ISSN 0280–5316
ISRN LUTFD2/TFRT--5527--SE

Matlab som "compute server" för inbyggda system – beräkningar av robotrörelser

Jonas Sonnerfeldt

Institutionen för Reglerteknik
Lunds Tekniska Högskola
Februari 1995

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
February 1995

Document Number
ISRN LUTFD2/TFRT--5527--SE

Author(s)
Jonas Sonnerfeldt

Supervisor
K. Nilsson, R. Johansson, A. Robertsson

Sponsoring organisation

Title and subtitle
Matlab som "compute server" för inbyggda system - beräkning av robotrörelser. (Matlab as a Compute Server for Embedded Systems – Trajectory Computation).

Abstract

Computation of trajectories for robot motions has been implemented using an available toolbox in Matlab. The Matlab calculations were connected to a robot control system via a computer network. The trajectory generator has been tested together with an ABB IRB-6 industrial robot. The embedded system's software has been extended to make use of the trajectories in real time. A regulator with feedforward from the torque has also been implemented. It has been experienced that the use of available advanced host computer software is convenient and powerful for rapid prototyping of embedded systems.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
Swedish

Number of pages
28

Recipient's notes

Security classification

Innehåll

1. Inledning

För att uppnå snabba och noggranna robotrörelser, så behöver man i robotens styrsystem använda en matematisk modell av robotens kinematik och dynamik. I ett industriellt robotsystem måste normalt denna typ av beräkningar vara inprogrammerade i styrsystemet på ett effektivt sätt och så att högt ställda realtidskrav uppfylls. Detta medför att en betydande ingenjörsinsats behövs om man vill kunna testa nya förbättrade algoritmer. Vid utveckling av algoritmer och modeller används idag interaktiva beräknings- och simuleringsprogram som Matlab. Så har skett också i detta arbete. För att möjliggöra snabba och enkla tester på en riktig robot, så har program utvecklats som medger att Matlab på en värddator används som en "compute server" från robotsystemet. Examensarbetet utgörs av fyra delar :

- Utvärdering av robotikpaket.
- Utveckling av en robotoberoende trajektoriegenerator i Matlabmiljö.
- Inkoppling av Matlabberäkningar till ett robotstyrsystem via datornätverk.
- Testning av trajektoriegeneratorn tillsammans med en IRB-6 industrirobot från ABB.

Uppläggningen av rapporten följer i stort sett examensarbetets utveckling: I kapitel 2 görs en utvärdering av två robotikpaket. Kapitel 3 innehåller en beskrivning av den teori som det robotikpaketet som har använts bygger på. Trajektoriegeneratorn gås igenom i kapitel 4. En översikt av IRB-6 kinematik ges i kapitel 5. I kapitel 6 beskrivs de modifieringar och tillägg som gjorts i programvaran till IRB-6 för att trajektoriegeneratorn skulle kunna testas. Slutligen finns en sammanfattning med kommentarer i kapitel 7.

2. Utvärdering av robotikpaket

Då det finns robotikpaket på marknaden och och då det är onödigt att utveckla redan existerande lösningar utnyttjades ett sådant. Ett val gjordes mellan två alternativ :

Robotica [4], som tagits fram vid Illinoisuniversitetet, är en samling funktioner utvecklade för att lösa diverse robotikproblem. Detta robotikpaket bygger på Mathematica [7] och har ett X-Windows baserat gränssnitt. Följande funktioner kan Robotica för närvarande erbjuda :

- symbolisk beräkning av direkt kinematik
- symbolisk beräkning av jakobianen
- symbolisk beräkning av dynamik m.h.a. Euler-Lagranges metod
- möjlighet att läsa in data från Simnon [11]
- animering

Robotics Toolbox [5], som tagits fram vid CSIRO i Preston är ett robotikpaket utvecklat i Matlab [6]. Robotics Toolbox innehåller liksom Robotica en mängd funktioner som är användbara om man vill lösa olika robotikproblem :

- numerisk beräkning av direkt kinematik
- numerisk beräkning av invers kinematik
- numerisk beräkning av jakobianen
- numerisk beräkning av dynamik m.h.a. Newton-Eulers metod
- trajektoriegenerering
- möjlighet att läsa in data från MEX-filer
- animering

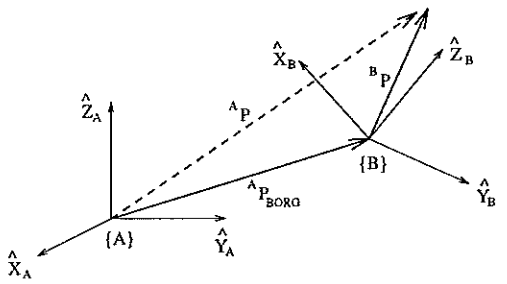
Det går egentligen inte att jämföra dessa båda paket. Robotica är ett genomarbetat och användarvänligt robotikpaket, som används vid undervisning. Robotics Toolbox däremot är en samling robotikverktyg som har strukturerats upp som en verktygslåda. Vidare är grafiken väl utvecklad i Robotica medan den i Robotics Toolbox är rudimentär. Trots detta valde jag att arbeta med Robotics Toolbox. Valet grundas på att jag tror att ett robotikpaket baserat på Matlab utgör ett flexiblare alternativ. Att Matlab är inarbetat på institutionen och att jag kunde få effektiv handledning i detta matematikprogram påverkade också mitt val. Jag har även undersökt möjligheten att använda symboliska rörelseekvationer, som beräknats i Maple [8] . Genom att använda Maples C-kodgenerator kan kod implemeneterad i Maple användas via Matlab. Ett symboliskt dynamikprogram som [9] skulle i detta sammanhang vara intressant att tillämpa. Då jag använt mig av Robotics Toolbox kommer jag endast att beskriva detta robotikpaket. Den som vill veta mera om Robotica hänvisas till [2]. Robotics Toolbox kommer dock inte att redovisas i detalj, då paketet är en rättfram implementering av robotteorin, vilken presenteras i kapitel 3. Den som vill få en bättre översikt av vad Robotics Toolbox innehåller kan studera bilaga 1 och den som vill få ett grepp om paketets struktur hänvisas till kapitel 4.

3. Robotteori

Här ges en kortfattad genomgång av av den teori som ligger till grund för Robotics Toolbox.

3.1 Direkt kinematik

Kinematik är vetenskapen som behandlar rörelsen och dess geometri utan hänsyn till de krafter som påverkar densamma. Inom robotkinematiken vill man kunna beskriva position och orientering av robotens sista länk. Ett naturligt sätt att göra detta i det kartesiska rummet är att position beskrivs med tre koordinater samt att orienteringen beskrivs med en rotationsmatris som anger hur ett till robotlänken fast koordinatsystem är roterat till ett i rummet fast referenskoordinatsystem :



Figur 3.1 En generell transformering av en vektor.

$${}^A P = {}^A_B R\, {}^B P + {}^A P_{BORG} \quad (1) \quad \text{där} \quad {}^A_B R = \textit{rotationsmatris} \ (3 \times 3)$$

Det finns alternativa sätt att beskriva rotationer . Genom att dela upp den totala rotationen i tre delrotationer kan denna beskrivas med tre vinklar. Ett sätt att göra detta är enligt följande :

$$ {}^A_B R(\gamma, \beta, \alpha) = ROT(\hat{Z}_A, \alpha) ROT(\hat{Y}_A, \beta) ROT(\hat{X}_A, \gamma) = $$

$$ \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} $$

$ROT(\hat{Z}_A, \gamma)$: rotation av B runt axel $\hat{z}_A$ med vinkeln γ
$ROT(\hat{Y}_A, \beta)$: rotation av B runt axel $\hat{y}_A$ med vinkeln β
$ROT(\hat{X}_A, \alpha)$: rotation av B runt axel $\hat{x}_A$ med vinkeln α

Vinklarna alfa,beta och gamma kallas då roll, tipp och gir.

Ett annat sätt :

$$ {}^A_B R(\alpha, \beta, \gamma) = ROT(\hat{Z}_B, \alpha) ROT(\hat{Y}_B, \beta) ROT(\hat{X}_B, \gamma) $$

$ROT(\hat{Z}_B, \alpha)$: rotationav B runt axel $\hat{z}_B$ med vinkeln α
$ROT(\hat{Y}_B, \beta)$: rotation av B runt axel $\hat{y}_B$ med vinkeln β

$ROT(\hat{X}_B, \gamma)$: rotation av B runt axel $\hat{x}_B$ med vinkeln γ

Vinklarna alfa,beta och gamma kallas då Eulers(Z-Y-X) vinklar.

