ISSN 0280–5316
ISRN LUTFD2/TFRT--5544--SE

Simulering av släphydrofon

Stefan Nilsson

Institutionen för Reglerteknik
Lunds Tekniska Högskola
November 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
November 1995

Document Number
ISRN LUTFD2/TFRT--5544--SE

Author(s)
Stefan Nilsson

Supervisor
Sven Erik Mattsson, Ola Dahl

Sponsoring organisation
Kockums Submarine Systems AB, Malmö

Title and subtitle
Simulering av släphydrofon (Simulation of towed array sonar)

Abstract

A Towed Array Sonar (TAS) is a cable towed behind a ship or a submarine. Sonars are attached to the cable for sound detection under water. The advantage is, compared to regular sonars attached to the boat, that low frequency sounds are easier to detect. Also, the sound from the submarine is less disturbing. In this report a model representing TAS-movement under water is developed and simulated. The mathematical model is described in detail. The result is a number of partial differential equations. These equations are implemented in the simulation environment Omsim/Omola and in the mathematical program Matlab. The model is then validated, i.e., compared to a real world experimental maneuver. Finally, a number of simulation results for different kinds of maneuvers are presented. The results are given as plots of the cable's time-dependent shape.

Key words
Modelling; simulation; Partial differential equation; Towed array sonar; Omola; Matlab; Ocean engineering

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
Swedish

Number of pages
88

Recipient's notes

Security classification

Innehållsförteckning

1. Sammanfattning

1.1 Sammanfattning på svenska

En släphydrofon är en kabel med sonarer, d v s mikrofoner med uppgift att fånga upp ljud i vattnet. Den bogseras bakom ett ytfartyg eller en ubåt. Fördelen med släphydrofon, jämfört med att ha hydrofoner monterade på ubåten, är att ubåtens egenljud påverkar mindre och att man kan uppfatta lågfrekventa ljud noggrannare.

I rapporten beskrivs en modell för en släphydrofons rörelse i vattnet och simuleringsresultat.

Först beskrivs den matematiska modellen i detalj. Modellen består av 6 st olinjära partiella differentialekvationer. Sedan beskrivs hur dessa ekvationer har diskretiserats i rummet och hur de resulterande ordinära differentialekvationerna har implementerats, dels i simuleringsverktyget Omsim/Omola, och dels i matematikverktyget Matlab.

Därefter beskrivs hur modellen har validerats, d v s jämförts med ett verkligt experiment. Detta ger en uppfattning om hur väl modellen beskriver verkligheten.

Avslutningsvis presenteras ett antal simuleringsresultat. I diagram visas kabelns tidsberoende form vid olika typer av manövrar.

1.2 Sammanfattning på engelska (Summary)

A Towed Array Sonar (TAS) is a cable with sonars attached to it for sound detection under water. The cable is towed behind a ship or a submarine. The advantages of TAS, compared to regular sonars attached to the boat, is that the sound from the submarine is less disturbing and that low frequency sounds are easier to detect.

In this report a model representing TAS-movement under water is developed and simulated.

Initially, the mathematical model is described in detail. The result is 6 partial differential equations. These equations are then discretisised in space and the resulting ordinary differential equations are implemented in the simulation environment Omsim/Omola and in the mathematical program Matlab.

The model is then validated, i e compared to a real world experimental manœuver. This gives some intention on how well the model works.

Finally, a number of simulations are presented. In diagrams the cable's time-dependent shape is plotted at different kinds of manœuvers.

2. Bakgrund

2.1 Sonarsystem

Olika typer av sonarsystem eller hydrofoner används för att upptäcka främmande ubåtar och för att skilja dem från djur och andra föremål i vattnet. Systemens uppgift är att bestämma avstånd och bäring till ubåtarna.

Sonarutrustningen placeras som regel i fören på ubåten eller utmed sidorna. Digital signalbehandling används för att dämpa ut vibrationer och egenljud från båten. Både passiva och aktiva hydrofoner används. Nackdelen med aktiva hydrofoner är att de avslöjar ubåtens läge med sitt ping.

Eftersom avståndet mellan sensorerna begränsas av ubåtens längd blir även noggrannheten begränsad vad gäller förmågan att upptäcka lågfrekventa ljud i vattnet. Avståndet mellan sensorerna motsvarar den våglängd som kan uppfattas och ökad våglängd motsvarar lägre frekvens.

2.2 Släphydrofon

Genom att bogsera en kabel utrustad med sonarsensorer efter ubåten så kan man detektera lågfrekventa ljud i vattnet bättre. Dessutom blir inverkan av ubåtens egenljud betydligt mindre eftersom avståndet till sonarsensorerna är större.

Släphydrofon används även på ytfartyg. Då har kabeln delats upp i en tyngre *Towcable*-del (endast kabel) och en *Array*-del med sonarutrustning som gjorts viktlös i vattnet. *Towcable*-delen har gjorts tung för att kabeln ska sjunka till ett visst djup från ytan och *Array*-delen har gjorts viktlös för att det ur signalbehandlingssynpunkt är önskvärt med en så rak form som möjligt på sonararrayen.

Vid användning av bogserad kabel efter ubåt behöver inte *Towcable*-delen göras tung eftersom man redan befinner sig på ett visst djup under vattenytan. Både *Towcable*-delen och *Array*-delen kan göras viktlösa i vattnet.

2.3 Collins

I Australien bygger Kockums en ny typ av ubåtar i Collinsklassen. Dessa baseras på den svenska ubåtsklassen Västergötland men har anpassats till australiensiska förhållanden. Eftersom de har mycket större områden att patrullera blir kraven högre när det gäller att förbli oupptäckt och att upptäcka andra ubåtar. Stora operationsområden innebär att en lång släphydrofon kan användas för att upptäcka andra ubåtar bättre

Collins släphydrofon är utvecklad av GEC-Marconi i Sydney där man även utvecklar traditionella båtmonterade hydrofoner. Kabeln är 1,5 km lång och har en diameter på 40 mm. På kabeln sitter sensorer av olika typer med någon meters mellanrum. Data från dessa hundratals sensorer samlas in, skickas digitalt och bearbetas ombord i ubåten för att bestämma avstånd och riktning till andra fartyg.

Kabeln går att vinscha in och ut utan assistans från andra fartyg vilket är en taktiskt viktig fördel. Tidigare har man varit tvungen att vinscha upp kabeln på en trumma som endast fått plats på ett ytfartyg. Detta beroende på kabelns styvhet och sonarernas storlek. Med betydligt mindre elektronik har man lyckats hålla nere kabelns diameter så att den uppvinschade kabeln får plats på ubåten.

Collins är även utrustad med båtmonterade sonarsystem. I fören sitter den stora cylindriska attackhydrofonen med vilken man bestämmer bäringen till målet och på sidorna sitter långbashydrofoner för lågfrekvent avståndsbestämning. Långbashydrofonerna används när man inte har den noggrannare släphydrofonen utvinschad.

3. Inledning

3.1 Uppgiften

Uppgiften var att utveckla och validera en simuleringsmodell för en *Towed Array Sonar*, en släphydrofon som utgörs av en kabel som bogseras bakom en ubåt.

Modellen skulle beskriva kabelns rörelse i tiden relativt ett jordfast koordinatsystem. Indata till simuleringsmodellen är ubåtens hastighet samt de fysikaliska data som beskriver kabeln. Dessutom skulle modellen valideras på något sett, d v s det skulle utredas hur bra den beskrev verkligheten.

3.2 Tidigare arbeten

Problemet har studerats av bl a Ablow och Schechter i en artikel i *Ocean engineering* 1983 [1] och av David Unger i ett examensarbete 1991 [2].

Ungers arbete behandlar främst kabelns tidsoberoende form då ubåten kör med konstant kurs och fart. Ett datorprogram för detta fall används och resultatet ges i form av ett antal diagram som visar kabelns form vid olika typer av vikt, diameter, hastighet och belastning i kabeländen. Andra delen i Ungers rapport behandlar teorin till det dynamiska fallet, d v s kabelns tidsberoende form då ubåtens hastighet och kurs är godtyckliga.

Som en uppföljning till Ungers arbete skrevs ett datorprogram i Fortran. Programmet beskriver den dynamiska lösningen. Detta program är dock stort och svåröverskådligt och använder inga färdiga matematikfunktioner, typ matris-multiplicering, utan allt är programmerat från grunden.

3.3 Mål och ansats

Ett mål med detta examensarbete var att utveckla en datorimplementerad modell till det dynamiska fallet, som var mer överskådlig än det givna Fortran-programmet. Genom att utgå från den matematiska beskrivningen av Unger, som i sin tur var hämtad från Ablow och Schechters artikel, så skulle en modell som beskrev det dynamiska fallet arbetas fram.

Därefter skulle den implementerade modellen valideras. Det skulle visas att de genererade simuleringarna överensstämde med ett verkligt experiment. Slutligen skulle ett antal simuleringsfall studeras.

3.4 Detta arbete

Under detta arbete har två olika simuleringsmiljöer använts. Omsim och Matlab.

Modellen som beskriver det dynamiska fallet har till en början implementerats i simuleringsverktyget Omsim, som är utvecklat på institutionen för Reglerteknik vid Lunds Tekniska Högskola. Omsim har tidigare använts bl a för att simulera kemiska processer.

Den valda modellen har även implementerats i matematikverktyget Matlab, som bl a används på Kockums Submarine Systems i Malmö.

Modellen har validerats. Simuleringarna har visat sig överensstämma med ett verkligt experiment, utfört i bassänger vid David Taylor Naval Ship Research and Development Center 1980.

4. Matematisk modell

Innehållet i detta kapitel följer i stora drag framställningen i Ablow och Schechters artikel i Ocean Engineering 1983 [1]. Modelleringen görs i detalj eftersom det visade sig att deras artikel innehöll fel.

4.1 Grundläggande antaganden

Kabeln behandlas som en lång, tunn böjlig cylinder i godtycklig rörelse under inverkan av gravitation, ubåtens manövrering, tröghetskrafter, töjning och hydrodynamiska krafter.

Rörelsen i roll-led, d v s kabelns vridning runt sin egen längdaxel, försummas och endast kabelns kurs- och trimvinkel tillåts variera. Kabelns material antas vara linjärt elastiskt.

4.2 Koordinatsystem

För att beskriva kabelns rörelse i vattnet införs två olika koordinatsystem, ett jordfast S_J (**i**,**j**,**k**) och ett lokalt koordinatsystem för kabeln S_K (**t**,**n**,**b**), med **t** i tangentens riktning. Koordinatsystemen är illustrerade i figur 1.

För att transformera punkter i rummet mellan de två koordinatsystemen så används en ortogonal transformationsmatris $\mathbf{L_{JK}}$. Samband:

$$(\mathbf{t}\quad \mathbf{n}\quad \mathbf{b}) = (\mathbf{i}\quad \mathbf{j}\quad \mathbf{k})\mathbf{L}_{\mathbf{JK}} \tag{4.2.1}$$

eller ekvivalent

$$(\mathbf{t}\quad \mathbf{n}\quad \mathbf{b})^T = \mathbf{L}_{\mathbf{JK}}^T(\mathbf{i}\quad \mathbf{j}\quad \mathbf{k})^T \tag{4.2.2}$$

Alla de kraftekvationer för kabeln som härleds kan nu uttryckas i båda koordinatsystemen. Uppställning och lösning av dessa ekvationer kommer framöver att ske endast i kabelns koordinatsystem S_K. Nu följer en härledning av matrisen $\mathbf{L_{JK}}$ genom att först beräkna $\mathbf{L_{JK}^T}$.

Matrisen $\mathbf{L_{JK}^T}$ är sammansatt av tre rotationsmatriser och erhålls på följande sätt:

1. Utgå från det jordfasta koordinatsystemet S_J (**i**,**j**,**k**). Rotera vinkeln (-θ) kring **k**-axeln. Detta innebär multiplikation från vänster med matrisen

$$C_{k,-\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Rotera därefter vinkeln (-ψ) kring den *nya* **i**-axeln, d v s multiplicera från vänster med

$$C_{i,-\psi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix}$$

3. Rotera slutligen vinkeln φ kring den nya **k**-axeln, vilket betyder multiplikation från vänster med

$$C_{k,\phi} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

De tre vinklarna θ, ψ och φ är s k Eulervinklar som beskriver hur koordinatsystemen roteras i förhållande till varandra.

