Simulering og modellering af robotter og mennesker

Af **Kenny Erleben,** DIKU

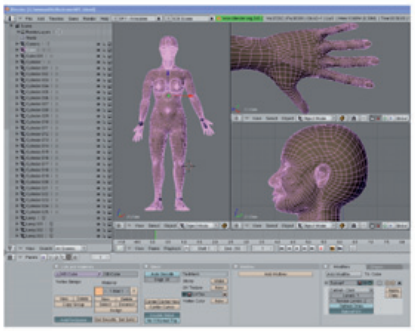
Simulering og modellering af robotter og mennesker er vel nok et af de ældste emner inden for datalogien. Simulering anvendes i dag i mange sammenhænge, fx når animatoren skal få en figur til at bevæge sig i et computerspil eller en film, eller når ingeniøren skal lave en opskrift på, hvordan en robot skal bevæge sig. Datalogen kommer således både animatoren og ingeniøren til hjælp, idet begge bruger computerprogrammer til at modellere og simulere hhv. robottens og menneskets bevægelse. Det er nemlig datalogen, som har skabt de modeller og algoritmer, der bruges i computerprogrammerne.

Ovenstående er et godt eksempel på, at datalogi er et tværfagligt fag. Datalogen bygger modeller af de problemer, der skal løses. Som led i problemløsningen kommer datalogen typisk i nærkontakt med matematik og andre fag såsom biologi, kemi og fysik. I denne artikel gives et eksempel på, hvordan en datalog modellerer ved hjælp af matematikken. Vi vil kigge på fænomenet **invers kinematik.** Invers kinematik danner grundlag for mange værktøjer, der bruges inden for faglige felter som robotik, biomekanik, computergrafik og computeranimation. Eksempler kan ses i figur 1.

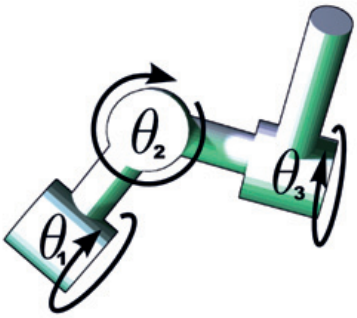
Invers kinematik bruges til at beregne, hvordan et menneske eller en robot skal placere sig for at kunne nå en ting som fx et stykke værktøj eller en kop kaffe. Før man kan begynde at løse det inverse kinematiske problem, må man bygge en model, der udregner, hvor hænderne og fødderne vil befinde sig, afhængigt af hvor meget man bøjer knæ og albuer. Når vi har denne model, kan vi løse problemet, nemlig: Givet hvor hænder og fødder er, hvor meget skal vi så bøje albuer og knæ for at kunne nå genstanden? Denne tilgang illustrerer den inverse problemstilling, også betegnet invers kinematik som er en matematisk beskrivelse af bevægelse, uafhængigt af bevægelsens årsag. Man opstiller bevægelsesligninger, der beskriver forholdene. Det centrale begreb er position opfattet som en funktion af tiden.

En matematisk model af et menneske

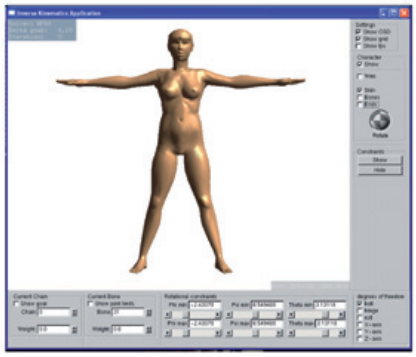
Et menneske er en meget kompleks organisme. Hvis vi ønsker kun at beskrive, hvordan et menneske bevæger sig i en afgrænset sammenhæng, kan vi bruge en simplere model. Hvis vi kun vil vide, hvorvidt en person har hånden oppe i luften, eller hvor meget en arm eller et knæ er bøjet, kan vi nøjes med at modellere et menneske som et skelet af stive stænger. Stængerne er sat sammen dér, hvor et menneske har et led. Overarmen og underarmen er forbundet af albueleddet, som rent mekanisk kan opfattes som et hængselled. Vi kan nu beskrive, hvordan underarmen er placeret i forhold til overarmen ved at beskrive vinklen imellem overarmen og underarmen. Tilsvarende kan vi opfatte et skulderled eller et hofteled som et kugleled, der hæfter overarmen fast på overkroppen og overlåret fast på underkroppen. Figur 2 viser et eksempel.