Genom införandet en operator i matrisform (4×4) , en homogen koordinattransform, kan uttrycket (1) skrivas :

$$^{A}P = {}^{A}_{B}T\, {}^{B}P$$

där

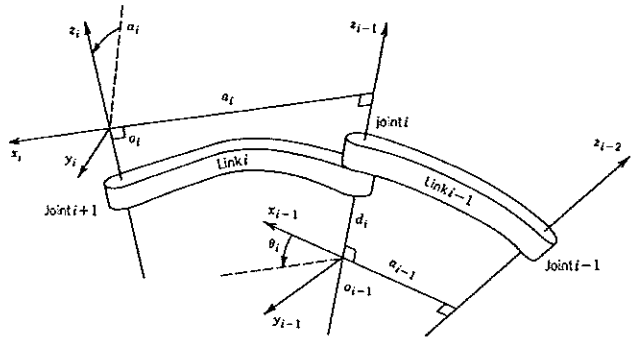
$${}^{A}_{B}T = \begin{pmatrix} & {}^{A}_{B}R & & {}^{A}P_{BORG} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

För att underlätta beskrivningen av läget och orienteringen av varje länk fästes ett koordinatsystem till varje robotled och translationen och rotationen mellan olika länkar kan då beskrivas med hjälp utav translationsmatriser :

$${}^{0}_{N}T = {}^{0}_{1}T\, {}^{1}_{2}T \ldots {}^{i-1}_{i}T \ldots {}^{N-1}_{N}T$$

Denavit och Hartenberg föreslog en metod att systematiskt fästa ett koordinatsystem vid varje länk. Denna metod som finns i två snarlika utföranden :

- Standard Denavit-Hartenbergdefinition [2]



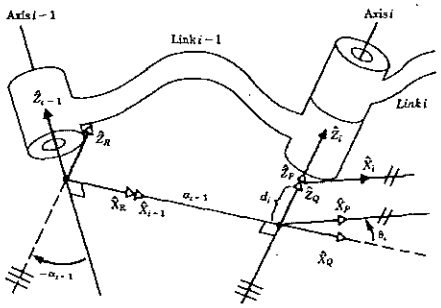
Figur 3.2 Länkparametrar och vinkeldefinitioner enligt standard Denavit-Hartenbergdefinition

a_i = avståndet längs x_i från o_i till skärningen mellan x_i och z_{i-1}
α_i = vinkeln mellan z_{i-1} och z_i mätt runt x_i
d_i = avståndet längs z_{i-1} från o_{i-1}till skärningen mellan x_i och z_{i-1}
Θ_i = vinkeln mellan x_{i-1} och x_i runt z_{i-1}

Transformationsmatriserna mellan varje koordinatsystem blir då :

$${}^{i-1}_{i}T = \begin{pmatrix} \cos\Theta_i & -\sin\Theta_i \cos\alpha_i & \sin\Theta_i \sin\alpha_i & a_i \cos\Theta_i \\ \sin\Theta_i & \cos\Theta_i \cos\alpha_i & -\cos\Theta_i \sin\alpha_i & a_i \sin\Theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Modifierad Denavit-Hartenbergdefinition [3] :



Figur 3.3 Länkparametrar och vinkeldefinitioner enligt modfierad Denavit-Hartenbergdefinition

a_{i-1} = avståndet från $\hat{Z}_i$ till $\hat{Z}_{i+1}$ mätt längs $\hat{X}_i$
α_{i-1} = vinkeln mellan $\hat{Z}_i$ och $\hat{Z}_{i+1}$ mätt runt $\hat{X}_i$
d_i = avståndet från $\hat{X}_{i-1}$ till $\hat{X}_i$ mätt längs $\hat{Z}_i$
Θ_i = vinkeln mellan $\hat{X}_{i-1}$ och $\hat{X}_i$ runt $\hat{Z}_i$

Transformationsmatriserna mellan varje koordinatsystem blir då :

$$ {}^{i-1}_{i}T = \begin{pmatrix} \cos\Theta_i & -\sin\Theta_i & 0 & a_{i-1} \\ \sin\Theta_i\cos\alpha_{i-1} & \cos\Theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\Theta_i\sin\alpha_{i-1} & \cos\Theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

3.2 Jakobianen

Jakobianen är en matris som fås om man strukturerar upp derivationen av en multidimensionell variabel :

$$ Y = \begin{pmatrix} y_1 \\ \vdots \\ y_i \end{pmatrix} \qquad F(X) = \begin{pmatrix} f_1(x_1,\ldots,x_j) \\ \vdots \\ f_i(x_1,\ldots,x_j) \end{pmatrix} $$

$$ \dot{Y} = J(X)\dot{X} \qquad J(X) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_i}{\partial x_1} & \cdots & \frac{\partial f_i}{\partial x_j} \end{pmatrix} $$

Jakobianen kan sålunda användas till att avbilda hastigheter mellen olika koordinatsystem. Men jakobianen används inte endast till att avbilda hastigheter mellan olika koordinatsystem utan har många andra användningssområden vid analys och kontroll av robotrörelser, t.ex. : planering av mjuka trajektorier, lokalisering av singulariteter, härledning av de dynamiska ekvationerna och transformering av krafter och moment mellan olika koordinatsystem.

3.3 Invers Kinematik

I den inversa robotkinematiken tar man reda på de erforderliga ledvinklarna för att beskriva läget och orientering för robotens sista länk:

$${}^0_N T(q_1, ..., q_n) = {}^0_1 T ... {}^{N-1}_N T \qquad T_{ij}(q_1, ..., q_n) = h_{ij} \qquad \text{i=1,2,3} \qquad \text{j=1,...,4}$$

$$q_k = f_k(h_{11}, ..., h_{34}) k = 1, ..., n$$

Sambandet mellan kartesiska koordinater till ledkoordinater är olinjärt och det finns inte någon generell algoritm för att lösa det inverskinematiska problemet på sluten form. Om man vill lösa inverskinematiken generellt måste man använda sig av en numerisk metod, vilket detta paket använder sig av (se kapitel 4). De flesta numeriska metoder är dock mycket långsammare än lösningar på sluten form. En annan nackdel med numeriska metoder är att man inte kan välja mellan de multipla lösningar som vanligtvis förekommer.

3.4 Dynamik

Robotdynamiken behandlar rörelseekvationerna och på vilket sätt en robot rör sig när ett vridmoment tillförs varje ledvinkel eller när externa krafter tillförs. Det finns två problem som är viktiga att lösa inom robotdynamiken :

Invers dynamik , där man löser robotens rörelseekvationer för en given rörelse för att kunna bestämma de generella krafter som behövs för att orsaka rörelsen.

$$f = M(q)\ddot{q} + V(q, \dot{q}) + G(q) \qquad (2)$$

f : $(n \times 1)$-vektor som representerar det moment eller kraft som påläggs roboten.

M(q) : $(n \times n)$-tröghetsmatris, som är symmetrisk och positivt definit.

$V(q, \dot{q})$: $(n \times 1)$-vektor som representerar de centrifugal- och Corioliskrafter som verkar på roboten.

G(q) : $(n \times 1)$-vektor som beskriver de gravitationskrafter som verkar på roboten.

Direkt dynamik , där rörelseekvationerna integreras för att bestämma de generaliserade koordinaterna som svarar mot de generaliseade krafter som tillförts. Ett vanligt sätt att lösa den direkta dynamiken är att ur (2) lösa ut accelerationen.

$$\ddot{q} = M^{-1}(q)(f - V(q, \dot{q}) - G(q))$$

$$\dot{q}(t + \Delta t) = \dot{q}(t) + \ddot{q}(t)\Delta t$$

$$q(t + \Delta t) = q(t) + \dot{q}(t)\Delta t + \tfrac{1}{2}\ddot{q}(t)\Delta t^2$$

I teorin som används anses robotlederna idealt stela (vilket är en rimlig approximation i de flesta fall). Om man vet läget för masscentrum och tröghetstensorn för varje led är massfördelningen fullständigt bestämd. Inom robotlitteraturen finns det många olika metoder utvecklade för att bestämma de dynamiska ekvationerna, men de vanligaste är Lagrange och Newton Euler. Lagrangemetoden bygger på energibetraktelser och Lagranges ekvationer. Robotics Toolbox använder sig av Newton-Eulers metod, vilken bygger på kraftbalans samt

Newtons ekvation : $F = ma_c$

Eulers ekvation : $N = {}^C I\dot{\omega} + \omega \times {}^C I\omega$

där a_c = accelerationen i masscentrum
${}^C I$ = tröghetsmomentet m.a.p. masscentrum