Rörelsen i roll-led försummas nu, d v s det antas att kabeln ej kan vrida sig kring sin egen längdaxel. Detta innebär att vinkeln ψ blir konstant. Om man sätter ψ = π/2 så erhålls transformationsmatrisen

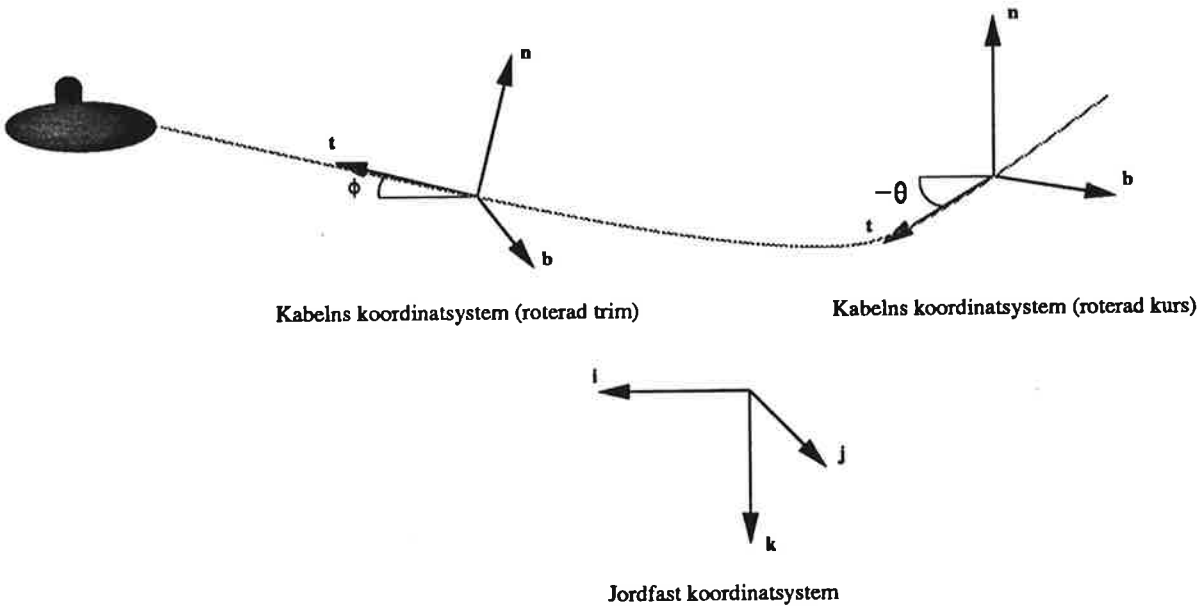
$$\mathbf{L}_{\mathbf{JK}}^{T} = C_{k,\phi} C_{i,-\pi/2} C_{k,-\theta} \tag{4.2.3}$$

som efter transponering blir[1]

$$\mathbf{L}_{\mathbf{JK}} = \begin{pmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ -\sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \\ -\sin\phi & -\cos\phi & 0 \end{pmatrix} \tag{4.2.4}$$

Vinkeln θ beskriver ubåtens kurs gentemot kabeln och med valet av $\psi = \pi/2$ så kommer ϕ att beskriva trimvinkeln, som beskriver kabelns vinkel gentemot det jordfasta horisontalplanet.

Detta illusteras i följande figur.



Figur 1. Illustration av jordfast och lokalt koordinatsystem.

[1] Transformationsmatrisen ABC i Ablow och Schechters artikel [1] innehåller teckenfel på två matriselement.

4.3 Kraftekvationer

Genom att utgå från krafterna som verkar på kabeln fås följande kraftekvation:

$$\frac{\partial}{\partial S}\mathbf{T} + \mathbf{W} + \mathbf{F} + \mathbf{B} = 0 \qquad (4.3.1)$$

där **T** är den inre kraften i kabeln , **W** är vikten i vattnet per längdenhet, **F** de yttre hydrodynamiska krafterna (p g a vattenmotståndet) per längdenhet och **B** tröghetskrafterna per längdenhet. Varibeln S beskriver sträckan längs den *töjda* kabeln.

Det jordfasta koordinatsystemet S_J är definierat så att **k** pekar rakt nedåt. Detta innebär att vikten i vattnet per längdenhet blir

$$\mathbf{W} = w\mathbf{k} / \frac{\partial S}{\partial s} \qquad (4.3.2)$$

med

$$w = (m - \rho A)g \qquad (4.3.3)$$

där m är massan per längdenhet, A tvärsnittsarean och ρ vattnets densitet. Lilla s beskriver sträckan längs den *otöjda* kabeln.

Genom att införa en ortsvektor **r** i det jordfasta systemet S_J så kan en punkt på kabeln skrivas

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \qquad (4.3.4)$$

En punkt på kabeln beskrivs alltså med de tre koordinaterna (x,y,z).

Ortsvektorn **r** används senare vid härledningen av de skalära ekvationerna.

Tröghetskraften **B** beror på massan och hastigheten. Om man tar hänsyn till både kabelns och det undanträngda vattnets massa och hastighet så blir denna tröghetskraft per längdenhet

$$\mathbf{B} = -\frac{\partial}{\partial t}\left[\left\{m_1 \frac{\partial \mathbf{r}}{\partial t} - \rho A[\mathbf{J} + (\mathbf{U}\cdot\mathbf{t})\mathbf{t}]\right\} / \frac{\partial S}{\partial s}\right] \qquad (4.3.5)$$

$$m_1 = m + \rho A \qquad (4.3.6)$$

där **t** är basvektorn i tangentiell led i kabelns koordinatsystem S_K. Vattnets strömning beskrivs av **J** och anges relativt det jordfasta koordinatsystemet. Kabelns hastighet relativt strömmen betecknas **U**. Alltså

$$\mathbf{U} = \frac{\partial \mathbf{r}}{\partial t} - \mathbf{J} \qquad (4.3.7)$$

Tröghetskraften **B** försvinner vid stationäritet ty då är alla tidsderivator noll.

Den inre kraften eller töjningen **T** har riktning i tangentiell led **t** samt amplitud T

$$\mathbf{T} = T\mathbf{t} \qquad (4.3.8)$$

Om sträckningen i kabeln betecknas ε och elasticitetsmodulen E så gäller

$$\varepsilon = \frac{\partial S}{\partial s} - 1 \qquad (4.3.9)$$

$$\varepsilon = eT \qquad (4.3.10)$$

där

$$e = 1 / EA \qquad (4.3.11)$$

För en oelastisk kabel gäller ε = 0.

Det återstår att bestämma den hydrodynamiska kraften **F** som orsakas av motståndet i vattnet. För en kabel med cirkulär tvärsnittsarea (cylinder) kan denna kraft per längdenhet sättas till

$$\mathbf{F} = -(1/2)\rho d\{\pi C_t(\mathbf{U}\cdot\mathbf{t})|\mathbf{U}\cdot\mathbf{t}|\mathbf{t} + C_n|\mathbf{U}-(\mathbf{U}\cdot\mathbf{t})\mathbf{t}|[\mathbf{U}-(\mathbf{U}\cdot\mathbf{t})\mathbf{t}]\} \qquad (4.3.12)$$

där d är kabelns tvärsnittdiameter och C_t och C_n utgör hydrodynamiska koefficienter i tangentiell led (**t**) och normalled (**n**).

Uttrycken för de fyra kraftkomponenterna i ekvation (4.3.1) finns nu tillgängliga att sätta in.

Hastigheterna (V_t,V_n,V_b) i de tre riktingarna (**t**,**n**,**b**) för en punkt på kabeln ges av

$$\frac{\partial \mathbf{r}}{\partial t} = V_t\mathbf{t} + V_n\mathbf{n} + V_b\mathbf{b} \qquad (4.3.13)$$

4.4 Skalära kraftekvationer

Kraftekvationen (4.3.1) gäller för alla tre riktningarna (**t**,**n**,**b**). Genom att ta komponenterna i dessa tre riktningar var för sig så erhåller man tre skalära ekvationer.

I det följande visas hur man erhåller komponenterna i de tre riktningarna för första termen i ekvation (4.3.1), nämligen

$$\frac{\partial}{\partial S}\mathbf{T}$$

Eftersom de flesta termer kommer att innehålla derivator av basvektorerna **t**, **n** och **b** så beräknas dessa derivator först.

Ekvation (4.2.1) ger sambandet

$$(\mathbf{t}\quad \mathbf{n}\quad \mathbf{b}) = (\mathbf{i}\quad \mathbf{j}\quad \mathbf{k})\mathbf{L}_{JK}$$

Detta uttryck deriveras nu med avseende på s. Eftersom det jordfasta basvektorerna (**i**,**j**,**k**) är *oberoende av s* (*och t*) så deriveras högerledet endast med avseende på $\mathbf{L}_{JK}$. Denna är en produkt av tre matriser enligt (4.2.3) och därför används produktderivering. För ytterligare detaljer om detta hänvisas till Ablow och Schechter's artikel [1].

Resultatet av denna derivering blir

$$\frac{\partial}{\partial s}(\mathbf{t}\quad \mathbf{n}\quad \mathbf{b}) = (\mathbf{t}\quad \mathbf{n}\quad \mathbf{b})\left[\frac{\partial\theta}{\partial s}\begin{pmatrix} 0 & 0 & \cos\phi \\ 0 & 0 & -\sin\phi \\ -\cos\phi & \sin\phi & 0 \end{pmatrix} + \frac{\partial\phi}{\partial s}\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right] \qquad (4.4.1)$$

De tre s-derivatorna av **t**, **n** och **b** är alltså

$$\frac{\partial \mathbf{t}}{\partial s} = \mathbf{b}\frac{\partial\theta}{\partial s}(-\cos\phi) + \mathbf{n}\frac{\partial\phi}{\partial s}$$

$$\frac{\partial \mathbf{n}}{\partial s} = \mathbf{b}\frac{\partial\theta}{\partial s}\sin\phi - \mathbf{t}\frac{\partial\phi}{\partial s} \qquad (4.4.2)$$

$$\frac{\partial \mathbf{b}}{\partial s} = \mathbf{t}\frac{\partial\theta}{\partial s}\cos\phi - \mathbf{n}\frac{\partial\theta}{\partial s}\sin\phi$$

Med kännedom om dessa derivator (här behövs endast derivatan av **t**) kan komponenterna i de tre riktningarna av första termen

$$\frac{\partial}{\partial S}\mathbf{T}$$

beräknas med produktderivering enligt

$$\frac{\partial}{\partial S}\mathbf{T} = \frac{\partial}{\partial s}(T\mathbf{t}) = \frac{\partial T}{\partial s}\mathbf{t} + T\frac{\partial \mathbf{t}}{\partial s} = \frac{\partial T}{\partial s}\mathbf{t} + T\left(\mathbf{b}\frac{\partial\theta}{\partial s}(-\cos\phi) + \mathbf{n}\frac{\partial\phi}{\partial s}\right) \qquad (4.4.3)$$

Komponenterna av denna första term är alltså

$$\frac{\partial T}{\partial s} \qquad \textbf{(t}\text{-riktning)}$$

$$T\frac{\partial \phi}{\partial s} \qquad \textbf{(n}\text{-riktning)} \qquad (4.4.4)$$

$$T\frac{\partial \theta}{\partial s}(-\cos\phi) \qquad \textbf{(b}\text{-riktning)}$$

Derivatorna av **t**, **n** och **b** med avseende på *tidsvariabeln* t kan beräknas på samma sätt som s-derivatorna. Detta används i avsnitt **4.5**.

Exemplet ovan visar hur man beräknar komponenterna i de tre riktningarna (**t**,**n**,**b**) för s-derivatan av **T** i ekvation (4.3.1). Om man på samma sätt beräknar delkomponenterna av **W**, **F** och **B** i denna ekvation så erhålls de tre skalära ekvationerna

$$\frac{\partial T}{\partial s} - w\sin\phi - (1/2)\rho d(1+\varepsilon)^{1/2}\pi C_t U_t |U_t| =$$
$$m\frac{\partial V_t}{\partial t} - (m_1 V_n - \rho A J_n)\frac{\partial \phi}{\partial t} + (m_1 V_b - \rho A J_b)\cos\phi\frac{\partial \theta}{\partial t} - mV_t\frac{\partial \varepsilon}{\partial t}/(1+\varepsilon) \qquad (4.4.5)$$

$$T\frac{\partial \phi}{\partial s} - w\cos\phi - (1/2)\rho d(1+\varepsilon)^{1/2} C_n U_n (U_n^2 + U_b^2)^{1/2} =$$
$$mV_t\frac{\partial \phi}{\partial t} + m_1\frac{\partial V_n}{\partial t} - \rho A\frac{\partial J_n}{\partial t} + (m_1 V_b - \rho A J_b)\sin\phi\frac{\partial \theta}{\partial t} - \qquad (4.4.6)$$
$$(m_1 V_n - \rho A J_n)\frac{\partial \varepsilon}{\partial t}/(1+\varepsilon)$$

$$-T\cos\phi\frac{\partial \theta}{\partial s} - (1/2)\rho d(1+\varepsilon)^{1/2} C_n U_b (U_n^2 + U_b^2)^{1/2} =$$
$$-mV_t\cos\phi\frac{\partial \theta}{\partial t} + (m_1 V_n - \rho A J_n)\sin\phi\frac{\partial \theta}{\partial t} + m_1\frac{\partial V_b}{\partial t} - \rho A\frac{\partial J_b}{\partial t} - \qquad (4.4.7)$$
$$(m_1 V_b - \rho A J_b)\frac{\partial \varepsilon}{\partial t}/(1+\varepsilon)$$

där uttrycken för de relativa hastigheterna i de tre riktningarna använts enligt

$$\mathbf{U} = \frac{\partial \mathbf{r}}{\partial t} - \mathbf{J} \tag{4.4.8}$$

d v s

$$\begin{aligned} U_t &= V_t - J_t \\ U_n &= V_n - J_n \\ U_b &= V_b - J_b \end{aligned} \tag{4.4.9}$$

4.5 Geometriska villkor

Antalet okända variabler är sex till antalet, nämligen den inre kraften T, de tre hastighetskomponenterna för en punkt på kabeln V_t,V_n och V_b (en i varje riktning), samt Eulervinklarna θ och ϕ som beskriver övergången till det jordfasta koordinatsystemet. Alla andra storheter anses kända. Eftersom antalet okända variabler är 6, men antalet ekvationer bara 3 så behövs alltså ytterligare 3 ekvationer för att kunna lösa ekvationssystemet.