(a)



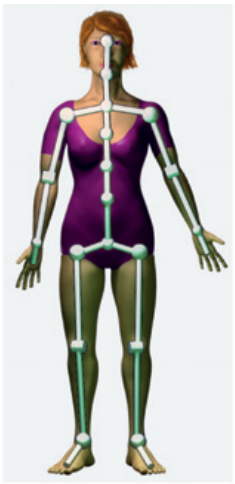
(b)



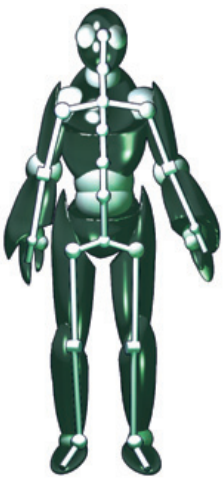
(c)

(d)

Figur 1: Forskellige eksempler på brug af modellering og simulering af mennesker. I (a) ses et animationsprogram, der blev brugt til at lave filmen Big Buck Bunny med. I (b) ses, hvordan en robot kan beskrives ud fra sin kinematik, hvorimod (c) viser en invers kinematik-software udviklet på DIKU. Endelig ses i (d) et eksempel, hvor softwaren fra (c) bruges til at opfange den menneskelige bevægelse med et kamera.

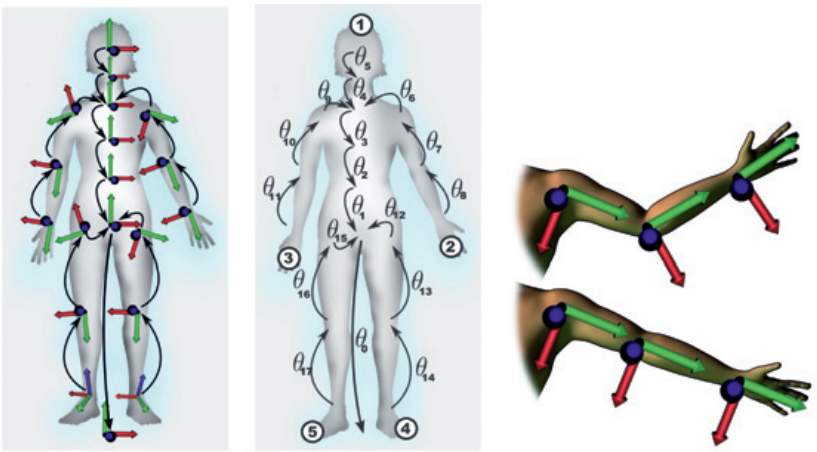


(a)



(b)

Figur 2: I (a) ses et eksempel på, hvordan et menneske kan gengives som en simpel model af stive stænger sammensat af forskellige slags bevægelige led. Bemærk i (b), at modellen også passer godt på en robot.



Figur 3: I (a) ses et eksempel på, hvordan koordinatsystemer og transformationer svarer til figur 2(a).
I(b) ses en indeksering af led og kæder af led.
I (c) ses et detaljeret eksempel på en arm, og hvordan koordinatsystemer kan placeres i den. Bemærk, hvordan koordinatsystemerne ændres, når armen strækkes.

Som det ses, er stangmodellen som skabt til også at beskrive fx robotter. Det problem, vi ønsker at løse, er: Givet albuevinklen og skuldervinklerne, hvor er hånden så placeret? Med andre ord, hvad er koordinaterne af modellens hænder og fødder, afhængigt af hvor meget den bøjer eller strækker sine led?

