}{4}\right)\ddot{\theta} + \left(\frac{m_p * l^2}{4}\right) * \ddot{\varphi} = u + m_p * g * l * \theta$$

Dan is de eerste formule lineair.

Daarna de tweede formule is als volgt:

$$\frac{m_p * g * l}{2} * \sin(\varphi + \theta) = \left(I_{cm} + \frac{1}{8} * m_p * l^2\right) * \left(\ddot{\varphi} + \ddot{\theta}\right) + \frac{m_p * l^2}{4} * \cos(\varphi) * \ddot{\theta} + \frac{m_p * l^2}{4} * \cos(\varphi) * \dot{\theta}^2$$

Als eerste gebruik je weer de kleine hoekbenadering waardoor $sin(x) = x, cos(x) = 1$ en $\dot{\theta}^2 = 0$:

$$\frac{m_p * g * l}{2} * (\varphi + \theta) = \left(I_{cm} + \frac{1}{8} * m_p * l^2\right) * \left(\ddot{\varphi} + \ddot{\theta}\right) + \frac{m_p * l^2}{4} * \ddot{\theta}$$

Dan werk je de haakjes weg waardoor je krijg:

$$\frac{m_p * g * l}{2} * \varphi + \frac{m_p * g * l}{2} * \theta = \left(I_{cm} + \frac{1}{8} * m_p * l^2\right) * \ddot{\varphi} + \left(I_{cm} + \frac{1}{8} * m_p * l^2\right)\ddot{\theta} + \frac{m_p * l^2}{4} * \ddot{\theta}$$

Dit kan je dan herschrijven naar:

$$\frac{m_p * g * l}{2} * \varphi + \frac{m_p * g * l}{2} * \theta = \left(I_{cm} + \frac{1}{8} * m_p * l^2\right) * \ddot{\varphi} + \left(I_{cm} + \frac{1}{8} * m_p * l^2 + \frac{m_p * l^2}{4}\right)\ddot{\theta}$$

En dan is de tweede formule lineair.

Matrix opstellen

De matrix wordt opgesteld volgens de volgende opstelling:

$$\mathrm{M1} * \dot{\vec{\mathrm{x}}} = \mathrm{M2} * \vec{\mathrm{x}} + \mathrm{M3} * \vec{\mathrm{u}}$$

Waar de vector $\vec{\mathrm{x}}$ er als volgt uitziet:

$$\vec{\mathrm{x}} = \begin{pmatrix} \theta \\ w \\ \varphi \\ \sigma \end{pmatrix}$$

Matrices eruit zien als volgt:

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & I + \frac{1}{2} * m_p * l^2 + \frac{m_p * l^2}{4} & 0 & \frac{m_p * l^2}{4} \\ 0 & 0 & 1 & 0 \\ 0 & I_{cm} + \frac{1}{8} * m_p * l^2 + \frac{m_p * l^2}{4} & 0 & I_{cm} + \frac{1}{8} * m_p * l^2 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ m_p * l * g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{m_p * g * l}{2} & 0 & \frac{m_p * g * l}{2} & 0 \end{pmatrix}$$

$$M_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Vervolgens wil je het omschrijven naar de volgende formule

$$\dot{\vec{\mathrm{x}}} = A * \vec{x} + B * \vec{u}$$

Waarin:

$$A = {M_1}^{-1} * M_2$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ \frac{-g * l^3 * {m_p}^2 + l^3 * {m_p}^2 + 8 * I_{cm} * l * m_p}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m_p} * I & 0 & \frac{-g * l^3 * {m_p}^2}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m} \\ 0 & 0 & 0 \\ \frac{3 * g * l^3 * {m_p}^2 - 3 * l^3 * {m_p}^2 - 8 * I_{cm} * l * m_p + 4 * g * l * m_p}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m_p} * I & 0 & \frac{3 * g * l^3 * {m_p}^2 + 4 * g * l * m_p}{8 * I_{cm} + 4 * I_{cm} * l^2 * m_p + l^2 * m} \end{pmatrix}$$

$$B = {M_1}^{-1} * M_3$$

$$B = \begin{pmatrix} 0 \\ \frac{8 * I_{cm} + l^2 * m}{8 * I_{cm} + 4 * I_{cm} * l^2 * m + l^2 * m} * I \\ 0 \\ \frac{-8 * I_{cm} - 3 * l^2 * m}{8 * I_{cm} + 4 * I_{cm} * l^2 * m + l^2 * m} * I \end{pmatrix}$$

Verantwoording

Er is gebruikt gemaakt van ChatGPT bij het assisteren van het schrijven van de code voor het achterhalen van de stabilisatietijd. Hieronder de prompt die gebruikt is.

- Does this code work for printing the points where my plot intersects one of the axhlines? *geschreven code*