Variationerna i rummet och tiden av de 6 okända variablerna är kopplade via 3 partiella differentialekvationer. Dessa erhålls genom att utnyttja oberoende derivations-ordning av ortsvektorn **r**. I det följande visas hur detta görs.

Först betraktas egenskaperna hos ortsvektorn **r**. Utan töjning av kabeln gäller att

$$\frac{\partial \mathbf{r}}{\partial s} = \mathbf{t} \tag{4.5.1}$$

$$\left| \frac{\partial \mathbf{r}}{\partial s} \right| = 1 \tag{4.5.2}$$

eftersom s kan ses som båglängd.

Vid töjning av kabeln införs S som är koordinaten längs den *töjda* kabeln. Enligt tidigare definition (4.3.9) av töjningen ε gäller att

$$\varepsilon = \frac{\partial S}{\partial s} - 1$$

Utan töjning av kabeln gäller specialfallet

$$\frac{\partial S}{\partial s} = 1$$

$$\varepsilon = 0$$

Ortsvektorn **r** beskriver positionen av en punkt på den töjda kabeln, d v s

$$\frac{\partial \mathbf{r}}{\partial s} \| \mathbf{t} \qquad (4.5.3)$$

vilket betyder att s-derivatan av **r** är parallell med **t**. Dessutom gäller att

$$\left| \frac{\partial \mathbf{r}}{\partial s} \right| = \frac{\partial S}{\partial s} \qquad (4.5.4)$$

vilket ger att uttrycket för derivatan av **r** blir

$$\frac{\partial \mathbf{r}}{\partial s} = \frac{\partial S}{\partial s}\mathbf{t} = (1+\varepsilon)\mathbf{t} \qquad (4.5.5)$$

Om man nu utnyttjar *oberoende derivationsordning* av **r** enligt

$$\frac{\partial}{\partial t}\left(\frac{\partial \mathbf{r}}{\partial s}\right) = \frac{\partial}{\partial s}\left(\frac{\partial \mathbf{r}}{\partial t}\right) \qquad (4.5.6)$$

så kan detta, med användande av s-derivatan av **r** (4.5.5), skrivas

$$\frac{\partial}{\partial t}((1+\varepsilon)\mathbf{t}) = \frac{\partial}{\partial s}\left(\frac{\partial \mathbf{r}}{\partial t}\right) \qquad (4.5.7)$$

Efter produktderivering av vänsterledet blir detta ekvivalent med

$$\frac{\partial \varepsilon}{\partial t}\mathbf{t} + (1+\varepsilon)\frac{\partial \mathbf{t}}{\partial t} = \frac{\partial}{\partial s}\left(\frac{\partial \mathbf{r}}{\partial t}\right) \tag{4.5.8}$$

Om man nu utnyttjar (4.4.2) fast med s-derivator ersatta med t-derivator (erhålls på samma sätt) så kan man ersätta t-derivatan av **t** i ekvation (4.5.8) vilket ger

$$\frac{\partial \varepsilon}{\partial t}\mathbf{t} + (1+\varepsilon)\left(\mathbf{b}\frac{\partial \theta}{\partial t}(-\cos\phi) + \mathbf{n}\frac{\partial \phi}{\partial t}\right) = \frac{\partial}{\partial s}\left(\frac{\partial \mathbf{r}}{\partial t}\right) \tag{4.5.9}$$

Vänsterledet är nu fullständigt uppdelat i skalära komponenter i riktingarna (**t**,**n**,**b**). Det återstår att även dela upp högerledet.

Enligt tidigare definition (4.3.13) ges hastigheterna (V_t,V_n,V_b) för en punkt på kabeln av

$$\frac{\partial \mathbf{r}}{\partial t} = V_t\mathbf{t} + V_n\mathbf{n} + V_b\mathbf{b}$$

Genom att utnyttja detta kan ekvation (4.5.9) skrivas

$$\frac{\partial \varepsilon}{\partial t}\mathbf{t} + (1+\varepsilon)\left(\mathbf{b}\frac{\partial \theta}{\partial t}(-\cos\phi) + \mathbf{n}\frac{\partial \phi}{\partial t}\right) = \frac{\partial}{\partial s}(V_t\mathbf{t} + V_n\mathbf{n} + V_b\mathbf{b}) \tag{4.5.10}$$

Produktderivering av högerledet ger

$$\frac{\partial \varepsilon}{\partial t}\mathbf{t} + (1+\varepsilon)\left(\mathbf{b}\frac{\partial \theta}{\partial t}(-\cos\phi) + \mathbf{n}\frac{\partial \phi}{\partial t}\right) = \frac{\partial V_t}{\partial s}\mathbf{t} + V_t\frac{\partial \mathbf{t}}{\partial s} + \frac{\partial V_n}{\partial s}\mathbf{n} + V_n\frac{\partial \mathbf{n}}{\partial s} + \frac{\partial V_b}{\partial s}\mathbf{b} + V_b\frac{\partial \mathbf{b}}{\partial s} \tag{4.5.11}$$

Med eliminering av s-derivatorna av **t**, **n** och **b** enligt (4.4.2) erhåller man

$$\frac{\partial \varepsilon}{\partial t}\mathbf{t}+(1+\varepsilon)\left(\mathbf{b}\frac{\partial \theta}{\partial t}(-\cos\phi)+\mathbf{n}\frac{\partial \phi}{\partial t}\right)=\frac{\partial V_t}{\partial s}\mathbf{t}+V_t\left(\mathbf{b}\frac{\partial \theta}{\partial s}(-\cos\phi)+\mathbf{n}\frac{\partial \phi}{\partial s}\right)+$$
$$\frac{\partial V_n}{\partial s}\mathbf{n}+V_n\left(\mathbf{b}\frac{\partial \theta}{\partial s}\sin\phi-\mathbf{t}\frac{\partial \phi}{\partial s}\right)+\frac{\partial V_b}{\partial s}\mathbf{b}+V_b\left(\mathbf{t}\frac{\partial \theta}{\partial s}\cos\phi-\mathbf{n}\frac{\partial \theta}{\partial s}\sin\phi\right) \qquad (4.5.12)$$

Nu är båda leden uppdelade i komponenter i de tre riktningarna **t**, **n** och **b**. Genom att ta dessa komponenter i var riktning för sig så erhålls följande tre skalära ekvationer

$$\frac{\partial \varepsilon}{\partial t}=\frac{\partial V_t}{\partial s}-V_n\frac{\partial \phi}{\partial s}+V_b\cos\phi\frac{\partial \theta}{\partial s} \qquad (4.5.13)$$

$$(1+\varepsilon)\frac{\partial \phi}{\partial t}=\frac{\partial V_n}{\partial s}+V_t\frac{\partial \phi}{\partial s}-V_b\sin\phi\frac{\partial \theta}{\partial s} \qquad (4.5.14)$$

$$-(1+\varepsilon)\cos\phi\frac{\partial \theta}{\partial t}=\frac{\partial V_b}{\partial s}-V_t\cos\phi\frac{\partial \theta}{\partial s}+V_n\sin\phi\frac{\partial \theta}{\partial s} \qquad (4.5.15)$$

4.6 Resulterande differentialekvation

De sex ekvationerna (4.4.5) - (4.4.7) och (4.5.13) - (4.5.15) utgör det system av olinjära partiella differentialekvationer som beskriver en kabels rörelse i tiden t, längs sträckan s. Om man skriver samman dessa ekvationer på matrisform med de sex okända variablerna samlade i en vektor y enligt

$$y=(T,V_t,V_n,V_b,\theta,\phi)^T \qquad (4.6.1)$$

så kan differentialekvationen skrivas

$$M\frac{\partial y}{\partial s}=N\frac{\partial y}{\partial t}+q \qquad (4.6.2)$$

där matriserna M, N och q ser ut enligt följande:

$$M=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & V_b\cos\phi & -V_n \\ 0 & 0 & 1 & 0 & -V_b\sin\phi & V_t \\ 0 & 0 & 0 & 1 & V_n\sin\phi - V_t\cos\phi & 0 \\ 0 & 0 & 0 & 0 & -T\cos\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & T \end{pmatrix}$$

$$N=\begin{pmatrix} -meV_t/(1+eT) & m & 0 & 0 & (m_1V_b-\rho AJ_b)\cos\phi & -(m_1V_n-\rho AJ_n) \\ e & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1+eT \\ 0 & 0 & 0 & 0 & -(1+eT)\cos\phi & 0 \\ -e(m_1V_b-\rho AJ_b)/(1+eT) & 0 & 0 & m_1 & (m_1V_n-\rho AJ_n)\sin\phi - mV_t\cos\phi & 0 \\ -e(m_1V_n-\rho AJ_n)/(1+eT) & 0 & m_1 & 0 & -(m_1V_b-\rho AJ_b)\sin\phi & mV_t \end{pmatrix}$$

$$q=\begin{pmatrix} w\sin\phi+(1/2)\rho d(1+eT)^{1/2}\pi C_tU_t|U_t| \\ 0 \\ 0 \\ 0 \\ (1/2)\rho d(1+eT)^{1/2}C_nU_b(U_n^2+U_b^2)^{1/2}-\rho A\dfrac{\partial J_b}{\partial t} \\ w\cos\phi+(1/2)\rho d(1+eT)^{1/2}C_nU_n(U_n^2+U_b^2)^{1/2}-\rho A\dfrac{\partial J_n}{\partial t} \end{pmatrix}$$

Sammanfattningsvis har man alltså i ekvation (4.6.2) ett system med sex olinjära partiella differentialekvationer som beror av tidvariabeln t och rumsvariabeln s. Till detta hör sedan ett antal randvillkor i varje ände av kabeln.

4.7 Randvillkor i båtänden

Ett naturligt randvillkor i båtänden är ubåtens hastighet och position.

Transformationsmatrisen $\mathbf{L_{JK}}$ ger sambandet mellan ubåtens hastighet (V_1,V_2,V_3) i det jordfasta koordinatsystemet S_J, och hastigheten (V_t,V_n,V_b) för denna punkt uttryckt i kabelns koordinatsystem S_K. Detta kan skrivas

$$(V_t \quad V_n \quad V_b)=(V_1 \quad V_2 \quad V_3)\mathbf{L_{JK}} \tag{4.7.1}$$

vilket, med användande av (4.2.4) ger de tre randvillkoren i båtändan utskrivna i skalär form

$$\begin{aligned} V_t &= V_1\cos\theta\cos\phi - V_2\sin\theta\cos\phi - V_3\sin\phi \\ V_n &= -V_1\cos\theta\sin\phi + V_2\sin\theta\sin\phi - V_3\cos\phi \\ V_b &= V_1\sin\theta + V_2\cos\theta \end{aligned} \tag{4.7.2}$$

En lösning till differentialekvationen (4.6.2) måste uppfylla dessa tre ekvationer vid alla tidpunkter. De måste vara uppfyllda i båtändan, d v s där rumsvariabeln s har sitt största värde vilket är det samma som kabelns längd.

4.8 Randvillkor i kabeländen

Randvillkoren i kabeländan, d v s den fria änden av kabeln beror på vilka yttre krafter som verkar på den.

I fallet då ingen kropp bogseras i kabeländen så blir den inre kraften T noll längst ut på kabeln. Eftersom detta resulterar i att matrisen M blir singulär kan detta randvillkor ej användas.

Istället ansätter man randvillkoren en liten bit Δs in på kabeln. Man antar då att kabeln uppför sig som en rak, stel stång längs denna sträcka i slutet på kabeln. Man kan då ansätta två randvillkor p g a att Eulervinklarna ej förändras i kabeländen, nämligen

$$\frac{\partial \theta}{\partial s} = 0 \qquad (4.8.1)$$

$$\frac{\partial \phi}{\partial s} = 0 \qquad (4.8.2)$$

Dessutom antar man att den inre kraften T i den punkt som ligger Δs in på kabeln är lika med den yttre kraften per längdenhet på en rak stång som släpar i vattnet.