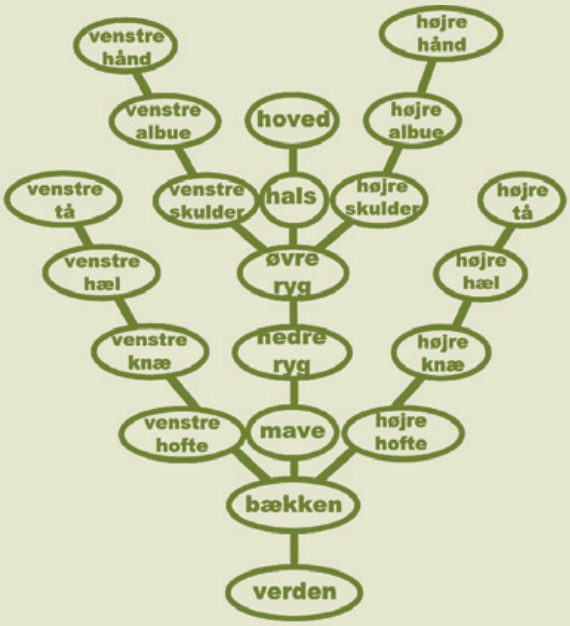
For at kunne regne effektivt med vores stangmodel af et menneske må vi udtrykke den ved hjælp af matematik. Vi benytter os af koordinatsystemer og transformationer imellem koordinatsystemer, idet vi placerer et koordinatsystem i hvert bevægelsesled af vores stangmodel, hvor de stive stænger beskriver, hvordan koordinatsystemerne er placeret i forhold til hinanden. Det er stangen, der beskriver, hvordan vi skal transformere et koordinatsystem i den ene ende af stangen, således at det transformerede koordinatsystem er sammenfaldende med koordinatsystemet i den anden ende af stangen. Figur 3 illustrerer idéen med at bruge koordinatsystemer og transformationer.

Vi introducerer nu en bestemt rækkefølge af vores koordinatsystemer og en entydig retning af vores koordinattransformationer. Først udvælger vi en stang i vores model, som vi kalder roden. I princippet kan man vælge vilkårligt, men for mennesker vil man typisk vælge den stang, som svarer til bækkenet på et menneske. Vi definerer nu en rækkefølge ved rekursivt at følge de stænger, der er forbundet til roden. Dette gør vi ved hele tiden at følge en stang, som vi ikke har set før, og som er fastgjort til en stang, vi allerede har kigget på. Når vi ikke kan komme videre fra en stang, har vi fundet det, der kaldes en slutstang. På denne måde skaber vi en mængde sammenkædede stænger, der alle går fra en rod til en slutstang. Med koordinattransformationerne beskriver vi, hvordan et koordinatsystem i retning mod slutstangen kan transformeres til koordinatsystemet i retning af roden.

Når mennesket bevæger sig rundt i verden, sker det ved, at vi flytter roden rundt i verden. Vi tilføjer derfor et ekstra koordinatsystem til at hjælpe os med at huske, hvor modellen er henne i forhold til verden. Det ekstra koordinatsystem angiver, hvor nulpunktet i verden er.

Vores valgte definition af rækkefølgen betyder at vores model danner en træstruktur på samme måde som et familietræ. Roden svarer til ens tip-tip-…-oldeforældre, som har startet det hele, og slutstængerne svarer til den seneste generation af børn i familien. De mellemliggende stænger kan sammenlignes med deres forældre og så videre. Med alle disse valg opnås den fordel, at vi ikke behøver at huske koordinatsystemerne, når vi senere skal bruge modellen i praksis; hver stang behøver nemlig kun at huske koordinattransformationen til sine egne forældre. Figur 4 illustrerer træstrukturen.

Transformationerne består af rotationer og translationer



Figur 4: Af illustrationen ses det, hvordan stangmodellen svarer til en træstruktur. Bemærk, at hver knude i træet blot skal huske, hvem dens forældre er for at konstruere det fulde træ.

Hver kæde af stænger svarer til en vej fra roden af træet til et af bladene. Vi er nu nået til den sidste beslutning i vores model, nemlig hvordan vi skal repræsentere koordinattransformationerne. Vi bruger matematiske udtryk for at opnå tilstrækkelig præcision. Transformationerne består af rotationer og translationer, og vi benytter en vektor- og matrixnotation til at udtrykke disse; se også eksemplet på næste side.