Algoritmen består av två delar. Först beräknas länkarnas hastigheter och accelerationer iterativt från länk1 till länk n och Newton-Eulers ekvationer används på varje länk. Sedan beräknas interaktionskrafter-och vridmoment och vridmomenten vid ledvinklarna beräknas sedan iterativt från länk n tillbaks till länk1 (se kapitel 4.3). Om man vill ta hänsyn till de friktionskrafter som uppstår när roboten rör på sig kan man använda följande modeller av friktionen :

$\tau_{friktion} = a\dot{q}$
$\tau_{friktion} = bsgn(\dot{q})$ $\quad a,\ b$ = konstanter

Genom att kombinera dessa båda modeller får man ofta en godtagbar modell av friktionen. Om en mer fullständig modell eftersträvas måste hänsyn tas till att friktionen också beror av q och att därmed sambandet (2) får följande utseende :

$$f = M(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(q,\dot{q})$$

3.5 Trajektoriegenerering

Det finns många olika sätt att få en robot att röra sig från en given position till en önskad position. Vilket val man än gör måste man se till att trajektoriebanan och hastigheterna är kontinuerliga. För att förhindra vibrationer och ryckiga rörelser krävs även att accelerationen är kontinuerlig. Om man specificerar trajektorian direkt i ledrummet så undviker man problemet med degenerationer. Denna typ av trajektorieberäkning används främst då roboten rör sig långt ifrån hinder och då minimumtid är utan betydelse. Om trajektorian istället specificeras i det kartesiska rummet får man en väldefinierad bana som är lämplig att använda i start- och finalsegment. Nackdelarna med detta sätt att specificera en trajektoria är att de beräknade kartesiska koordinaterna måste kontinuerligt transformeras till ledkoordinater och det faktum att man måste ta hänsyn till degenerationer, s.k. singulära punkter.

4. Trajektoriegenerator

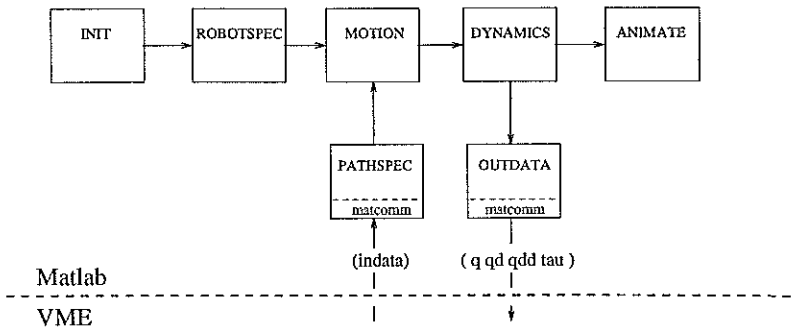
Robotics Toolbox har utnyttjats till att utveckla en trajektoriegenerator. De delar av robotikpaketet som använts är följande :

- beräkning av direkt kinematik
- beräkning av invers kinematik
- beräkning av drivtransformen för linjär rörelse
- beräkning av dynamik m.h.a. Newton-Eulers metod
- animering

Generatorn kan antingen köras i Matlab eller användas som en "compute server" för ett målsystem. Om rörelsen definieras i Matlab kan rörelsen simuleras (se kapitel 4.5). Kommunikationen med omvärlden sker i matrisform och för detta ändamål har en funktion (`matcomm`) som utvecklats på institutionen används.

4.1 Struktur

Vid implementeringen av trajektoriegeneratorn har en modulariserad struktur eftersträvats för att det ska vara enkelt att göra framtida tillägg och modifieringar. Robotikpaketet inkluderar en generell metod att representera en robots kinematik- och dynamikdata. Denna robotrepresentation definieras i Robotspec (se kapitel 4.2) och har påverkat generatorns struktur i övrigt. Data förs över mellan de olika funktionerna i matrisform.



Figur 4.1 Trajektoriegenerator i Matlab. Varje block är en m-funktion.

4.2 Robotspec

Varje robot specifiseras med hjälp av en datamatris. Robotens kinematik och dynamik beskrivs generellt med hjälp av Denavit-Hartenbergs standardkonvention. Varje rad representerar en robotlänk och varje kolumn innehåller data enligt följande :

Kolumn	*Symbol*	*Beskrivning*
1	α	vinkeln mellan z_{i-1} och $z_i$mättrunt x_i
2	a	avst. längs x_i från o_i till skärn. mellan $x_i$och z_{i-1}
3	Θ	vinkeln mellan x_{i-1} och x_i runtz_{i-1}
4	d	avst. längs z_{i-1} från o_{i-1}till skärn. mellan x_i och z_{i-1}
5	σ	ledtyp : 0 om roterande, 1 annars
6	m	länkens massa
7	rx	tyngdpkt m.a.p. länkens tillhörande koordinatsystem
8	ry	
9	rz	
10	Ixx	elementen i tröghetsmatrisen m.a.p. länkens tyngdpkt
11	Iyy	
12	Izz	
13	Ixy	
14	Iyz	
15	Ixz	
16	Jm	massströghet för ställdon m.a.p. ledens rotationsaxel
17	G	vinkelhastighet/ledhastighet
18	B	viskös friktion
19	Tc+	Coulomb-friktion (positiv rotation)
20	Tc−	Coulomb-friktion (negativ rotation)

4.3 Motion

Trajektorieberäkning

Från Pathspec fås de indata (specificeras här eller av målsystemet), som sedan används i Motion för att definiera olika rörelser. Det finns för närvarande en trajektorieberäkning i ledkoordinater och två i kartesiska koordinater. Då de trajektorieberäkningarna som följer med robotikpaketet inte ser till att hastighets- och accelerationsprofilerna är kontinuerliga implementerades trajektoriedefinitioner som finns beskrivna i [5]. Trajektorieberäkningen i ledkoordinater bygger på linjär approximation mellan de punkter som man vill att trajektorian ska passera. I start- och ändpunkter samt vid eventuella viapunkter anpassas ett fjärdegradspolynom för att försäkra sig om kontinuitet i position, hastighet och acceleration i övergången mellan de olika trajektoriesegmenten.

Om $-t_{acc} < t < t_{acc}$

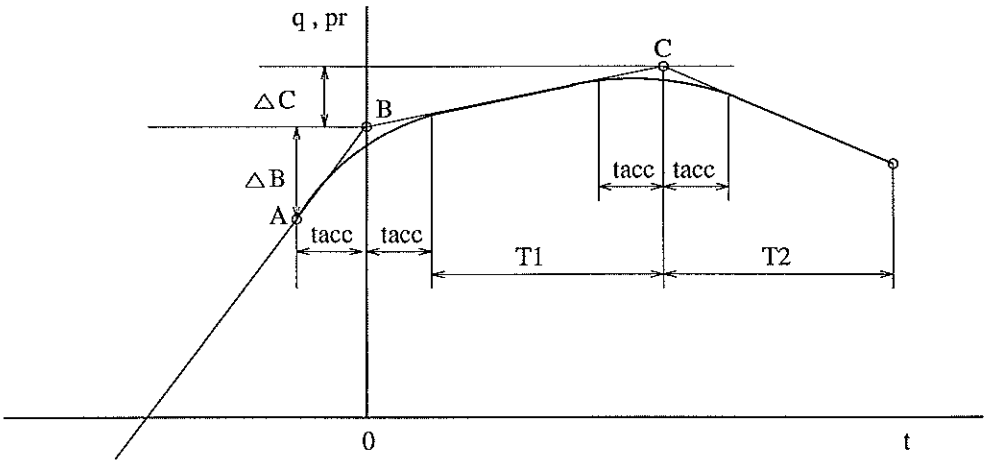
$$q = ((\Delta C \frac{t_{acc}}{T_1} + \Delta B)(2 - h)h^2 - 2\Delta B)h + B + \Delta B$$

$$\dot{q} = ((\Delta C \frac{t_{acc}}{T_1} + \Delta B)(1.5 - h)2h^2 - \Delta B)\frac{1}{t_{acc}}$$

$$\ddot{q} = (\Delta C \frac{t_{acc}}{T_1} + \Delta B)(1 - h)\frac{3h}{t_{acc}^2}$$

$$h = \frac{t + t_{acc}}{2t_{acc}}$$

Om $t_{acc} \leq t \leq T_1 - t_{acc}$



Figur 4.2 Övergångar mellan trajektoriesegment.