Man använder den skalära kraftekvationen i **t**-led (4.4.5) med tidsderivatorna satta till noll. Sedan antas att w, ϕ och U_t är konstanta på sträckan Δs och att T=0 då s=0. Detta ger approximationen

$$T = \left(w \sin\phi + (1/2)\rho d C_t \pi U_t |U_t|\right)\Delta s \qquad (4.8.3)$$

4.9 Diskretisering

Den resulterande ekvationen (4.6.2)

$$M\frac{\partial y}{\partial s} = N\frac{\partial y}{\partial t} + q$$

löses genom att diskretisera någon av (eller båda) de partiella derivatorna av y. Det finns flera diskretiseringsmetoder att välja mellan. De skiljer sig åt vad beträffar noggrannhet och stabilitetsegenskaper.

I kapitel **5**, som behandlar ekvationslösning med Omsim/Omola, används *trapetsmetoden* för att diskretisera s-derivatorna. Trapetsmetoden är en kombination av explicit och implicit Euler. Tidsderivatorna diskretiseras i simuleringsverktyget Omsim (se avsnitt **5.6**).

I kapitel **6**, som behandlar ekvationslösning med Matlab, används *implicit Euler* för att diskretisera både s- och t-derivatorna.

5. Rörelsesimulering i Omsim/Omola

5.1. Numerisk metod i Omsim/Omola

Den partiella differentialekvationen som ska lösas är (4.6.2) nämligen

$$M\frac{\partial y}{\partial s} = N\frac{\partial y}{\partial t} + q \qquad (5.1.1)$$

där y är en vektor med de 6 obekanta variablerna

$$y = (T, V_t, V_n, V_b, \theta, \phi)^T$$

och där matriserna M, N och q är givna i avsnitt **4.6**.

För att lösa differentialekvationen med hjälp av simuleringsmiljön Omsim krävs att den partiella differentialekvationen först diskretiseras i någon av de två variablerna s eller t. Lämpligast är att diskretisera rumsderivatorna (s), d v s man delar in kabeln i ett antal segment och gör en differensapproximation.

Genom att göra en diskretisering i s-led så får man ett system med *ordinära* differentialekvationer i variabeln t, en för varje del av kabeln.

Man implementerar sedan en modell av *ett* sådant *kabelelement* som beskrivs med 6 ordinära differentialekvationer, en för varje variabel. I Omsim kan man sedan koppla ihop sådana element, grafiskt eller med hjälp av programmeringsspråket Omola, och simulera hela kabelns uppförande i tiden.

Kabelelementen närmast ubåten och i kabeländen (slutet på kabeln) kommer att, förutom sina ordinära differentialekvationer, beskrivas med vardera tre ekvationer för randvillkoren enligt (4.7.2) samt (4.8.1-3).

5.2 Diskretisering i s-led

Första steget är alltså att göra en diskretisering av s-derivatan av y.

Om *explicit* Euler tillämpas på ekvation (5.1.1) med avseende på variabeln s på kabelelement nr i så erhålles

$$M_{i-1}(y_i - y_{i-1})/\Delta s = N_{i-1}\frac{\partial y_{i-1}}{\partial t} + q_{i-1} \qquad i = 1..n \qquad (5.2.1)$$

där n är antalet kabelelement i s-led. Därefter tillämpas *implicit* Euler på samma ekvation (5.1.1). Då erhålles

$$M_i(y_i - y_{i-1})/\Delta s = N_i\frac{\partial y_i}{\partial t} + q_i \qquad i = 1..n \qquad (5.2.2)$$

Summan av ekvation (5.2.1) och (5.2.2) blir

$$(M_i + M_{i-1})(y_i - y_{i-1})/\Delta s = N_i\frac{\partial y_i}{\partial t} + q_i + N_{i-1}\frac{\partial y_{i-1}}{\partial t} + q_{i-1} \qquad i = 1..n \qquad (5.2.3)$$

När man använder summan av explicit och implicit Euler så brukar det kallas för *trapetsmetoden.*

De ordinära differentialekvationerna (5.2.3) skall lösas för alla tider t>=0. Ekvationen gäller för ett kabelelement med index i som är anslutet till två andra kabelelement, ett i varje ände i s-led.

Gränssnittet mellan kabelelementen utgörs av *noder*, i vilka de okända variablerna i vektorn y finns. Ekvation (5.2.3) beskriver alltså sambandet mellan variablerna y i de två noder som är anslutna till ett kabelelement.

5.3 Dynamisk lösning

Med hjälp av trapetsmetoden i s-led har problemet med att lösa kabelns form och inre kraft approximeras till ett system av *ordinära* differentialekvationer.

Om man kompletterar dessa 6n st ekvationer med de sex randvillkoren så får man följande uppsättning av 6(n+1) st olinjära ekvationer.

$$(M_i + M_{i-1})(y_i - y_{i-1}) / \Delta s = N_i \frac{\partial y_i}{\partial t} + q_i + N_{i-1} \frac{\partial y_{i-1}}{\partial t} + q_{i-1} \qquad i = 1..n$$

$$r_1(y_0) = 0$$
$$r_2(y_0) = 0$$
$$r_3(y_0) = 0$$

$$e_1(y_n) = 0$$
$$e_2(y_n) = 0$$
$$e_3(y_n) = 0$$

där r är randvillkoren i båtänden (4.7.2) och e randvillkoren i kabeländen (4.8.1-3).

Genom att implementera dessa ekvationer i Omola så kan man lösa problemet med kabelns form och inre kraft för alla tidsögonblick.

Ytterligare ekvationer behövs för att beräkna positionerna av punkterna på kabeln. Positionerna fås genom integrering av hastigheterna på kabeln.

5.4 Integrering av kabelhastigheterna

Eftersom tillståndsvariablerna (y) i ekvationssystemet (5.2.3) inte innehåller några kabelpositioner utan bara hastigheterna (V_t,V_n,V_b) i tre riktningar, så måste man lägga till tre ekvationer i varje nod för att beräkna positionerna av kabelelementen i det jordfasta koordinatsystemet.

Matrisen $\mathbf{L_{JK}}$ i (4.2.1) beskriver övergången från det det jordfasta koordinatsystemet till kabelns koordinatsystem. Låt (V_i,V_j,V_k) beteckna hastigheten för en punkt på kabeln uttryckt i det jordfasta koordinatsystemet.

Ekvation (4.2.1) ger nu

$$(V_t \quad V_n \quad V_b) = (V_i \quad V_j \quad V_k)\mathbf{L_{JK}} \tag{5.4.1}$$

där transformationen görs med hjälp av matrisen $\mathbf{L_{JK}}$ i enligt (4.2.4) d v s

$$\mathbf{L_{JK}} = \begin{pmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ -\sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \\ -\sin\phi & -\cos\phi & 0 \end{pmatrix} \tag{5.4.2}$$

Detta ger kabelhastigheterna (V_i,V_j,V_k) i det jordfasta koordinatsystemet enligt

$$(V_i \quad V_j \quad V_k) = (V_t \quad V_n \quad V_b)\mathbf{L_{JK}}^{-1} \tag{5.4.3}$$

där invertering av $\mathbf{L_{JK}}$ är lika med transponering p g a ortogonalitet.

Kabelpositionerna i kabelns koordinatsystem betecknas mx, my och mz. Detta ger hastigheterna

$$V_i = \frac{dmx}{dt} \qquad V_j = \frac{dmy}{dt} \qquad V_k = \frac{dmz}{dt}$$

Ekvation (5.4.3) kan nu skrivas

$$\begin{aligned} \frac{dmx}{dt} &= V_t\cos\theta\cos\phi - V_n\cos\theta\sin\phi + V_b\sin\theta \\ \frac{dmy}{dt} &= -V_t\sin\theta\cos\phi + V_n\sin\theta\sin\phi + V_b\cos\theta \\ \frac{dmz}{dt} &= -V_t\sin\phi - V_n\cos\phi \end{aligned} \tag{5.4.4}$$

Kabelpositionerna erhålls genom att integrera (5.4.4) med avseende på tiden.

Dessa ekvationer läggs till i varje punkt på kabeln som studeras, d v s i varje nod mellan de n st kabelelementen. Detta resulterar i ytterligare 3 ekvationer i var och en av de (n+1) st noderna.

De ekvationer som beskriver problemet är de 6(n+1) st ekvationerna i avsnitt **5.3** samt de 3(n+1) st ekvationerna ovan för integrering av kabelhastigheterna.

5.5 En orientering om Omsim

Efter diskretiseringen i s-led återstår ett antal ordinära differentialekvationer att lösa, för varje kabelelement. Nästa steg är att göra en modell i Omsim av ett sådant kabelelement. Först ges en orientering om Omsim.

Omsim är ett *objektorienterat* simuleringsverktyg i vilket man kan bygga upp bibliotek med *komponenter*, som beskrivs med matematiska uttryck. Dessa komponenter kan man sedan använda för att bygga upp en större *modell* som man vill simulera.

Komponenter kopplas ihop med hjälp av *terminaler*, ett slags gränssnitt mellan olika objekt. En sammankoppling av två terminaler innebär att man erhåller ekvationer som talar om att variablernas värden i de två terminalerna är lika. (Det finns även andra typer av terminaler, t ex nollsumme-terminaler.)

I Omsim kan man t ex bygga upp komponenter för att simulera en elektrisk krets med resistorer, kondensatorer och andra elektriska komponenter. Varje elektrisk komponent beskrivs av ett matematiskt samband mellan spänning och ström. När man implementerat de olika komponenterna kan man bygga godtyckliga elektriska kretsar och simulera dessa. Det finns även verktyg för att plotta upp tidskurvor av de variabler som man är intresserad av.

I Omsim finns numeriska metoder för att lösa differentialekvationer och olinjära ekvationssystem.

Modelleringsspråket som används i Omsim benämns *Omola* och ger stöd för objektorientering, d v s här finns möjligheter för inkapsling av objekt och ärvning. Detta innebär att man kan bygga upp färdiga bibliotek med generella komponenter som sedan kan utnyttjas av användaren dels direkt men också i modifierade former genom att utnyttja ärvning. Användarens komponenter ärver då de egenskaper som de redan implementerade bibliotekskomponenterna har, d v s dess ekvationer och variabler, medan det står fritt att utöka användar-komponenterna med ytterligare egenskaper som kan behövas i det aktuella fallet.

Omsim är implementerat i C++ och körs under Unix och X-Windows.

5.6 Simulering i Omsim

När man kör en simulering i Omsim försöker symboliska manipuleringsrutiner först att reducera antalet okända variabler. När det passet är klart återstår ett antal tillståndsvariabler (kan även vara derivator) som beskriver modellens tillstånd, samt ett ekvationssystem av de ingående tillstånden.

I ett annat av det inledande passen försöker Omsim att lösa begynnelsevärdes-problemet, nämligen att hitta värden på alla ingående variabler som satisfierar systemet av ekvationer. Man kan hjälpa Omsim på traven genom att ange startvärden som man tror ligger i närheten av en sådan lösning.

Simulering i tiden innebär en lösning av ekvationssystemet, som i många fall blir olinjärt. De ordinära differentialekvationerna som ingår i ekvationsystemet löses i Omsim med differentialalgebraiska lösare så som *DASSL*, som är en flerstegsmetod, eller *Radau5* som är en implicit Runge-Kutta-metod. Dessa två lösare är bra på styva system.

5.7 Modell och klassbibliotek

I Omola skiljer man på begreppen modell och klassbibliotek.

I *klassbiblioteket* lägger man alla de komponenter som man implementerat med hjälp av programspråket Omola. Här finns alla de objekt som är beskrivna i avsnitt **5.12** implementerade.

För att simulera ett system bygger man en *modell* med hjälp av komponenterna i biblioteket. Även detta kan göras med hjälp av programspråket Omola, men görs enklast i den grafiska editorn *Med* (Model Editor) i Omsim.

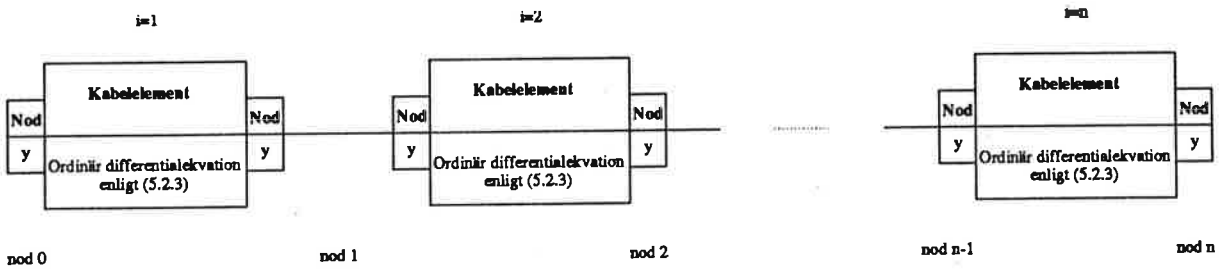
5.8 Kommandofiler

I Omsim finns möjlighet att använda s k kommandofiler (ocl-filer).

Kommandofilerna kan användas till att bygga upp ett komplett simuleringsfall med modell, plottrar och simulator. Man kan även sätta initialvärden på variabler och parametrar.