Eksempel på vektor- og matrixnotation samt forskellige regneoperationer

En matrix er en tabel af talværdier. Størrelsen af tabellen angiver dimensionen af matricen. Når vi regner med formler, bruger vi symboler i stedet for talværdier. Dvs., vi skriver:

$$\mathbf{A} = \begin{bmatrix} A_{11} & \dots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{M1} & \dots & A_{MN} \end{bmatrix} \tag{1}$$

Her vil **A** altså være en matrix med dimensionerne *M* x *N*. Talværdien i den *i*'te række og den *j*'te kolonne skrives som A_{ij}. Har en matrix kun én kolonne, kaldes den for en vektor. Ligesom vi kan regne med tal, kan vi også regne med matricer og vektorer. To matricer af samme dimension kan lægges sammen og trækkes fra hinanden elementvis, dvs:

$$\mathbf{C} = \mathbf{A} - \mathbf{B} \quad \text{hvor} \quad C_{ij} = A_{ij} - B_{ij} \tag{2a}$$

$$\mathbf{C} = \mathbf{A} + \mathbf{B} \quad \text{hvor} \quad C_{ij} = A_{ij} + B_{ij} \tag{2b}$$

Matricer kan også transponeres, hvilket betyder, at vi spejler dem på skrå ned langs deres midte. Det skriver vi som:

$$\mathbf{C} = \mathbf{A}^T \quad \text{hvor} \quad C_{ij} = A_{ji} \tag{3}$$

Endelig kan to matricer **A** og **B** med dimensionerne *M x K* og *K x N* også multipliceres med hinanden:

$$\mathbf{C} = \mathbf{AB} \quad \text{hvor} \quad C_{ij} = \sum_{k=1}^{K} A_{ik} B_{kj} \tag{4}$$

Bemærk, at **C** vil have dimensionen *M x N*. Endelig kan en matrix multipliceres med et enkelt tal *a*:

$$\mathbf{C} = a\mathbf{A} \quad \text{hvor} \quad C_{ij} = aA_{ij}. \tag{5}$$

Som en lille testopgave kan læseren prøve at udregne følgende lille problem.
Givet

$$\mathbf{H} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{og} \quad \mathbf{p} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{6}$$

hvad er så værdien af vektoren **q**?

$$\mathbf{q} = \frac{1}{2}\mathbf{H}^T\mathbf{p} \tag{7}$$

I et to dimensionelt koordinatsystem er et punkt $\mathbf{p} = [x\ y]$ givet ved to koordinater x- og y koordinaterne. En rotation med vinklen Θ af punktet $\mathbf{p}$ kan skrives som

$$\mathbf{p}_{\text{rot}} = \begin{bmatrix} x_{\text{rot}} \\ y_{\text{rot}} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{\mathbf{R}(\theta)} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{R}(\theta)\mathbf{p} \tag{8}$$

En translation kan skrives matematisk som

$$\mathbf{p}_{\text{trans}} = \begin{bmatrix} x_{\text{trans}} \\ y_{\text{trans}} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \underbrace{\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}}_{\mathbf{t}} = \mathbf{p} + \mathbf{t} \tag{9}$$

Vi kan kombinere både translation og rotation i en fælles matematisk notation ved hjælp af et lille matematisk trick

$$\begin{bmatrix} \mathbf{p}_{\text{rot}} + \mathbf{p}_{\text{trans}} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{R}(\theta) & \mathbf{t} \\ 0 & 1 \end{bmatrix}}_{\mathbf{T}(\theta)} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{10}$$

Tricket består i at udvide alle vores punkter med en ekstra koordinat, som altid har værdien 1. Herved kan vi skrive en koordinattransformation bestående af både en translation og rotation som en matrixmultiplikation med matricen $\mathbf{T}(\Theta)$. For hver af koordinattransformationerne svarende til pilene i figur 3 har vi præcis en matrix $\mathbf{T}(\Theta)$. Til at hjælpe os med at bevare overblikket vil vi benytte indeksnotation, således at matricen $\mathbf{T}_{ij}(\Theta_j)$ svarer til koordinattransformationen mellem den j'te stang fra den i'te række/kæde og dens forældre.