$$q = \Delta C h + B$$

$$\dot{q} = \frac{\Delta C}{T_1}$$

$$\ddot{q} = 0$$

$$h = \frac{t}{T_1}$$

Trajektorieberäkningen i kartesiska koordinater bygger på idén att övergången från en transform till en annan kan göras genom en translation och två rotationer. Den första rotationen ser till att vrida upp orienteringen av den sista leden till den önskade riktningen på gripdonet. Den andra rotationen är en rotationsrörelse runt gripdonet riktning, så att den önskade orienteringen nås. Genom att på detta sett dela upp rotationsrörelsen i två delar så blir spridningen på ledvinklarna förutsägbar. En rörelse från en punkt A till en punkt B kan uttryckas med hjälp av en drivtransform, D(r) :

$$s(r\Theta) = \sin(r\Theta) \qquad c(r\Theta) = \cos(r\Theta) \qquad v(r\Theta) = (1 - \cos(r\Theta))$$

$$Tr(r) = \begin{pmatrix} 1 & 0 & 0 & rx \\ 0 & 1 & 0 & ry \\ 0 & 0 & 1 & rz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Ro(r) = \begin{pmatrix} c(r\phi) & -s(r\phi) & 0 & 0 \\ s(r\phi) & c(r\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad k(r) = \begin{pmatrix} -s\Psi \\ c\Psi \\ 0 \\ 1 \end{pmatrix}$$

$$Ra(r) =$$

$$\begin{pmatrix} k_x k_x v(r\phi) + c(r\phi) & k_y k_x v(r\phi) - k_z s(r\phi) & k_z k_x v(r\phi) - k_y s(r\phi) & 0 \\ k_x k_y v(r\phi) + k_z s(r\phi) & k_y k_y v(r\phi) + c(r\phi) & k_z k_y v(r\phi) - k_x s(r\phi) & 0 \\ k_x k_z v(r\phi) - k_y s(r\phi) & k_y k_z v(r\phi) + k_x s(r\phi) & k_z k_z v(r\phi) + c(r\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

D(r)=Tr(r)Ra(r)Ro(r) $\quad T_B = T_A D_{AB}(1)$

Den första av trajektorierna som är definierad i kartesiska koordinater är en linjär rörelse. Samma algoritm som användes i ledkoordinter används här, men i detta fall är det drivtransformens parametrar som varieras (se fig). Eftersom drivparametrarna, rx, ry, rz, $r\Theta$ och $r\phi$ utgår från samma koordinatsystem T_B, kan man beskriva rörelsen genom positionerna A, B och C genom att variera drivparametrarna från $rx_A, ry_A, rz_A, r\Theta_A$ och $r\phi_A$ via 0 till $rx_C, ry_C, rz_C, r\Theta_C$ och $r\phi_C$.

$D_{BC}(1) = T_B^{-1} T_C \quad D_{BA}(1) = T_B^{-1} T_A$

A=$rx_A, ry_A, rz_A, r\Theta_A$ eller $r\phi_A$

B=0

C=$rx_C, ry_C, rz_C, r\Theta_C$ eller $r\phi_C$

Om $-t_{acc} < t < t_{acc}$

$rp = ((\Delta C \frac{t_{acc}}{T_1} + \Delta B)(2-h)h^2 - 2\Delta B)h + \Delta B$

$\Psi = (\Psi_C - \Psi_A)h + \Psi_A$

$h = \frac{t+t_{acc}}{2t_{acc}}$

Om $t_{acc} \leq$ t $\leq T_1 - t_{acc}$

$rp = \Delta C h$

$\Psi = \Psi_C$

$h = \frac{t}{T_1}$

Med hjälp av drivtransformerna beräknas de transformer som definierar rörelsen och som därefter konverteras med hjälp av invers kinematik till ledkoordinater. Hastigheter och accelerationer beräknas sedan med differensapproximationer. Den andra trajektorietypen som är definierad i det kartesiska koordinater är en godtycklig cirkulär rörelse och liksom den förra rörelsen används drivtransformen. Här används dock drivtransformen direkt, d.v.s. inget försök har gjorts för att få en "snäll" trajektoria.

Då man använder sig av inverskinematiken vid en definition av en trajektoria i kartesiska koordinater, tar beräkningen längre tid än om man definierar en trajektoria direkt i ledkoordinater. Om positioner med ett längre tidsintervall väljs och sedan en linjärinterpolation görs i ledrummet går beräkningen snabbare. Om tidsintervallet har valts lämligt så påverkar inte denna approximation inte slutresultatet märkvärt. En funktion, som utför en linjärinterpolation har implementerats.

Invers kinematik

Från robotikpaketet följde det med en funktion som löser det inverskinematiska problemet numeriskt genom att använda en iterativ metod.

En differentiell transformation kan beskrivas med hjälp av en differentiell translation och rotation :

dT=Transl(dx,dy,dz)Rot(k,dΘ)T=ΔT

$$\Delta = \begin{pmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{pmatrix}$$

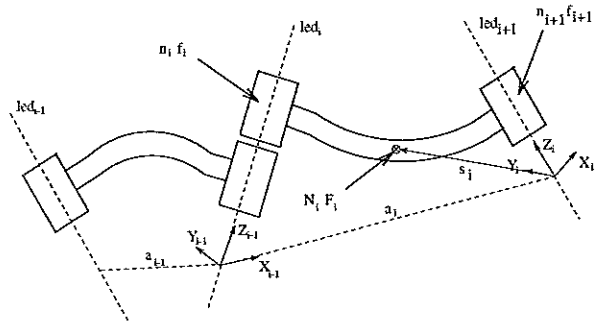
Genom att ta differensen mellan den önskade transformen och en starttransform och beräkna den vinkeldifferens detta motsvarar med hjälp av jakobianen och den differentiella rörelsevektorn, D kan man få ett nytt värde på de önskade ledvinklarna.Om man sedan med direkt kinematik beräknar den transform som motsvarar de nya ledvinklarna kan en ny differens räknas fram. På så sätt kan de önskade ledvinklarna itereras fram.

$dq = {}^{N}_{0}JD$
q=q+dq

Men då denna metod är långsam och då man inte kan välja bland de existerande lösningarna användes vid test av trajektoriegeneratorn tillsammans IRB-6 en sluten lösning (se kapitel 5.2).

4.4 Dynamics

För att beräkna vridmomenten används Newton-Eulers metod baserad på Denavit-Hartenbergs standardmetod, vilken följde med robotikpaketet :



Figur 4.3 Definition av parametrar enligt Denavit- Hartenbergs standarddefinition samt krafter och moment som verkar på en länk.

"Utåtriktade" iterationer : $1 \leq i \leq n$

Om axel i+1 är en rotationaxel :

$$^{i+1}\omega_{i+1} = {}^{i+1}R_i(^i\omega_i + z_0\dot{q}_{i+1})$$
$$^{i+1}\dot{\omega}_{i+1} = {}^{i+1}R_i(^i\dot{\omega}_i + z_0\ddot{q}_{i+1} + {}^i\omega_i \times (z_0\dot{q}_{i+1}))$$
$$^{i+1}v_{i+1} = {}^{i+1}\omega_{i+1} \times {}^{i+1}p^*_{i+1} + {}^{i+1}R_i\,{}^i v_i$$
$$^{i+1}\dot{v}_{i+1} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}p^*_{i+1} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}p^*_{i+1}) + {}^{i+1}R_i\,{}^i\dot{v}_i$$

Om axel i+1 är en translationsaxel :

$$^{i+1}\omega_{i+1} = {}^{i+1}R_i\,{}^i\omega_i$$
$$^{i+1}\dot{\omega}_{i+1} = {}^{i+1}R_i\,{}^i\dot{\omega}_i$$
$$^{i+1}v_{i+1} = {}^{i+1}R_i(z_0\dot{q}_{i+1} + {}^i v_i) + {}^{i+1}\omega_{i+1} \times {}^{i+1}p^*_{i+1}$$
$$^{i+1}\dot{v}_{i+1} = {}^{i+1}R_i(z_0\ddot{q}_{i+1} + {}^i\dot{v}_i) + {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}p^*_{i+1} + 2^{i+1}\omega_{i+1} \times ({}^{i+1}R_i z_0\dot{q}_{i+1})$$
$$+ {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}p^*_{i+1})$$

$$^i\dot{v}^c_i = {}^i\dot{\omega}_i \times s_i + {}^i\omega_i \times ({}^i\omega_i \times s_i) + {}^i\dot{v}_i$$
$$^iF_i = m^i_i\dot{v}^c_i$$
$$^iN_i = I^i_i\,{}^i\dot{\omega}_i + {}^i\omega_i \times (I^i_i\,{}^i\omega_i)$$

"Inåtriktade" iterationer : n $\geq$ i $\geq$ 1

$$^if_i = {}^iR_{i+1}\,{}^{i+1}f_{i+1} + {}^iF_i$$
$$^in_i = {}^iR_{i+1}({}^{i+1}n_{i+1} + ({}^{i+1}R_i\,{}^ip^*_i) \times {}^{i+1}f_{i+1}) + ({}^ip^*_i + s_i) \times {}^iF_i + {}^iN_i$$

$$Q_i = \begin{cases} ({}^in_i)^T({}^iR_{i+1}z_0) \text{ om axel i+1 är en rotationsaxel} \\ ({}^if_i)^T({}^iR_{i+1}z_0) \text{ om axel i+1 är en translationsaxel} \end{cases}$$