5.9 En Omsim-modell av ett kabelelement

Ekvationen (5.2.3) beskriver sambandet mellan variablerna i två noder som angränsar till ett kabelelement. Eftersom noderna fungerar som gränssnitt mellan kabelelement så är det lämpligt att modellera noderna som *terminaler* i Omsim. Det behövs alltså två huvudtyper av objekt, nämligen ett som beskriver en nod, i vilken de okända variablerna y finns, samt ett objekt som beskriver ett kabelelement med tillgång till två noder och med fysikaliska egenskaper beskrivna av ekvation (5.2.3). En sådan modell av kabeln beskrivs i figuren.



Figur 2. En Omsim-modell av kabeln.

När två noder (terminaler) N1 och N2 kopplas samman, t ex grafiskt, så erhålles ekvationer som talar om att y i N1 är lika med y i N2, vilket är vad som önskas.

5.10 Två typer av noder

Det visar sig att det är bra att införa två typer av noder i Omsim-modellen.

Om beräkningen av matriserna M, N och q görs i varje nod så kommer varje sådant uttryck att evalueras två gånger eftersom samma nod ingår som vänsternod i ett kabelelement och högernod i ett annat kabelelement. För att undvika detta onödiga beräkningsarbete så införs två olika typer av noder, *Node* och *MatrixNode*.

Beräkningen av matriserna M, N och q görs endast i *MatrixNode* medan matriserna i *Node* får sina värden vid den grafiska sammankopplingen av en *MatrixNode* och en *Node*. Ett kabelelement har nu en MatrixNode och en *Node*. Undantaget är det sista kabelelementet (*CableEnd*) som har två *MatrixNode*-noder eftersom inget kopplas i kabeländen. För att kunna överföra matrisernas värden från en MatrixNode till en Node så måste dessa (matriserna) vara terminaler.

5.11 Randvillkoren

Till den ursprungliga ekvationen (5.2.3) hör tre randvillkor i varje ände, d v s totalt sex st. Dessa ekvationer måste ingå i Omsim-modellen. Det innebär att de kabelelementen som finns längst ut i ändarna av kabeln (*CableEnd* och *BoatEnd*) kommer att ha något annorlunda egenskaper än de som återfinns i mitten av kabeln (*CableElement*). Här passar det bra att utnyttja Omolas objektorienterade egenskap, ärvning.

Båtänden (*BoatEnd*) beskrivs av ett *specialfall* av ett kabelelement (*CableElement*) som innehåller ekvation (5.2.3), samt 3 ekvationer för randvillkoren enligt (4.7.2). Kabeländen (*CableEnd*) beskrivs på samma sätt av ett specialfall av ett kabelelement med ekvation (5.2.3), samt 3 randvillkorsekvationer enligt (4.8.1-3).

Totalt fås alltså de sex ekvationerna (5.2.3) i varje kabelelement (inklusive ändarna *BoatEnd* och *CableEnd*), samt tre randvillkor i varje ände. Detta räcker till att bestämma de 6 obekanta (y) som finns i varje nod mellan kabelelementen

5.12 Klasser för att lösa kabelproblemet

I Omola finns en fördefinierad klass *Model* som måste vara superklass till den modell som man vill simulera. Därför implementeras en underklass till *Model* som beskriver kabelmodellen (*CableModel*). För att tillåta simulering av flera kabelmodeller finns även en klass *Sea*.

I *CableModel* beskrivs sådant som är gemensamt för hela kabelmodellen, d v s för varje kabelelement. Underklass till *CableModel* är *CableElement* , som beskriver ett kabelelement (ej ände). I de båda kabeländarna finns klasserna *BoatEnd* (i båtändan) och *CableEnd* (i den fria kabeländen). Dessa är båda underklasser till *CableElement*.

Varje kabelelement (*CableElement*, *BoatEnd* och *CableEnd*) består av ekvationen (5.2.3) och två noder (se nedan). I *BoatEnd* finns dessutom tre randvillkor för dess ena nod angivna. I dessa uttryck ingår båtens hastighet $V=(V_1,V_2,V_3)$ som är given. I *CableEnd* finns randvillkoren för den fria kabeländen angivna i den ena noden.

Klassen *Node* beskriver en generell nod med terminaler för de okända variablerna y, M, N och q. *MatrixNod*e är en underklass till *Node*, och innehåller ekvationer för y, M, N och q. Värden för y, M, N och q överförs från en *MatrixNode* till en *Node* med en (grafisk) förbindelse.

Både *BoatEnd* och *CableElement* har en *MatrixNode* och en *Node* medan *CableEnd* har två *MatrixNode*-noder. Detta beror på att noden i kabeländan inte får värden för matriserna M, N och q genom någon grafisk koppling. Denna nod måste därför också vara en *MatrixNode*.

I figuren visas ett OMT-diagram över de beskrivna klasserna och deras relationer.

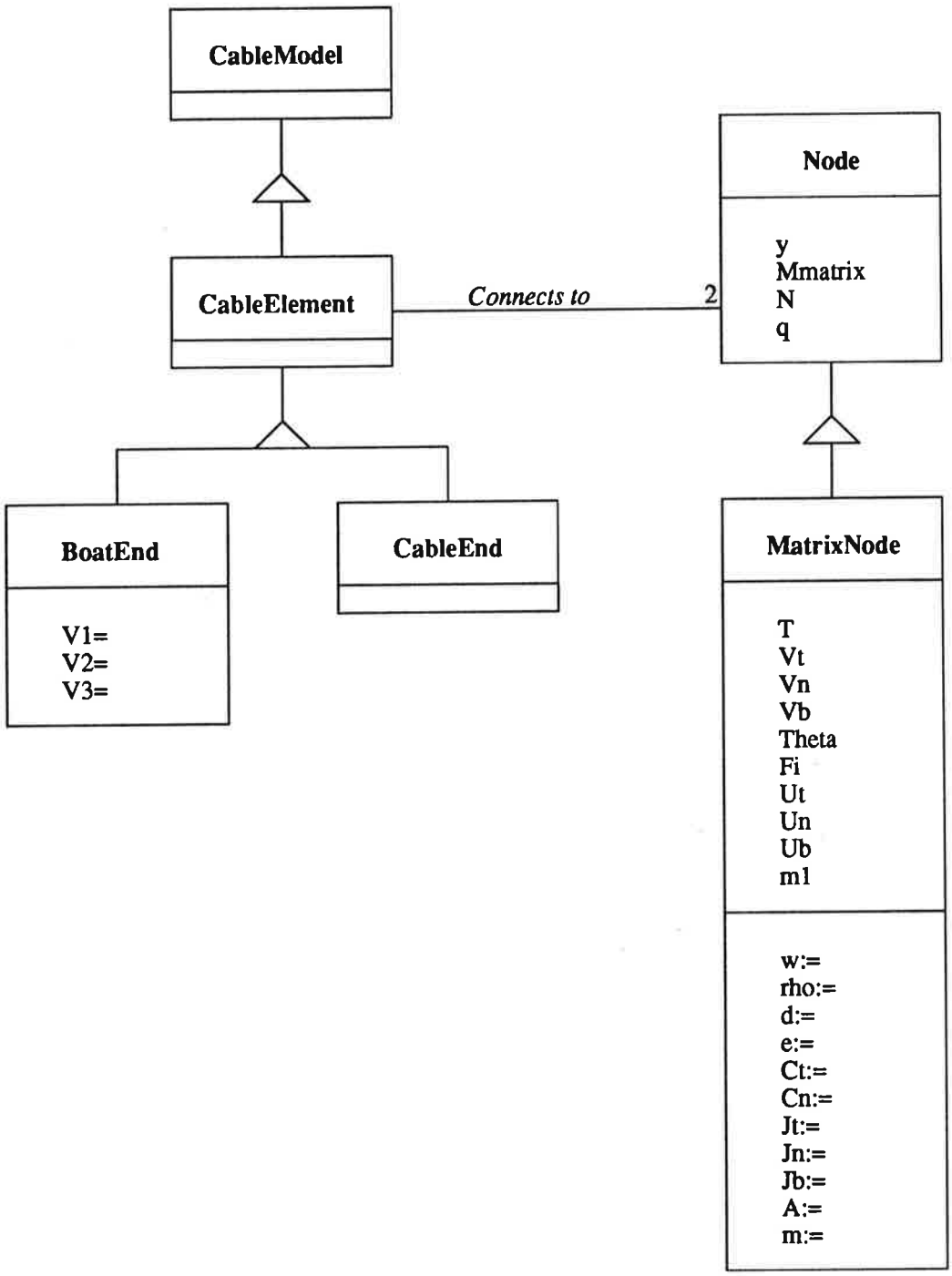


Fig 3. Omola-klasser för att lösa kabelproblemet.

5.13 Dynamisk lösning - Sammanfattning

Vid beräkning av kabelns dynamiska uppförande ingår alltså följande steg:

1. Bestäm ubåtens hastighet $V=(V_1,V_2,V_3)$ under hela det tidsförlopp som skall simuleras. Se till att uttrycket för denna hastighet finns tillgänglig i kabelelementet närmast båten (*BoatEnd*).

2. Bestäm vilket starttillstånd som kabeln befinner sig i och initiera värdena på variablerna i y för varje kabelelement inklusive ändarna. Initieringen görs m h a en kommandofil. Om inga startvärden sätts manuellt blir dessa automatiskt 0 (default). Ett typiskt starttillstånd kan vara rak liggande kabel med vinklarna (kurs och trim) noll.

3. Se till att de valda startvärden på variablerna i y stämmer överens med startvärden för kabelns positioner (mx,my,mz). Om man väljer att starta med rak liggande kabel blir alla positionerna 0 initialt, förutom en förskjutning (bias) längs med kabeln (i-axeln i det jordfasta koordinatsystemet) vilket motsvarar längden av det aktuella elementet (Δs). Denna förskjutning adderas i efterhand.

4. Gör övriga definieringar i kommandofilen. Definiera en simulator och en resultatfil, dit värdena på variablerna skrivs vid valda tidpunkter under simuleringen.

5. Exekvera kommandofilen. Simulering är nu möjlig.

6. Kör simuleringen så länge som önskas.

7. När simuleringen är klar finns värdena på kabelelementens positioner (mx,my,mz) i varje tidpunkt i resultatfilen. För att presentera kabelns form använd Matlab och läs in positionerna från resultatfilen.

Programlistor till Omola finns i Appendix A.

6. Rörelsesimulering i Matlab

6.1 Numerisk metod i Matlab

Den partiella differentialekvationen som ska lösas är (4.6.2) nämligen

$$M\frac{\partial y}{\partial s} = N\frac{\partial y}{\partial t} + q \qquad (6.1.1)$$

där y är en vektor med de 6 obekanta variablerna

$$y = (T, V_t, V_n, V_b, \theta, \phi)^T$$

och där matriserna M, N och q är givna enligt avsnitt **4.6**.

För att lösa den *partiella* differentialekvationen (6.1.1) i Matlab bör problemet först formuleras om till ett system av *ordinära* differentialekvationer. Om man väljer att diskretisera i s-led och låta tidsderivatorna utgöra vänsterled så får man ett system av ekvationer skrivna på ODE-form (*Ordinary Differential Equation*).

I det följande används *implicit* Euler i s-led (jfr 5.2.2) vilket leder till uttryck av typen

$$\frac{\partial y_i}{\partial t} = h(y_i, y_{i-1}, t) \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1 \ldots n \qquad (6.1.2)$$

där n är antalet kabelelement (i s-led) och där varje element alltså innehåller 6 tidsderivator av de 6 okända variablerna i y. Totalt får man ett system med 6n ordinära differentialekvationer av typen (6.1.2).

Antalet okända variabler $y_{0\ldots n}$ är 6(n+1) st. Det behövs alltså 6 ekvationer ytterligare. Detta är de 3 randvillkorsekvationerna i varje ände på kabeln.

Alla tidsderivator först måste lösas ut. Dessutom skall derivatorna med avseende på variabeln s diskretiseras (med implicit Euler).

En ekvation av typen (6.1.2) kan t ex lösas med en Runge-Kutta-metod. Då man har ett system av sådana ekvationer kan detta lösas direkt i Matlab med hjälp av en färdiginbyggd funktion, t ex *ode23*. Problemet med att använda denna funktion är att den kräver endast ODE-uttryck, vilket randvillkorsekvationerna inte är.

Trots detta kan det vara lämpligt att använda sig av omskrivningen (6.1.2). En fördel med att formulera om problemet till ett system av ODE-ekvationer i tidsvariabeln t är att en *stationär* lösning kan beräknas genom att söka lösningen till det olinjära ekvationssystemet bestående av

$$h(y_i, y_{i-1}, t) = 0 \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n$$

och de 6 randvillkoren. Detta gäller eftersom alla tidsderivator är noll vid stationäritet. Lösningen till ett sådant problem kan erhållas genom att använda Newton's metod [3].

6.2 Tidsderivatorna

Det första steget är alltså att lösa ut de 6 tidsderivatorna för varje kabelelement. Dessa uttryck kommer i högerledet att innehålla endast s-derivator och oderiverade variabler. Så länge man behåller s-derivatorna så kommer varje elements uppsättning av de 6 ordinära differentialekvationerna att se likadana ut. Man kan därför betrakta ett godtycklig element i s-led.