Bemærk, at Θ_j er vinklen mellem den j'te stang og dens forældre. Vi kan nu løse vores oprindelige problem: Givet alle vinkelværdier Θ_j'er, hvor befinder hænder og fødder sig så henne i verden? Rent matematisk kan dette svar skrives som:

$$\underbrace{\begin{bmatrix} \mathbf{p}_{\text{hoved}} \\ \mathbf{p}_{\text{venstre hånd}} \\ \mathbf{p}_{\text{højre hånd}} \\ \mathbf{p}_{\text{venstre fod}} \\ \mathbf{p}_{\text{højre fod}} \end{bmatrix}}_{\mathbf{g}} = \underbrace{\begin{bmatrix} (\mathbf{T}_{1,0}(\theta_0)\mathbf{T}_{1,1}(\theta_1)\mathbf{T}_{1,2}(\theta_2)\mathbf{T}_{1,3}(\theta_3)\mathbf{T}_{1,4}(\theta_4)\mathbf{T}_{1,5}(\theta_5)) \\ (\mathbf{T}_{1,0}(\theta_0)\mathbf{T}_{1,1}(\theta_1)\mathbf{T}_{1,2}(\theta_2)\mathbf{T}_{1,3}(\theta_3)\mathbf{T}_{2,6}(\theta_6)\mathbf{T}_{2,7}(\theta_7)\mathbf{T}_{2,8}(\theta_8)) \\ (\mathbf{T}_{1,0}(\theta_0)\mathbf{T}_{1,1}(\theta_1)\mathbf{T}_{1,2}(\theta_2)\mathbf{T}_{1,3}(\theta_3)\mathbf{T}_{3,9}(\theta_9)\mathbf{T}_{3,10}(\theta_{10})\mathbf{T}_{3,11}(\theta_{11})) \\ (\mathbf{T}_{1,0}(\theta_0)\mathbf{T}_{4,12}(\theta_{12})\mathbf{T}_{4,13}(\theta_{13})\mathbf{T}_{4,14}(\theta_{14})) \\ (\mathbf{T}_{1,0}(\theta_0)\mathbf{T}_{5,12}(\theta_{12})\mathbf{T}_{5,13}(\theta_{15})\mathbf{T}_{5,16}(\theta_{17})) \end{bmatrix}}_{\mathbf{F}(\theta_1,\ldots,\theta_n)} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{11}$$

En animeret 3d-model af det menneskelige skulderkompleks. Datasættet stammer fra et forskningsprojekt om muskelsimulation.

Her har vi udnyttet, at hænder, fødder og hoved befinder sig i nulpunktet af deres eget koordinatsystem, hvilket skrives som $[0\ 0\ 1]^T$. Vi har også udnyttet, at kæderne deler nogle af deres transformationer, f.eks er $\mathbf{T}_{1,1}(\Theta_1) = \mathbf{T}_{2,1}(\Theta_1)$ og så fremdeles. Vores idealiserede matematiske model af et menneske kan nu skrives kompakt i en ligning som:

$$\mathbf{g} = \mathbf{F}(\theta_1, \ldots, \theta_n) \tag{12}$$

Det inverse problem

Vores model fra ligning (12) kan bruges til at finde ud af, hvor hænder og fødder er henne. Det inverse problem består i at finde ud af, hvordan knæ og albuer skal bøjes for at få figurens hænder til at nå et mål, fx at samle en kop op fra et bord. Det er af stor værdi at kunne løse dette problem, idet det indebærer, at animatoren så blot behøver at fortælle computerprogrammet, at hans figur skal samle en kop op - så finder programmet selv ud af, hvordan figuren skal stå. På samme måde kan en robot bruge det inverse problem til at finde ud af, hvordan den skal styre sine motorer, så den kan samle et værktøj op.