Q_i : kraft eller moment som verkar på led i

I^i_i : länk is tröghetsmoment m.a.p. tyngdpunkten

n_i : momentet som verkar på länk i p.g.a. länk *i-1*

f_i : kraften på länk i p.g.a. länk *i-1*

N_i : det totala momentet som verkar på länk i m.a.p. dess masscentrum

F_i : den totala kraften som verkar på länk i i dess masscentrum

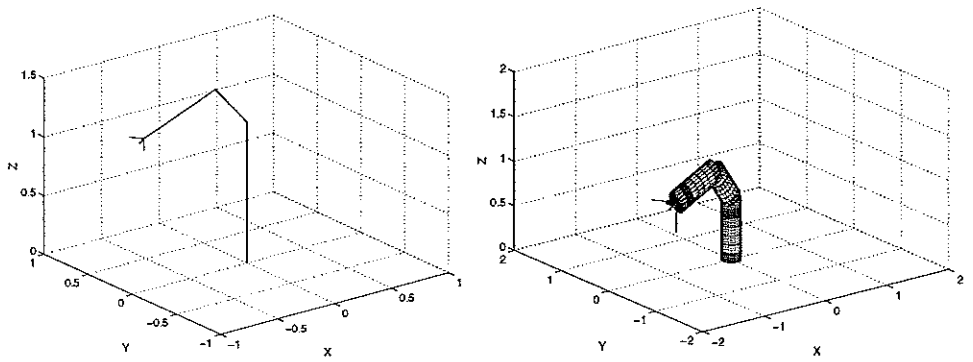
$^{i-1}_iR$: Ortonomala rotationsmatrisen som beskriver koordinatsystem is orientering i förhållande koordinatsystem *i-1*

$^ip^*_i$: translationsvektorn som beskriver koordinatsystem is position i förhållande koordinatsystem *i-1*

z_0 : enhetsvektorn för z_0-riktningen

4.5 Animate

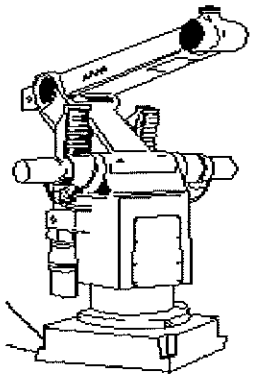
Två olika animationer kan utföras. Den första animationen (vänster), som följde med robotikpaketet ritar upp en bild av roboten genom att dra linjesegment mellan de olika koordinatsystemen fästa vid de olika länkarna. Den andra (höger) har implementerats m.h.a. Matlabs inbyggda funktioner för 3D-modellering (Se fig 4.4). Den är sålunda grafiskt sätt mer avancerad, men också långsammare. Tanken med 3D-modellen är att ledernas grafiska modell (i detta fall cylindrar) skall bytas ut mot mera korrekta grafiska modeller, helst direkt från ett CAD-system.



Figur 4.4 Två olika sätt att animera robotens rörelse.

5. IRB-6

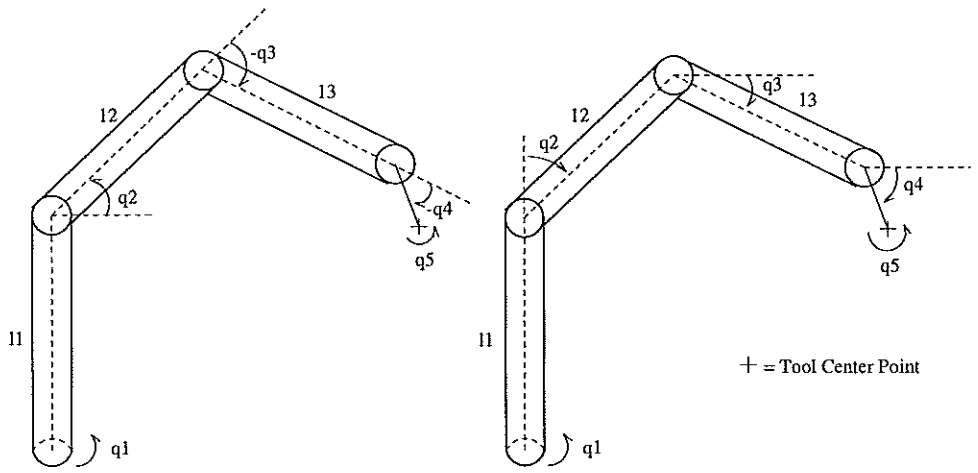
IRB-6 är en industrirobot från ABB. Den har fem leder (fem frihetsgrader), tre armlänkar, en handled och ett gripdon. I Robotens rörelsemönster ingår sålunda koordinering och reglering av fem leder samt öppning och stängning av gripdonet.



Figur 5.1 IRB-6

5.1 Direkt kinematik

ABB har en egen vinkeldefinition som skiljer sig från Denavit-Hartenbergdefinitionen :



Figur 5.2 Denavit-Hartenbergs definition (vänster) och ABBs vinkeldefinition (höger).

$$
\begin{aligned}
q_1^{DH} &= q_1^{ABB} \\
q_2^{DH} &= 90 - q_2^{ABB} \\
q_3^{DH} &= 270 - (q_3^{ABB} - q_2^{ABB}) \\
q_4^{DH} &= q_4^{ABB} - q_3^{ABB} \\
q_5^{DH} &= q_5^{ABB}
\end{aligned}
$$

Med hjälp av Denavit-Hartenberg formalism kan en tabell ställas upp som ger en sammanfattande karaktäristik av IRB-6 kinematik. Genom att utnyttja de olika länkparametrarna i tabellerna, antingen standardparametrar (vänster) eller modifierade parametrar (höger) kan sedan transformationsmatriserna för de olika lederna bestämmas :

i	α_i	a_i	d_i	Θ_i
1	90°	0	l_1	Θ_1
2	0	l_2	0	Θ_2
3	0	l_3	0	Θ_3
4	−90°	0	0	Θ_4
5	0	0	0	Θ_5

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	0	0	0	Θ_1
2	90°	0	l_1	Θ_2
3	0	l_2	0	Θ_3
4	0	l_3	0	Θ_4
5	−90°	0	0	Θ_5

5.2 Invers kinematik

En sluten lösning av den inversa kinematiken för IRB-6 implementerades. Genom att utnyttja den direkta kinematiken i 5.1.1 kan inversa kinematiken beräknas genom en kombination av algebraisk manipulation och enkel geometrisk analys [1] :

$$
{}^{N}_{0}T = \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

L_1, L_2, L_3 : längd på led 1,2,3

$q_1 = \arctan 2(p_x, p_y)$

$D = \frac{p_x^2+p_y^2+p_z^2-L_2^2-L_3^2}{2L_2L_3}$

$q_3 = -\arctan 2(\sqrt{1-D^2}, \cos q_3)$

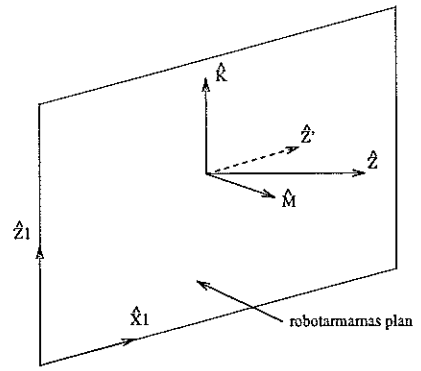
$q_2 = \arctan 2(p_z - L_1, \sqrt{p_x^2+p_y^2}) + \arctan 2(L_3 \sin q_3, L_2 + L_3 \cos q_3)$

$q_{234} = \arctan 2(r_{13}\cos q_1 + r_{23}\sin q_1, r_{33})$

$q_4 = q_2 + q_3 - q_{234}$

$q_5 = \arctan 2(r_{21}\cos q_1 - r_{11}\sin q_1, r_{22}\cos q_1 - r_{12}\sin q_1)$

Då IRB-6 inte har sex frihetsgrader och därmed inte har full rörelsefrihet i rymden måste man ta hänsyn till detta; verktygets rikning måste ligga i robotarmarnas plan. Om verktyget inte ligger i detta plan och handledens koordinatsystem endast skiljer sig från verktygets med en translation kan man rotera in redskapet in i planet :



Figur 5.3 Rotering av verktyget in i robotarmarnas plan.