När man sedan diskretiserar s-derivatorna så kommer ekvationerna för varje element att innehålla variabler från två noder. Detta beror på att s-derivatorna approximeras med en differenskvot i s.

Först löser man alltså ut tidsderivatorna av de 6 variablerna i y ($T, V_t, V_n, V_b, \theta, \phi$). Här används de skalära uttrycken (4.4.5-7) samt (4.5.13-15) istället för matriserna M, N och q.

Från ekvation (4.5.14) kan man med $\varepsilon = eT$ (4.3.10) lösa ut tidsderivatan av ϕ

$$\frac{\partial \phi}{\partial t} = \frac{1}{(1+eT)}\left(\frac{\partial V_n}{\partial s} + V_t \frac{\partial \phi}{\partial s} - V_b \sin\phi \frac{\partial \theta}{\partial s}\right) \qquad (6.2.1)$$

Ekvation (4.5.15) ger tidsderivatan av θ

$$\frac{\partial \theta}{\partial t} = \frac{1}{(1+eT)\cos\phi}\left(-\frac{\partial V_b}{\partial s} + V_t \cos\phi \frac{\partial \theta}{\partial s} - V_n \sin\phi \frac{\partial \theta}{\partial s}\right) \qquad (6.2.2)$$

och från ekvation (4.5.13) erhålles med $\varepsilon = eT$ derivatan av T

$$\frac{\partial T}{\partial t} = \frac{1}{e}\left(\frac{\partial V_t}{\partial s} - V_n \frac{\partial \phi}{\partial s} + V_b \cos\phi \frac{\partial \theta}{\partial s}\right) \qquad (6.2.3)$$

Nu återstår att lösa ut tidsderivatorna av V_t, V_n, V_b. Dessa kan erhållas ur ekvationerna (4.4.5-7) som dock innehåller tidsderivator av ϕ, θ och T. Genom att utnyttja de redan framräknade tidsderivatorna (6.2.1-3) av dessa, så kan man eliminera tidsderivatorna av ϕ, θ och T i dessa uttryck.

Ekvation (4.4.5) ger följande uttryck för tidsderivatan av V_t

$$\frac{\partial V_t}{\partial t} = \frac{1}{m}\Big[(m_1 V_n - \rho A J_n)\frac{\partial \phi}{\partial t} - (m_1 V_b - \rho A J_b)\cos\phi\frac{\partial \theta}{\partial t} + mV_t e\frac{\partial T}{\partial t}/(1+eT) + \frac{\partial T}{\partial s} - w\sin\phi - (1/2)\rho d(1+\varepsilon)^{1/2}\pi C_t U_t|U_t|\Big]$$

men detta uttryck innehåller tidsderivator av ϕ, θ och T. Genom att utnyttja (6.2.1-3) så kan dessa elimineras och man erhåller istället

$$\begin{aligned}\frac{\partial V_t}{\partial t} = \frac{1}{m}\Big[&(m_1 V_n - \rho A J_n)\frac{1}{(1+eT)}\left(\frac{\partial V_n}{\partial s} + V_t\frac{\partial \phi}{\partial s} - V_b\sin\phi\frac{\partial \theta}{\partial s}\right) - \\ &(m_1 V_b - \rho A J_b)\frac{1}{(1+eT)}\left(-\frac{\partial V_b}{\partial s} + V_t\cos\phi\frac{\partial \theta}{\partial s} - V_n\sin\phi\frac{\partial \theta}{\partial s}\right) + \\ &mV_t\left(\frac{\partial V_t}{\partial s} - V_n\frac{\partial \phi}{\partial s} + V_b\cos\phi\frac{\partial \theta}{\partial s}\right)/(1+eT) + \frac{\partial T}{\partial s} - w\sin\phi - \\ &(1/2)\rho d(1+eT)^{1/2}\pi C_t U_t|U_t|\Big]\end{aligned} \qquad (6.2.4)$$

Ekvationen (6.2.4) är nu skriven på önskad form, med endast derivator med avseende på variabeln s i högerledet. Dessa ska senare diskretiseras.

Nu återstår att lösa ut tidsderivatorna av V_n och V_b. Detta görs genom att utnyttja ekvationerna (4.4.6) och (4.4.7), med eliminering av tidsderivatorna av ϕ, θ och T i dessa på samma sätt som förut.

Ekvationen (4.4.6) ger tidsderivatan av V_n

$$\frac{\partial V_n}{\partial t} = \frac{1}{m_1}\Big[-mV_t\frac{\partial \phi}{\partial t} + \rho A\frac{\partial J_n}{\partial t} + (m_1 V_b - \rho A J_b)\sin\phi\frac{\partial \theta}{\partial t} + (m_1 V_n - \rho A J_n)e\frac{\partial T}{\partial t}/(1+eT) + T\frac{\partial \phi}{\partial s} - w\cos\phi - (1/2)\rho d(1+eT)^{1/2}C_n U_n(U_n^2 + U_b^2)^{1/2}\Big]$$

vilket efter eliminering av tidsderivatorna ϕ, θ och T övergår i

$$\frac{\partial V_n}{\partial t} = \frac{1}{m_1}\Bigg[-mV_t\frac{1}{(1+eT)}\left(\frac{\partial V_n}{\partial s}+V_t\frac{\partial \phi}{\partial s}-V_b\sin\phi\frac{\partial \theta}{\partial s}\right)+\rho A\frac{\partial J_n}{\partial t}+$$
$$(m_1V_b-\rho AJ_b)\sin\phi\frac{1}{(1+eT)\cos\phi}\left(-\frac{\partial V_b}{\partial s}+V_t\cos\phi\frac{\partial \theta}{\partial s}-V_n\sin\phi\frac{\partial \theta}{\partial s}\right)+ \qquad (6.2.5)$$
$$(m_1V_n-\rho AJ_n)\left(\frac{\partial V_t}{\partial s}-V_n\frac{\partial \phi}{\partial s}+V_b\cos\phi\frac{\partial \theta}{\partial s}\right)/(1+eT)+T\frac{\partial \phi}{\partial s}-$$
$$w\cos\phi-(1/2)\rho d(1+eT)^{1/2}C_nU_n(U_n^2+U_b^2)^{1/2}\Bigg]$$

Slutligen skall tidsderivatan av V_b lösas ut. Ekvation (4.4.7) ger först

$$\frac{\partial V_b}{\partial t} = \frac{1}{m_1}\Bigg[(mV_t\cos\phi-(m_1V_n-\rho AJ_n)\sin\phi)\frac{\partial \theta}{\partial t}+\rho A\frac{\partial J_b}{\partial t}+$$
$$(m_1V_b-\rho AJ_b)e\frac{\partial T}{\partial t}/(1+eT)-T\cos\phi\frac{\partial \theta}{\partial s}-$$
$$(1/2)\rho d(1+eT)^{1/2}C_nU_b(U_n^2+U_b^2)^{1/2}\Bigg]$$

vilket efter eliminering av tidsderivatorna θ och T övergår i

$$\frac{\partial V_b}{\partial t} = \frac{1}{m_1}\Bigg[(m_1V_b-\rho AJ_b)\left(\frac{\partial V_t}{\partial s}-V_n\frac{\partial \phi}{\partial s}+V_b\cos\phi\frac{\partial \theta}{\partial s}\right)/(1+eT)+$$
$$(mV_t\cos\phi-(m_1V_n-\rho AJ_n)\sin\phi)\frac{1}{(1+eT)}\left(\frac{\partial V_n}{\partial s}+V_t\frac{\partial \phi}{\partial s}-V_b\sin\phi\frac{\partial \theta}{\partial s}\right)- \qquad (6.2.6)$$
$$T\cos\phi\frac{\partial \theta}{\partial s}-(1/2)\rho d(1+eT)^{1/2}C_nU_b(U_n^2+U_b^2)^{1/2}+\rho A\frac{\partial J_b}{\partial t}+\Bigg]$$

Nu är alla sex tidsderivatorna i varje kabelelement skrivna på formen

$$\frac{\partial y_i}{\partial t} = \mathrm{h}(y_i,t,\frac{\partial y_i}{\partial s}) \qquad y=(T,V_t,V_n,V_b,\theta,\phi)^T \qquad i=1\ldots n \qquad (6.2.7)$$

Vattnets strömning antas konstant. Då kan man bortse från tidsderivatorna av **J**.

6.3 Diskretisering i s-led

Nästa steg är att diskretisera derivatorna med avseende på variabeln s i de sex ekvationerna för tidsderivatorna (6.2.1-6) som beräknats ovan. Varje s-derivata approximeras med *implicit* Euler enligt

$$\frac{\partial y_i}{\partial s} \approx \frac{y_i - y_{i-1}}{\Delta s} \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n \qquad (6.3.1)$$

d v s derivatan i s-led för ett element skrivs som en differenskvot av variablerna i de båda noderna som ansluter elementet till föregående och efterföljande element.

Efter insättning av approximationen (6.3.1) i ekvationerna för tidsderivatorna (6.2.1-6) erhålles de önskade uttrycken på ODE-formen (6.1.2) d vs

$$\frac{\partial y_i}{\partial t} = h(y_i, y_{i-1}, t) \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n$$

6.4 Stationär lösning

För att beskriva kabelns uppförande i tiden då ubåtens hastighet och kurs ändras så är det lämpligt att först lösa det stationära fallet. Denna lösning kommer att utgöra startvärden för alla variabler vid simulering av det dynamiska fallet.

Man kan visserligen starta den dynamiska simuleringen i ett läge som är icke-stationärt. Om man kör ubåten med konstant fart och kurs så kommer kabeln ändå att nå stationäritet så småningom. Det är dock en fördel om man direkt kan beräkna den stationära lösningen. Då behöver inte den dynamiska simuleringen först invänta stationäritet utan man kan ändra ubåtens hastighet och kurs direkt utifrån det stationära läget.

Då kabeln form och inre kraft (T) nått sitt stationära läge så är alla tidsderivator noll. Man skall då lösa det olinjära ekvationssystemet

$$h(y_i, y_{i-1}, t) = 0 \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n$$

kompletterat med de 6 ekvationerna för randvillkoren. Totalt blir detta ett system med 6(n+1) st olinjära ekvationer. Detta kan lösas t ex med Newton's algoritm.

Problemet med att beräkna den stationära lösningen kan ses som ett specialfall av det dynamiska fallet med tidsderivatorna satta till 0. Därför beskrivs i det följande istället hur man beräknar den dynamiska lösningen.

6.5 Diskretisering i t-led

För att beskriva kabelns dynamiska uppförande då tidsderivatorna är skilda från noll behöver man i varje tidssteg lösa systemet med de 6n st ordinära differentialekvationerna (6.1.1) nämligen

$$\frac{\partial y_i}{\partial t} = h(y_i, y_{i-1}, t) \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n \qquad (6.5.1)$$

Om man kompletterar detta med randvillkoren så erhåller man ett system med 6n st olinjära ODE samt 6 olinjära randvillkorsekvationer, totalt 6(n+1) st ekvationer.

Eftersom Matlabfunktionen *ode23*, som löser ett system med ODE-uttryck, kräver att *alla* ingående ekvationer är skrivna som ODE, så används istället en annan metod, nämligen differensapproximering av tidsderivatorna.

Varje tidsderivata approximeras enligt

$$\frac{\partial y_i}{\partial t} \approx \frac{y_{1,i} - y_{0,i}}{\Delta t} \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n$$

där index 0 anger de kända värdena av variablerna i y i föregående tidssteg, medan index 1 anger de okända värdena som ska beräknas i nästa tidssteg. Om man väljer att sätta index 1 på alla variabler i högerledet av (6.5.1) enligt

$$\frac{y_{1,i} - y_{0,i}}{\Delta t} = h(y_{1,i}, y_{1,i-1}, t) \qquad y = (T, V_t, V_n, V_b, \theta, \phi)^T \qquad i = 1...n$$

så erhåller man ett *implicit* problem till varje sådan ordinär differentialekvation. Denna metod kallas även bakåt-Euler och ger betydligt bättre stabilitet än framåt-Euler, som är en *explicit* metod och erhålls genom att sätta index 0 på alla variabler i y i högerledet.

6.6 Dynamisk lösning

Den dynamiska lösningen utgörs av kabelns form och inre kraft vid varje tidssteg när ubåtens kurs och fart varierar. Denna lösning erhålls alltså genom att lösa det olinjära systemet bestående av de 6(n+1) st ekvationerna

$$g(y_{1,i}, y_{1,i-1}, y_{0,i}, t) = -\frac{y_{1,i,} - y_{0,i}}{\Delta t} + h(y_{1,i}, y_{1,i-1}, t) = 0 \qquad i = 1...n \qquad \text{(6n ekvationer)}$$

$$\begin{aligned} r_1(y_{1,0}) &= 0 \\ r_2(y_{1,0}) &= 0 \\ r_3(y_{1,0}) &= 0 \end{aligned} \qquad \text{(3 ekvationer)}$$

$$\begin{aligned} e_1(y_{1,n}) &= 0 \\ e_2(y_{1,n}) &= 0 \\ e_3(y_{1,n}) &= 0 \end{aligned} \qquad \text{(3 ekvationer)}$$

där g är de 6 kabelekvationerna i varje nod, $r_{1...3}$ är de tre randvillkoren i båtänden och $e_{1...3}$ är de tre randvillkoren i kabeländen.