Det er ikke altid praktisk muligt at finde den inverse funktion

Rent matematisk kan vi beskrive vores problem som: Givet **g** ønsker vi at finde alle Θ_j'er, således at:

$$\underbrace{\begin{bmatrix}\theta_1 & \ldots & \theta_n\end{bmatrix}^T}_{\theta} = \mathbf{F}^{-1}(\mathbf{g}) \tag{13}$$

hvor $\mathbf{F}^{-1}(\bullet)$ er den inverse funktion af $\mathbf{F}(\bullet)$. Det vil sige, at $\mathbf{g} = \mathbf{F}(\mathbf{F}^{-1}(\mathbf{g}))$.
Vores problem er altså nu at finde den inverse funktion $\mathbf{F}^{-1}(\bullet)$. De fundne Θ_j-værdier kaldes for en løsning og skrives som Θ^*. Stjernenotationen angiver, at der er tale om en løsning.
Umiddelbart er dette et temmelig svært problem at løse, og det er ikke altid praktisk muligt at finde den inverse funktion, da den ikke nødvendigvis eksisterer eller er entydig. Hvad værre er, det er ikke sikkert, at det ønskede **g** rent faktisk kan nås. I sådanne tilfælde forsøger vi i stedet at finde de Θ_j'er, som bringer os tættest muligt på **g**. Så i stedet for at løse ligning (13) løser vi et lidt andet problem, der om ikke andet har de samme løsninger som problemet i ligning (13). Som alternativ ønsker vi at finde de Θ_j'er, som giver den mindst mulige afstand mellem det givne **g** og $\mathbf{F}(\Theta_1,\ldots,\Theta_n)$. Dette kan skrives som:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \|\mathbf{g} - \mathbf{F}(\theta)\| \tag{14}$$

Vi ønsker at minimere objektfunktionen med hensyn til argumentet Θ. Afstandsfunktionen af en n-dimensionel vektor **v** er defineret som:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + \cdot + v_n^2} \tag{15}$$

Denne funktion er altid ikke-negativ og simpelt monoton, og da værdien nul af vores objektfunktion giver os den løsning, vi leder efter, kan vi kvadrere vores objektfunktion for at slippe af med kvadratroden. Herved kan vi skrive:

$$\|\mathbf{v}\|^2 = v_1^2 + \cdot + v_n^2 = \mathbf{v}^T\mathbf{v} \tag{16}$$

Vi kan ligeledes multiplicere vores objektfunktion med en vilkårlig positiv talværdi uden at ændre på løsningen. Vores endelige problemformulering bliver derfor:

$$\theta^* = \underset{\theta}{\mathbf{argmin}} \underbrace{\frac{1}{2}\left(\mathbf{g} - \mathbf{F}(\theta)\right)^T \left(\mathbf{g} - \mathbf{F}(\theta)\right)}_{f(\theta)} \tag{17}$$

Vi er nødt til at omforme problemet til noget, computeren forstår

Her er $f(\Theta)$ vores endelig objektfunktion, som vi vil bruge. Den har de samme løsninger som objektfunktionen i ligning (14). Fordelen ved f er, at den er nemmere at regne med og giver nogle mere simple ligninger.

Den diskrete løsning

Modellerne fra ligningerne (12), (13) og (17) er, hvad man kalder for idealiseringer. De er pæne matematiske modeller, der forsøger at beskrive, hvordan mennesker i den rigtige verden bevæger sig. En del af idealiseringen består i at skære ting væk, som ikke er væsentlige for det problem, vi ønsker at løse. Fx er hårfarven irrelevant i forhold til at finde ud af, hvor ens hænder befinder sig. Vi kan ikke umiddelbart indtaste en matematisk ligning som (17) i en computer, men er nødt til at omforme problemet til noget, som computeren forstår. Vi er også nødt til at beskrive, hvordan computeren rent faktisk skal løse problemet. Denne næste fase af modelleringsprocessen kaldes for en diskretisering.