p_x och p_y är x och y koordinaterna för önskad position för verktyget

$$\hat{X} = \begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \end{pmatrix} \quad \hat{Y} = \begin{pmatrix} r_{12} \\ r_{22} \\ r_{32} \end{pmatrix} \quad \hat{Z} = \begin{pmatrix} r_{13} \\ r_{23} \\ r_{33} \end{pmatrix} \quad \hat{M} = \frac{1}{\sqrt{p_x^2+p_y^2}} \begin{pmatrix} -p_y \\ p_x \\ 0 \end{pmatrix}$$

$$\hat{Z}' = \hat{M} \times \hat{Z}$$

$$\hat{Y}' = \cos\Theta\hat{Y} + \sin\Theta(\hat{K} \times \hat{Y}) + (1 - \cos\Theta)(\hat{K} \cdot \hat{Y})\hat{K}$$

$$\hat{X}' = \hat{Y}' \times \hat{Z}'$$

5.3 Motorvinklar

Förhållandet mellan Denavit-Hartenbergs vinkeldefinition och robotens motorvinklar :

$$q_1 = n_1 s_1$$

$$q_2 = 270 - a - b - \arccos\left(\frac{D^2+E^2-x_2^2}{2DE}\right) = [x_2 = \frac{n_2 s_2}{360} + x_{02}]$$

$$= 270 - a - b - \arccos\left(\frac{D^2+E^2-(\frac{n_2 s_2}{360}+x_{02})^2}{2DE}\right)$$

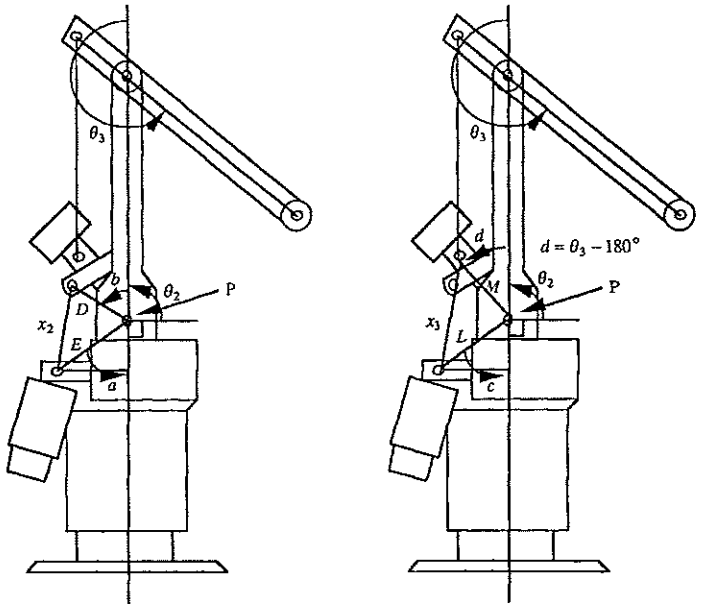
$$q_3 = 450 - c - \arccos\left(\frac{L^2+M^2-x_3^2}{2LM}\right) - q_2 = [x_3 = \frac{n_3 s_3}{360} + x_{03}]$$

$$= 450 - c - \arccos\left(\frac{L^2+M^2-(\frac{n_3 s_3}{360}+x_{03})^2}{2LM}\right) - q_2$$

$$q_4 = n_4 s_4$$

$$q_5 = n_5\left(s_5 - \frac{s_4}{n_4}\right)$$

n_i= ledhastighet/länkhastighet för varje led

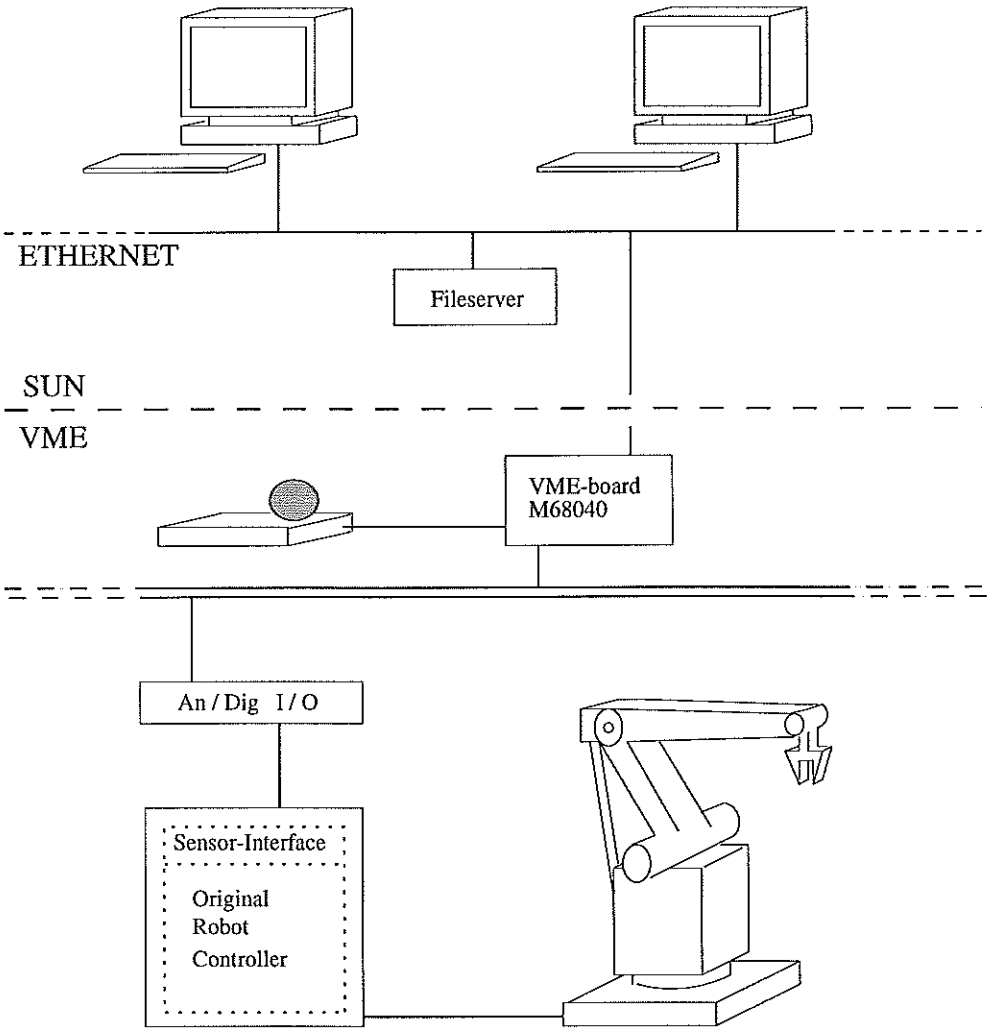


Figur 5.4 Beskrivning av förhållandet mellan ledvinklar(definierade enligt Denavit-Hartenbergs metod) och motorvinklar för led 2 (vänster) och led 3 (höger). Led 2 betecknas med P.

Det bör nämnas att kopplingen mellan led 2 och 3 försvinner om man använder ABBs vinkeldefinition.

6. Test med IRB-6

Trajektoriegeneratorn testades på en robot av typ IRB-6 från ABB. Denna robot ingår i ett robotsystem som finns på Institutionen för Reglerteknik. Med robotsystemet kan man styra roboten manuellt med hjälp av en styrspak, genom direkt inläsning av en tabell innehållande de önskade koordinaterna eller via ett program skrivet i ett enkelt robotspråk. Systemets uppbyggnad framgår av figur 6.1.



Figur 6.1 Systemöversikt över hårdvara.

Mjukvara

Mjukvarudelen är implementerad i Modula-2. Huvuddelen av mjukvaran utgörs av olika processer med högt ställda realtidskrav som tar emot och tolkar rörelsebeskrivningar och kommandon, och omsätter dessa till parametrar och börvärden till den underliggande regleringen. I det följande ges en orientering av de delar av mjukvaran som är relevant och vilka ändringar och modifikationer som har gjorts för att trajektoriegeneratorn skulle kunna testas. Se figur 6.5.

6.1 Interpreter

Denna modul innhåller en implementering av ett Basicliknande robotspråk. Ett antal instruktioner som tillåter enkla aritmetiska operationer och rörelsefunktioner hade tidigare definierats. Till de redan existerande instruktionerna har följande instruktioner lagts till :

IPRETRAJ : När detta kommando exekveras startas PreCompute- processen. För närvarande måste trajektorigeneratorn initieras i Matlab. Det vore lämpligt om denna instruktion även såg till att generatorn initierades.

JPOS x y z t4 t5 t : Om detta kommando exekveras beräknas en linjär rörelse i ledvinkelkoordinater och den beräknade trajektorian skickas till Regul.