Om man samlar alla 6(n+1) variabler $y_{1,i}$ (i=0..n) i en enda lång vektor y och skriver de 6(n+1) ekvationer ovan som en vektor f, med de tre randvillkoren i båtänden (r) först, därefter kabelekvationerna (g) och sist de tre randvillkoren i kabeländen (e) så erhålls ett olinjärt ekvationssystem av typen

$$f(y_1, y_0, t) = 0 \qquad (6.6.1)$$

som ska lösas i varje tidssteg. Variablerna med index 0 är kända sedan föregående tidssteg, medan variablerna med index 1 är okända och skall beräknas.

I varje kabelnod finns variablerna

$$y = (T, V_t, V_n, V_b, \theta, \phi)^T$$

och dessa beror på tidsvariabeln t och rumsvariabeln s.

Figuren visar hur problemet kan beskrivas. I varje tidssteg är de gamla värdena $y(t_0)$ kända och de nya värdena på $y(t_1)$ ska beräknas. Denna steg-för-steg-beräkning pågår under hela simuleringstiden. Steglängden i tidsled är fix.

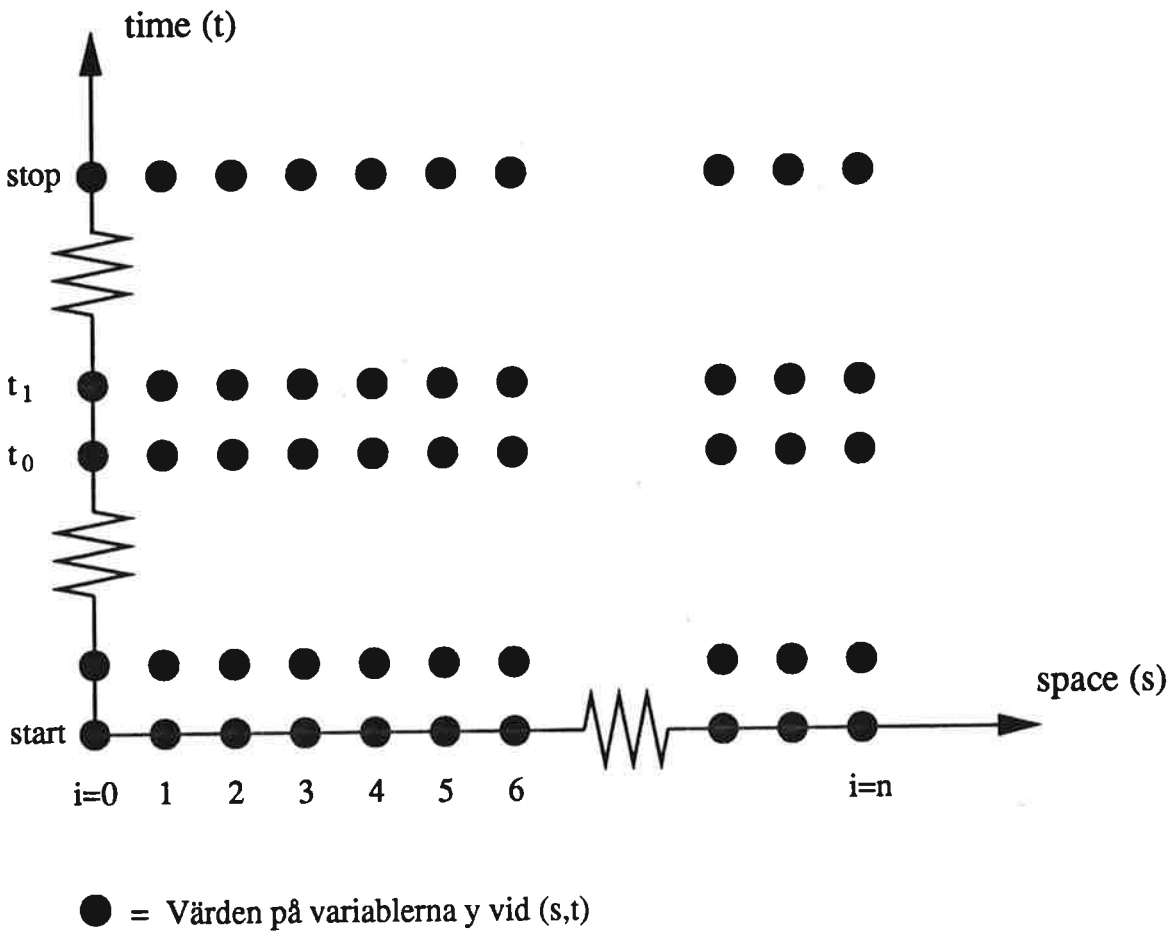


Fig 4. Illustration av problemet i varje tidssteg.

6.7 Newtons metod

Det olinjära ekvationssystemet (6.6.1) kan lösas t ex med Newtons metod. Om man skriver alla ingående variabler som en vektor y och alla ekvationer som en vektor f så återstår att i varje tidssteg lösa

$$f(y) = 0 \tag{6.7.1}$$

Här har inte tidsvariabeln t tagits med som eftersom lösningen till (6.7.1) beräknas varje gång vid en fix tidpunkt. Vektorn med de gamla värdena av y (y_0) har inte heller tagits med eftersom dessa är kända innan varje tidssteg.

Det finns olika typer av Newton-algoritmer för olinjära ekvationssystem. Den vanligaste kan skrivas

$$J^k(y^{k+1} - y^k) + f(y^k) = 0 \tag{6.7.2}$$

där J är jacobianmatrisen (funktionalmatrisen) till f och differensen (y^{k+1}-y^k), som brukar betecknas δ (delta), anger hur mycket variablerna i y skall justeras i varje iterationssteg k. Metoden innebär att man börjar innan varje tidssteg med att gissa på en lösning y^0 och sedan förbättrar man denna lösning iterativt.

Varje ekvation i f beror av 12 okända variabler i y. Av dessa finns 6 av dem i nod *i* och 6 i nod *i-1*. Detta gör att varje vektor av de partiella derivatorna av en ekvation i f, med avseende på y, kommer att innehålla 12 element skilda från 0. Dessa partiella derivator ingår i jacobianmatrisen J.

Newtons algoritm:

1. Gissa på en lösning y^k till (6.7.1).
2. Beräkna jacobianmatrisen J^k genom att använda de aktuella värdena y^k.
3. Beräkna aktuella värden på ekvationerna $f(y^k)$.
4. Beräkna förbättrade lösningen y^{k+1} ur (6.7.2) genom

$$y^{k+1} = y^k - (J^k)^{-1} f(y^k) = 0$$

5. Om differensen (y^{k+1}-y^k) inte är tillräckligt liten så gå till 2, annars färdig.

Andra typer av Newton-metoder är t ex *modifierad* Newton, där man behåller samma jacobianmatris i varje iterationssteg och på så vis undviker att beräkna denna och att invertera den vilket är tidsödande. Nackdelen med modifierad Newton är en långsammare konvergens. En annan algoritm är *kvasi* Newton.

6.8 Jacobianmatrisen

Newtons algoritm kräver att man beräknar jacobianmatrisen J till f. I detta fall blir J, som ibland kallas funktionalmatris, block-tri-diagonal. Den innehåller de partiella derivatorna av ekvationerna i f med avseende på variablerna i y enligt

$$J = \begin{pmatrix}
\frac{\partial r_1}{\partial T_0} & \cdots & \frac{\partial r_1}{\partial \phi_0} & & & & & & & & & \\
\frac{\partial r_2}{\partial T_0} & \cdots & \frac{\partial r_2}{\partial \phi_0} & & & & & & & & & \\
\frac{\partial r_3}{\partial T_0} & \cdots & \frac{\partial r_3}{\partial \phi_0} & & & & & & & & & \\
\frac{\partial g_1}{\partial T_0} & \cdots & \frac{\partial g_1}{\partial \phi_0} & \frac{\partial g_1}{\partial T_1} & \cdots & \frac{\partial g_1}{\partial \phi_1} & & & & & & \\
\vdots & & \vdots & \vdots & & \vdots & & & & & & \\
\frac{\partial g_6}{\partial T_0} & \cdots & \frac{\partial g_6}{\partial \phi_0} & \frac{\partial g_6}{\partial T_1} & \cdots & \frac{\partial g_6}{\partial \phi_1} & & & & & & \\
 & & & \frac{\partial g_7}{\partial T_1} & \cdots & \frac{\partial g_7}{\partial \phi_1} & \frac{\partial g_7}{\partial T_2} & \cdots & \frac{\partial g_7}{\partial \phi_2} & & & \\
 & & & \vdots & & \vdots & \vdots & & \vdots & & & \\
 & & & \frac{\partial g_{12}}{\partial T_1} & \cdots & \frac{\partial g_{12}}{\partial \phi_1} & \frac{\partial g_{12}}{\partial T_2} & \cdots & \frac{\partial g_{12}}{\partial \phi_2} & & & \\
 & & & & & & & & & \ddots & & \\
 & & & & & & & & & \frac{\partial g_{n-5}}{\partial T_{n-1}} & \cdots & \frac{\partial g_{n-5}}{\partial \phi_{n-1}} & \frac{\partial g_{n-5}}{\partial T_n} & \cdots & \frac{\partial g_{n-5}}{\partial \phi_n} \\
 & & & & & & & & & \vdots & & \vdots & \vdots & & \vdots \\
 & & & & & & & & & \frac{\partial g_n}{\partial T_{n-1}} & \cdots & \frac{\partial g_n}{\partial \phi_{n-1}} & \frac{\partial g_n}{\partial T_n} & \cdots & \frac{\partial g_n}{\partial \phi_n} \\
 & & & & & & & & & \frac{\partial e_1}{\partial T_{n-1}} & \cdots & \frac{\partial e_1}{\partial \phi_{n-1}} & \frac{\partial e_1}{\partial T_n} & \cdots & \frac{\partial e_1}{\partial \phi_n} \\
 & & & & & & & & & \frac{\partial e_2}{\partial T_{n-1}} & \cdots & \frac{\partial e_2}{\partial \phi_{n-1}} & \frac{\partial e_2}{\partial T_n} & \cdots & \frac{\partial e_2}{\partial \phi_n} \\
 & & & & & & & & & \frac{\partial e_3}{\partial T_{n-1}} & \cdots & \frac{\partial e_3}{\partial \phi_{n-1}} & \frac{\partial e_3}{\partial T_n} & \cdots & \frac{\partial e_3}{\partial \phi_n}
\end{pmatrix}$$

där de tre första raderna innehåller derivatorna av randvillkoren i båtänden (r) och de tre sista raderna innehåller derivatorna av randvillkoren i kabeländen (e). Varje rad innehåller 12 element eftersom alla ekvationer, förutom randvillkoren i båtänden, beror av dels de 6 variablerna i noden närmast före i s-led, dels de 6 variablerna i noden närmast efter i s-led, d v s totalt 12 variabler i y.

Beräkningen av de partiella derivatorna kan göras analytiskt genom att derivera de ingående ekvationerna i f med avseende på variablerna i y. Eftersom varje ekvation i f endast innehåller endast 12 av variablerna i y så kommer många av elementen i jacobianmatrisen att bli noll.

Ett alternativ till analytisk derivering är att beräkna de partiella derivatorna med delta-approximation, som innebär att man gör en differensapproximation med avseende på den variabel man vill derivera, och håller övriga variabler konstanta. Om man t ex ska beräkna den första partiella derivatan i J så blir detta

$$\frac{\partial f_1}{\partial T_0} \approx \frac{f_1(T_0 + \delta T_0, V_{t0}, V_{n0} \ldots\ldots) - f_1(T_0, V_{t0}, V_{n0} \ldots\ldots)}{\delta T_0} \qquad (6.8.1)$$

Om ekvationen är linjär så överensstämmer detta med det exakta värdet. Ju mer olinjäritet som förekommer desto större blir felet i denna approximation.

Fördelen med att använda delta-approximation är att samma ekvationer kan användas för att beräkna J^k och $f(y^k)$ i Newtons algoritm (punkt 2 och 3).

6.9 Integrering av kabelhastigheterna

Alla ekvationer som beskrivits ovan löses i kabelns koordinatsystem S_K.

Tillståndsvariablerna (y) i den partiella differentialekvationen innehåller inte några postioner för punkterna på kabeln utan endast kabelns hastigheter (V_t,V_n,V_b) i de tre riktningarna. Därför måste man, efter att ha beräknat de nya värdena av y i nästa tidssteg, först transformera hastigheterna (V_t,V_n,V_b) till det jordfasta koordinatsystemet och sedan utföra integrering av de nya hastigheterna (V_i,V_j,V_k) för att erhålla kabelelementens *positioner* i detta koordinatsystem.