Vi anvender en iterativ metode til at løse optimeringsproblemet i ligning (17). Først får vi givet et startgæt for Θ^* skrevet som Θ^0. Vi vil så forsøge at beregne et nyt og bedre gæt Θ^1, således at $f(\Theta^1) < f(\Theta^0)$ Hvis Θ^1 ikke minimerer $f(\Theta^1)$, forsøger vi at beregne endnu et gæt Θ^2 og så fremdeles. Vi bliver ved med at gætte, indtil vi finder et gæt, vi kan lide. Vi kan tænke på $f(\Theta)$ som en højdefunktion af et landskab på nøjagtig samme måde som et højdekort i et atlas over jordkloden. Når vi står med et eller andet gæt Θ^k, ønsker vi at finde en måde at ændre Θ^k på til Θ^{k+1}, således at:

$$f(\theta^{k+1}) < f(\theta^k) \tag{18}$$

Vi ved, at gradienten af en funktion i punktet Θ^k, $\nabla f(\Theta^k)$ altid peger i den retning, som funktionen vokser mest i. Se faktaboks for information om gradienten af en funktion.

Definition af en gradient

Har vi en funktion

$$f(\theta) = a\theta^2 + b\theta + c \tag{19}$$

hvor *a*, *b* og *c* er vilkårlige men konstante talværdier, kan vi differentiere f med hensyn til θ.
Det skriver vi som:

$$\frac{\partial f(\theta)}{\partial \theta} = 2a\theta + b \tag{20}$$

Har vi imidlertidig en funktion, som afhænger af flere argumenter, kaldes det en vektorfunktion og vi skriver

$$f(\theta_1, \theta_2, \ldots, \theta_n) = f(\theta) \quad \text{hvor} \quad \theta = \begin{bmatrix} \theta_1 & \theta_2 & \ldots & \theta_n \end{bmatrix}^T \tag{21}$$

Vi kan nu differentiere f med hensyn til et af dens argumenter θ_i, dvs vi udregner:

$$\frac{\partial f(\theta)}{\partial \theta_i} \tag{22}$$

Gør vi dette for alle argumenter og samler resultaterne i en vektor, så får vi det, man kalder for gradienten af funktionen. Vi skriver dette som:

$$\nabla f(\theta) = \begin{bmatrix} \frac{\partial f(\theta)}{\partial \theta_1} & \ldots & \frac{\partial f(\theta)}{\partial \theta_n} \end{bmatrix}^T \tag{23}$$

Rent intuitivt kan gradienten opfattes som en vektor, der peger i den retning, som funktionen vil vokse hurtigst i.

Når vi skal finde en bedre værdi end θ^k, kan vi kigge i den modsatte retning af gradienten. Man kan tænke på gradienten som en måling af, hvor meget landskabet skråner, lige der hvor man står, dvs. det er en lokal måling, der kun gælder i et lille område omkring ens målepunkt. Man skal derfor passe på ikke at gå for langt i den modsatte retning af gradienten. Derfor multiplicerer vi vores gradients skridt med en tilpas lille positiv værdi τ. Vores nye bud kan findes som:

Vores eneste problem er nu at finde værdien τ i ligning (24). Det gør vi ved at gætte os frem til en værdi på en intelligent måde. Vi starter med at sætte $\tau = 1$ og tester så, om dette vil give os en θ^{k+1}-værdi, der opfylder ligning (18). Vi ved, at gradienten er en lokal måling, så vi skal bare gøre

$$\theta^{k+1} = \theta^k - \tau \nabla f(\theta^k) \tag{24}$$

skridtlængden kortere og kortere, indtil vi er i et område, der er lokalt nok til, at gradienten giver en god måling. Vi halverer værdien af τ og tester igen ligning (18). Sådan fortsætter vi iterativt, indtil vi har fundet en passende τ-værdi.