JVIA x y z t4 t5 t : Anger att den specificerade viapunkten ska passeras. Används tillsammans med JPOS.

CPOS x y z t4 t5 t : Om detta kommando exekveras beräknas en linjär rörelse i kartesiska koordinater och den beräknade trajektorian skickas till Regul.

CVIA x y z t4 t5 t : Anger att den specificerade viapunkten ska passeras. Används tillsammans med CPOS.

QPRETRAJ : Denna instruktion avslutar trajektoriegeneratorn i Matlab, stänger ner nätverksförbindelsen och terminerar PreCompteTrajprocessen.

Rörelseinstruktionernas argument ges av :

x,y och **z** : kartesiska koordinater i mm
t4 och **t5** : ledvinklar för led 4 och led 5 i grader
t : tid i sekunder

Exempel :

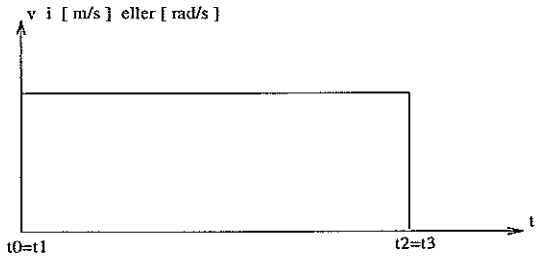
```
10 IPRETRAJ
20 JVIA 928.58 0.00 1122.93 5.97 -78.21 2.0
30 JPOS 794.20 4.76 718.94 5.97 -78.21 3.0
40 CVIA 879.46 -165.49 1381.27 5.11 -77.89 1.0
50 CPOS 594.14 554.94 1160.23 5.20 -77.73 4.0
60 QPRETRAJ
70 STOP
```

6.2 PreComputeTraj

För att systemets realtidskrav skulle uppfyllas krävdes det att en process implementerades som tar emot rörelsespecifikationerna från Interpreter för vidare befordran till trajektoriegeneratorn och som sedan tar tar hand om de beräknade trajektorierna. Detta har gjorts. Då rörelsen specificeras med hjälp av Denavit-Hartenbergs vinkeldefinition i generatorn och robotsystemet arbetar med ABBs vinkeldefiniton sker en konvertering av rörelsespecfikationerna och de förberäknade trajektorierna mellan de olika vinkeldefinitionerna. För att minska den oundvikliga tidsfördröjningen som uppkommer då en trajektoria och vridmoment förberäknas läggs trajektorierna i en lista : Om en begärd trajektoria redan finns beräknad, så hämtas den direkt ur listan. Innan den förberäknade trajektorian läggs in i listan kollas att rörelsen befinner sig inom robotens arbetsområde. När PreComputeprocessen har lagt eller hämtat en trajektoria i listan skickas trajektorian via Interpreter till Trajecprocessen.

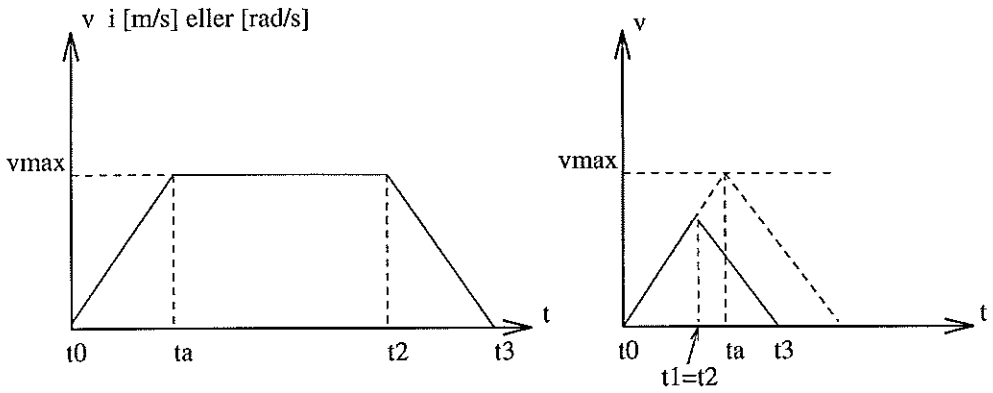
6.3 Trajec

Modulen Trajec är systemets referensgenerator som genererar börvärden i ledkoordinater, alternativt i kartesiska koordinater. Ursprungligen genererades trajektorier baserade på två olika hastighetsprofiler för varje led, såväl i kartesiska som i ledkoordinater, i denna modul. Den första av hastighetsprofilerna kräver egentligen oändlig acceleration i start- och ändpunkter (Se figur 6.2). Denna typ av trajektoriegenerering används vid manuell körning. Genom att man i detta fall inte använda framkoppling av hastigheten, får man ett större reglerfel och därmed en mjuk rörelse med dålig banföljning. Men då manuell körning utförs med låg hastighet får den bristande banföljningen ingen större betydelse.



Figur 6.2 Sträckan eller vinkeln som ska avverkas motsvaras av ytan under kurvan.

Den andra typen av hastighetsprofil innebär att accelerationen blir begränsad (Se figur 6.3). Den här typen av trajektoria är sålunda mer sofistikerad. Fortfarande är dock accelerationen diskontinuerlig och hänsyn tas inte till varierande dynamiska begränsningar.

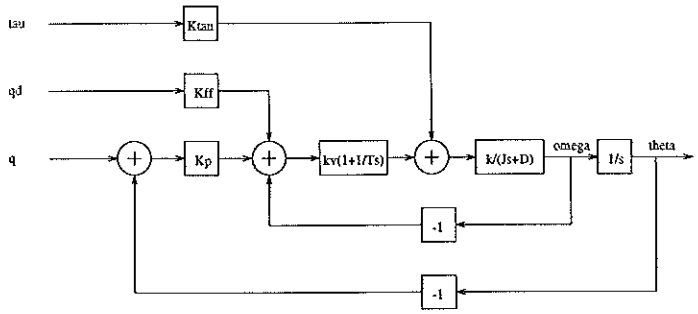


Figur 6.3 Sträckan eller vinkeln som ska avverkas motsvaras av ytan under kurvan. Här fås två olika profiler beroende på om $v_{max} * t_a \leq$ ytan eller inte.

För att en förberäknad och mera optimerad trajektoria ska kunna användas har Trajecprocessen modifierats. När en förberäknad trajektoria har beräknats, så tar trajecprocessen hand om denna och ser till att de beräknade börvärdena skickas ner till regulatorn med korrekt samplingstid. Om förbindelsen mellan trajektoriegeneratorn och målsystemet bryts under exekveringen, så beräknas en trajektoria i målsystemet enligt den ursprungliga metoden.

6.4 Regul

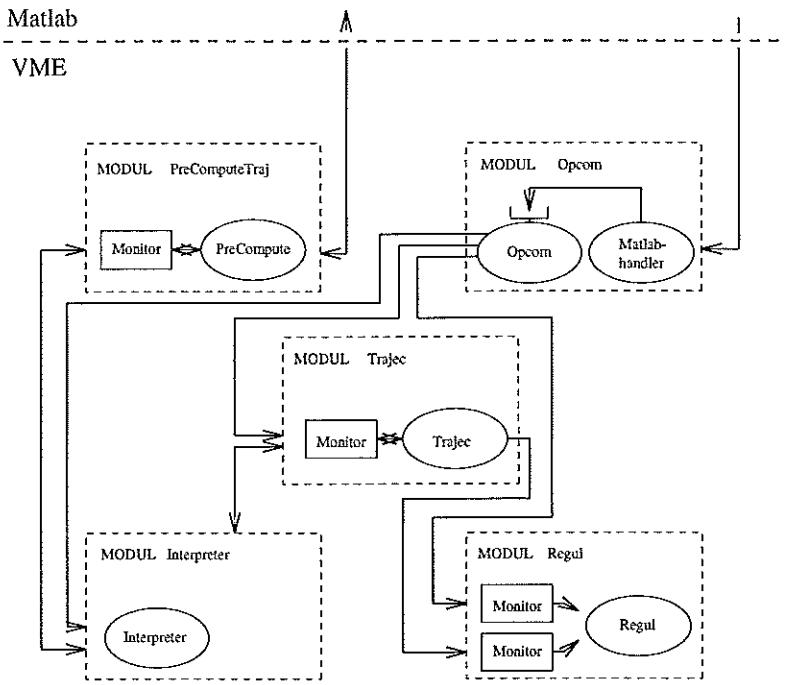
IRB-6 har fem robotservon. Varje servo regleras oberoende av varandra. Detta sätt att lösa reglerproblemet tar dock inte hänsyn till att en ledvinkels rörelse påverkar de övriga ledvinklarnas rörelse. Ett sett att kompensera för de reglerfel som uppstår på grund av olinjära kopplingar och samtidigt motverka reglerfel på grund av Coriolis, centripetal och gravitationskrafter är att framkoppla ett förberäknat vridmoment. En sådan framkoppling har implementerats. Som grund har använts en redan existerande regulator med hastighetsframkoppling $(e * Kp + v * Kff)$. Modulen Regul innehåller den del av regleringen som implementerats i programvaran. Se figur 6.4.