Vi kan nu opskrive den fulde algoritme, som computeren bruger til at finde en løsning. Til at starte med gives en værdi for målene og et startgæt på vinklerne θ^o,

$$
\begin{array}{ll}
1: & k \leftarrow 0 \\
2: & \textbf{while} \quad \left\|\nabla f(\theta^k)\right\| > \varepsilon \quad \textbf{do} \\
3: & \quad \tau \leftarrow 1 \\
4: & \quad \textbf{while} \quad f(\theta^k - \tau \nabla f(\theta^k)) \geq f(\theta^k) \quad \textbf{do} \\
5: & \quad \tau \leftarrow \frac{1}{2}\tau \\
6: & \quad \textbf{end} \\
7: & \quad \theta^{k+1} \leftarrow \theta^k - \tau \nabla f(\theta^k) \\
8: & \quad k \leftarrow k + 1 \\
9: & \textbf{end}
\end{array}
$$

Læg især mærke til testen, der udføres i linje 2 af algoritmen. I denne linje testes der for, om gradienten i det punkt, man står i, er tilpas tæt på nul. Her svarer ε til en lille positiv talværdi, som er givet. Det er nødvendigt at gennemføre en sådan test i en computer, fordi afrundings- og præcisionsfejl gør, at man ikke vil kunne få præcis værdien nul i en udregning. Hvis gradienten af en funktion er nul, betyder det, at vi har fundet et ekstremum, dvs. enten et minimumspunkt, saddelpunkt eller maksimumpunkt. Da det netop er et minimumspunkt, der skal være vores løsning, vil algoritmen altså slutte med at gætte på en θ-værdi, som kan være et minimum. Vores algoritme er nu færdigdesignet og klar til at blive programmeret, så den kan bruges i et computerprogram, eksempelvis et, der kan beregne, hvordan mennesker og robotter skal bevæge sig for at nå et mål.

Modellering bruges i samfundet

I dag har invers kinematik udviklet sig til mange flere anvendelser, end hvad vi har beskrevet i denne introduktion til emnet. På Datalogisk Institut arbejdes der i skrivende stund med at kombinere invers kinematik-modeller med statistik i et forsøg på at gøre det muligt at lære en computer at kunne se og afkode ved hjælp af et kamera, hvordan mennesker bevæger sig. På denne måde kan vi konstruere nye og bedre måleapparater, som andre forskere igen kan bruge til at opnå endnu bedre forståelse af, hvordan vi mennesker virker.

Der findes mange andre eksempler på anvendelser. Man kunne fx lære en robot, hvordan den skal løse en opgave ved at lade den se og kopiere, hvordan et menneske foretager en eller anden bevægelse, eller man kunne konstruere en computertræner, der kan lindre ældre menneskers gigt gennem fysioterapi eller træne yngre mennesker, der har fået sportsskader. Disse samfundsgavnlige anvendelser af computerens mange muligheder skaber en meningsfyldt hverdag for os forskere på Datalogisk Institut, der arbejder sammen med læger, fysioterapeuter og idrætslærere, samtidig med at vi bruger masser af matematik og koder programmer, der hjælper andre mennesker.

Nedenfor er en oversigt over de matematiske emner, vi har brugt i modelleringsprocessen:

- Homogene koordinater
- Vektorfunktioner
- Differentiering af vektorfunktioner
- Ikke-lineær optimering
- Steepest Descent metoden. ❖

Læs mere

Robert Messer: Linear Algebra, Gateway to Mathematics, HarperCollins College Publishers
Robert A. Adams: Calculus, A complete course, Addison-Wesley
Jorge Nocedal og Stephen J. Wright: Numerical Optimization, Springer

Kenny Erleben

Kenny Erleben har været lektor på DIKU siden 2009 og forsker bl.a. i fysisk baseret simulation og anvendt matematik. Han blev kandidat i datalogi i 2001 og ph.d. i 2005. Kenny er leder af eScience-forskerskolen ved Det Naturvidenskabelige Fakultet, har været formand for OpenTissue-open source-projektet siden 2007 og modtog et NVidia-professor-partnerskab i 2008.

Kenny underviser bl.a. i computergrafik, numerisk optimering og computational physics. Af udvalgte artikler kan nævnes: "Physics-Based Animation" (2005) og "Velocity-Based Shock Propagation for Multibody Dynamics Animation" (2007).