Figur 6.4 Förenklat blockschema för regleringen av en led m.h.a. framkoppling av vridmoment.

6.5 Opcom

I modulen Opcom finns två processer, Matlabhandler och Compro, som tar hand om kommandon som kommer från operatören. Matlabhandler konverterar Matlab-meddelanden, som kommer över Ethernet från värddatorn, till avkodade meddelanden för realtidsprogrammet. En del av dessa skickas vidare till Compro för vidare bearbetning där. Processen tar emot meddelanden i en brevlåda. Dessa tolkas och beroende på vilken typ de är av utförs olika operationer. Här beräknas t.ex. trajektorierna genom anrop av rutiner i trajektoriegeneratorn, eftersom detta avlastar den högre prioriterade processen Trajec. I denna modul har inga modifieringar gjorts.



Figur 6.5 Systemöversikt över relevant realtidsmjukvara. Modulerna Trajec, Interpreter och Regul har utvidgats. PreComputeTraj har lagts till.

7. Sammanfattning och kommentarer

För att minimera arbetsbördan vid utveckling av nya algoritmer och modeller använder man sig ofta av interaktiva beräknings- och simuleringsprogram som Matlab. I detta arbete har Matlab använts för att utveckla en robotoberoende trajektoriegenerator. Som grund har använts ett robotikpaket, som efter utvärdering av två olika paket, valts ut. Valet föll på Robotics Toolbox och detta ställningstagande var baserat på traditions- och flexibilitetsskäl. Trajektoriegeneratorn arbetar med matrisstrukturer och all kommunikation sker i matrisform. En modulariserad struktur på generatorn har eftersträvats för att det ska vara enkelt att göra tillägg och modifieringar. Följande funktioner som används i trajektoriegeneratorn har implementerats i Matlab :

- linjär rörelse med viapunkter i ledkoordinter
- linjär rörelse med viapunkter i kartesiska koordinater
- cirkulär rörelse i kartesiska koordinater
- linjärinterpolation
- sluten lösning av invers kinematik för IRB-6
- animering av robotrörelser m.h.a. 3D-funktioner

En programmodul har utvecklats och lagts till den existerande mjukvaran hos IRB-6 för att möjliggöra snabba och enkla tester. Denna modul och modifieringar i övrig mjuvara har gjort det möjligt att använda trajektoriegeneratorn som en "compute server" från IRB-6s styrsystem. Då det inte framgår av rapporten kan det vara värt att påpeka att hälften av den arbetsinsats som gjorts har gått åt till att få trajektoriegeneratorn och IRB-6 att fungera tillsammans. På grund av att tröghetsmatrisen för IRB-6 saknades, testades endast trajektoriegenereringen fullt ut med den regulator som fanns tidigare, vilken endast använde sig av framkoppling av hastigheten. Den nya regulatorn med vridmomentsframkoppling har grovinställts. De tröghetsdata som använts har tagits fram med hjälp av ett identifieringsverktyg som utvecklats på institutionen. Vid fortsatt utveckling kan förslagsvis följande saker behandlas :

- singularitetstest
- effektivare generell inversberäkning
- vridmomentsoptimering
- symbolisk dynamikberäkning
- banplanering

Användning av interaktiva beräkningsprogram som värddatorprogram måste anses vara ett kraftfullt laborations- och utvecklingsverktyg. Det system som utvecklats borde vara användbart inom såväl forskning som undervisning.

8. Referenser

[1] CRAIG, JOHN J. (1986) : Introduction to Robotics : Mechanics and Control, Addison-Wesley.

[2] SPONG, MARK W. and VIDYASAGAR,M. (1989) : Robot Dynamics and Control, John Wiley & Sons.

[3] PAUL, RICHARD P. (1981) : Robot Manipulators, The Massachusetts Institute of Technology.

[4] NETHERTY, JOHN (1993) : Robotica, User's Guide and Reference Manual

[5] CORKE, PETER I. (1994) : Robotics Toolbox, for Use with MATLAB

[6] THE MATHWORKS INC (1992) : Matlab, User's Guide

[7] WOLFRAM STEPHEN (1991) : Mathematica, A System for Doing Mathematics by Computer, Addison-Wesley.

[8] WATERLOO MAPLE PUBL. (1991) : Maple Languages Reference Manual

[9] WHEELER, GRAHAM THOMAS CHARLES wheegr@hal.wwc.edu (1994) : Solution to the Dynamic Equations of Motion fo Any Robot Given the Denavit-Hartenberg Joint Variables and Link Parameters

[10] KING, K.N (1988) : Modula-2 A Complete Guide, D.C. Heath and Company

[11] ELMQVIST, H. (1975) : An Interactice Simulation Program for Nonlinear Systems, User's Manual, Department of Automatic Control, Lund Institute of Technology.

For an n-axis manipulator the following matrix naming and dimensional conventions apply.

Symbol	Dimensions	Description
dh	$n \times 5$	manipulator kinematic description matrix
dyn	$n \times 20$	manipulator kinematic and dynamic description matrix
q	$1 \times n$	joint coordinate vector
q	$m \times n$	m-point joint coordinate trajectory
qd	$1 \times n$	joint velocity vector
qd	$m \times n$	m-point joint velocity trajectory
qdd	$1 \times n$	joint acceleration vector
qdd	$m \times n$	m-point joint acceleration trajectory
T	4×4	homogeneous transform
T	$m \times 16$	m-point homogeneous transform trajectory
Q	1×4	unit-quaternion
v	3×1	Cartesian vector
t	$m \times 1$	time vector
d	6×1	differential motion vector

A trajectory is represented by a matrix in which each row corresponds to one of m time steps. For a joint coordinate, velocity or acceleration trajectory the columns correspond to the robot axes. Things are a little more complicated for homogeneous transform trajectories since MATLAB does not (yet) support 3-dimensional matrices. The approach used in this Toolbox is that each row is a homogeneous transform that has been 'flattened' using the `(:)` operator. Each row can be restored to a 4×4 matrix by using the `reshape` function.

Homogeneous Transforms	
eul2tr	Euler angle to homogeneous transform
oa2tr	orientation and approach vector to homogeneous transform
rotx	homogeneous transform for rotation about X-axis
roty	homogeneous transform for rotation about Y-axis
rotz	homogeneous transform for rotation about Z-axis
rpy2tr	Roll/pitch/yaw angles to homogeneous transform
tr2eul	homogeneous transform to Euler angles
tr2rpy	homogeneous transform to roll/pitch/yaw angles

Quaternions	
q2tr	quaternion to homogeneous transform
qinv	inverse of quaternion
qnorm	normalize a quaternion
qqmul	multiply (compound) quaternions
qvmul	multiply vector by quaternion
qinterp	interpolate quaternions
tr2q	homogeneous transformto unit-quaternion

Kinematics	
dh	Denavit-Hartenberg conventions
diff2tr	differential motion vector to transform
fkine	compute forward kinematics
ikine	compute inverse kinematics
jacob0	compute Jacobian in base coordinate frame
jacobn	compute Jacobian in end-effector coordinate frame
linktrans	compute a link transform homogeneous transform
mdh	modified Denavit-Hartenberg conventions
mfkine	compute forward kinematics (modified Denavit-Hartenberg)
mlinktrans	compute a link transform homogeneous transform(modified Denavit-Hartenberg)
tr2diff	homogeneous transform to differential motion vector
tr2jac	homogeneous transform to Jacobian

Dynamics	
cinertia	compute normalized manipulator inertia matrix
coriolis	compute centripetal/coriolis torque
dyn	dynamics conventions
friction	joint friction
gravload	compute gravity loading
inertia	compute manipulator inertia matrix
itorque	compute inertia torque
rne	inverse dynamics

Manipulator Models	
puma560	Puma 560 data
stanford	Stanford arm data

Trajectory Generation	
ctraj	Cartesian trajectory
drivepar	Cartesian trajectory parameters
jtraj	joint space trajectory
trinterp	interpolate homogeneous transforms

Graphics	
plotbot	animate robot

Other	
cross	vector cross product
manipblty	compute manipulability
rtdemo	toolbox demonstration
unit	unitize a vector