Eftersom V_t,V_n,V_b,θ och ϕ är kända efter lösning av den partiella differential-ekvationen, via det olinjära ekvationssystemet f(y)=0, så kan kabelns *hastigheter* i varje tidssteg beräknas direkt ur (5.4.4).

För att integrera fram positionerna ur (V_i,V_j,V_k) kan man t ex använda en Runge-Kutta metod. Detta kräver dock att man har tillgång till flera värden på hastigheterna i tidsintervallet. En enklare metod är att approximera positionsändringen enligt

$$\mathbf{pos}_1 = \mathbf{pos}_0 + \begin{pmatrix} V_i & V_j & V_k \end{pmatrix} \Delta t \qquad (6.9.4)$$

d v s anta att hastigheten är ungefär konstant i varje tidintervall.

Efter detta är kabelns positioner i nästa tidssteg kända och problemet är löst.

6.10 Dynamisk lösning - Sammanfattning

Vid beräkning av kabelns dynamiska uppförande ingår alltså följande steg:

1. Bestäm ubåtens hastighet V=(V_1,V_2,V_3) vid den givna tidpunkten.

2. Initiera vektorn med de okända variablerna (y). Detta innebär att man antingen först beräknar en stationär lösning som i så fall kommer att satisfiera även de dynamiska ekvationerna, eller helt enkelt startar från ett valfritt tillstånd, t ex rak liggande kabel med vinklarna (kurs och trim) noll.

3. Se till att kabelns positioner i ubåten koordinatsystem uppfyller det valda starttillståndet. Om man väljer rak liggande kabel så kommer alla dessa positioner att vara noll förutom förskjutningen (bias) längs med kabeln (**i**-axeln i det jordfasta koordinatsystemet) vilket motsvarar längden av ett kabelelement (Δs). Denna bias adderas i efterhand.

4. Antag att kabelns tillstånd är stationärt d v s att de gamla värdena i y (yprev) är lika med de i nästa tidssteg. Detta behöver inte nödvändigtvis vara sant, men med detta antagande så får Newton-algoritmen en iteration på sig att komma i närheten av den sökta lösningen.

5. Justera variablerna i y genom att iterera med Newtons algoritm till dess att justeringen (δ) är tillräckligt liten. Då är ekvationssystemet f(y)=0 i det närmaste satisfierat. I varje iteration beräknas jacobianmatrisen med hjälp av delta-approximation av alla ekvationerna i f. Dessutom beräknas det aktuella värdet av f(y) som också behövs i Newtons algoritm.

6. När justeringen (δ) är tillräckligt liten så är tillståndsvariablerna i y kända i nästa tidssteg.

7. Integrera fram kabelpositionerna i det jordfasta koordinatsystemet med hjälp av transformationsmatrisen $\mathbf{L_{JK}}$.

8. Spara undan nuvarande värden på tillståndsvariabler (y) och kabelpostioner (pos) för senare presentation. Spara även aktuellt tidsögonblick.

9. Spara nuvarande tillståndvariabler (y) som y i föregående tidssteg (yprev). Nu är dessa båda uppsättningar lika för ett kort ögonblick, d v s y=yprev, men det ändras så fort man påbörjar nästa tidssteg.

10. Ändra eventuellt ubåtens hastighet.

11. Påbörja nästa tidssteg genom att gå till 5.

7. Validering av modellen

7.1 Syfte

Syftet med en validering är att fastställa hur väl en simulering med den valda modellen överensstämmer med verkligheten.

Validering görs genom att betrakta kabelns form under en känd manöver och jämföra detta resultat med resultatet av motsvarande modellsimulering. Den kända manövern härrör från ett verkligt experiment gjort i bassänger utrustade med mätinstrument.

Den valda manövern har utförts genom experiment under ledning av P.Rispin på David Taylor Naval Ship Research and Development Center 1980 och resultaten presenteras i en artikel i Ocean Engineering [1].

Alla fysikaliska data för kabeln finns tillgängliga i det aktuella fallet, liksom båtens hastighet och kurs under hela manövern. Kabeln är i detta fall konfigurerad för användning på ytfartyg. Dess uppförande i vattnet är dock detsamma oavsett om den bogseras av ytfartyg eller ubåt.

7.2 Validerad manöver

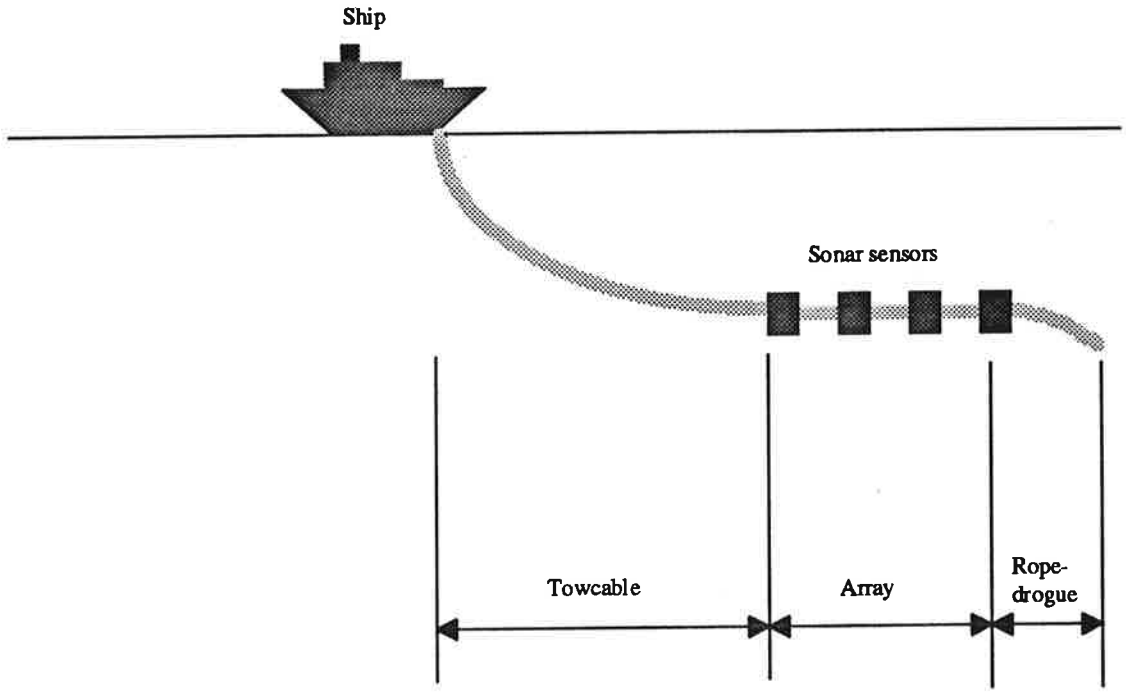
För validering används en manöver i konstant fart 18.5 knop där båten girar enligt en 375°cirkulär rörelse med radien 640 m.

Båten inleder manövern med att hålla rak kurs i konstant hastighet, vilket gör att kabeln intar sitt stationära tillstånd. Efter detta startar båten den cirkulära rörelsen varvid kabelns djup minskar. När båten slutfört ett varv av den cirkulära rörelsen fortsätter den rakt fram med kursen 15° jämfört med den inledande kursen. Till slut intar kabeln samma stationära tillstånd och djup som före den cirkulära rörelsen.

Som resultat betraktas differensen mellan kabelns inledande maximala djup och det minimala djup som kabeln intar under den cirkulära rörelsen.

7.3 Konfigurering av kabeln

Kabeln består av tre delar. Den längsta delen, benämnd *Towcable*, är även den tyngsta och finns närmast båten. Sedan följer en *Array* med sonarutrustning som är viktlös i vattnet. Därefter avslutas kabeln med en *Rope drouge* som har en viss vikt i vattnet men är förhållandevis kort.



Anledningen till att man använder en tung *Towcable* är att kabeln bogseras av ett ytfartyg och man vill få ner Arraydelen till ett visst djup. Vid bogsering med ubåt kan även *Towcable*-delen göras viktlös i vattnet.

Kabelns fysikaliska data utgörs av vikt, diameter, längd och hydrodynamiska egenskaper för dessa tre delar. Dessa data presenteras i nedanstående tabell:

	Diameter (inch)	Längd (feet)	Vikt i vattnet (lb/feet)	C_n	C_t
Rope drogue	1.0	100	0.039	1.8	0.02168
Array	3.125	900	0.0	1.8	0.00898
Towcable	1.6	2372	0.16	2.0	0.0150

Dessa data är givna i det engelska måttsystemet. Med omvandlingsformlerna

1 inch = 0.0254 m
1 feet = 0.3048 m
1 lb = 0.454 kg
1 kg = 9.81 N

så fås istället följande data

	Diameter (m)	Längd (m)	Vikt i vattnet (N/m)	C_n	C_t
Rope drogue	0.0254	30.48	0.5699	1.8	0.02168
Array	0.079375	274.32	0.0	1.8	0.00898
Towcable	0.04064	722.9856	2.3379	2.0	0.0150

Dessa data används för att beskriva kabeln vid den valda manövern.

Varje del av kabeln delas upp i ett antal kabelelement. För varje kabelelement sätts parametern Δs så att summan av dessa dellängder blir lika med kabeldelarnas respektive längd.

Vid denna simulering har 9 kabelelement använts enligt följande fördelning:

	Längd (m)	Antal kabelelement	Δs	Representerad längd (m)
Rope drogue	30.48	1	30.5	30.5
Array	274.32	3	91.5	274.5
Towcable	722.9856	5	144.6	723.0

Alla data som behövs är nu bestämda. Varje kabelelement beskrivs av:

	Beteckning	*Enhet*
Diameter	d	m
Vikt i vattnet	w	N/m
Δs	delta_s	m
C_n	Cn	-
C_t	Ct	-

7.4 Båtens hastighet

Båtens hastighet under hela manövern är konstant 18.5 knop.

Med omvandlingsformlerna

1 knop = 1.854 km/h
1 m/s = 3.6 km/h

blir denna hastighet

V = 9.5275 m/s

7.5 Resultat av validerad manöver

Simulering av den ovan beskrivna manövern har gjorts i Omsim. Som ett mått på hur väl resultatet överensstämmer med det verkliga experimentet tas differensen mellan kabelns inledande djup och kabelns djup under den cirkulära rörelsen.

Den punkt på kabeln som man mäter djupdifferensen i benämns **A** och är belägen ca 731 m från båtänden, d v s ca 8 m in på *Array*-delen. I Omsim-simuleringen tas mätpunkten **A** i kabelnoden mellan *Towcable* och *Array*, d v s ca 723 m från båtänden.

I artikeln i Ocean Engineering [1] presenteras dels resultatet av Rispins experiment, dels författarna Ablow och Schechters simuleringsresultat. Tabellen nedan visar dessa resultat samt resultatet av Omsim-simuleringen.

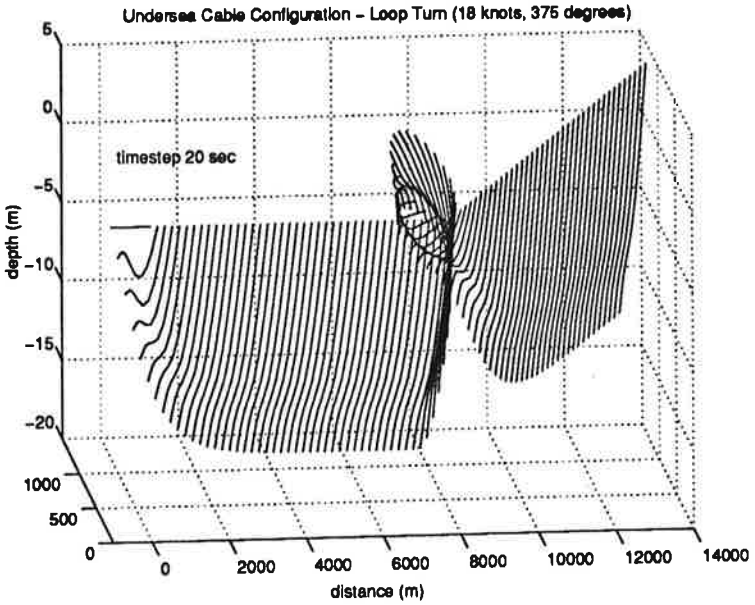
Beräknade djup av mätpunkten **A** *(m)*	*Rispins experiment*	*Ablow/Schechters simulering*	*Omsim-simulering*
Inledande djup (före cirkulär rörelse)	10.04	10.95	11.17
Minimalt djup (under cirkulär rörelse)	2.51	3.54	2.88
Differens	7.52	7.41	8.29
Avlutande djup (efter cirkulär rörelse)	10.16	10.82	11.15

Som värdena i tabellen visar är överensstämmelsen god mellan experimentellt framtagna data och resultaten av Omsim-simuleringen. Avvikelserna kan bl a antas bero på att manövern i det experimentiella fallet inte var helt cirkulär.

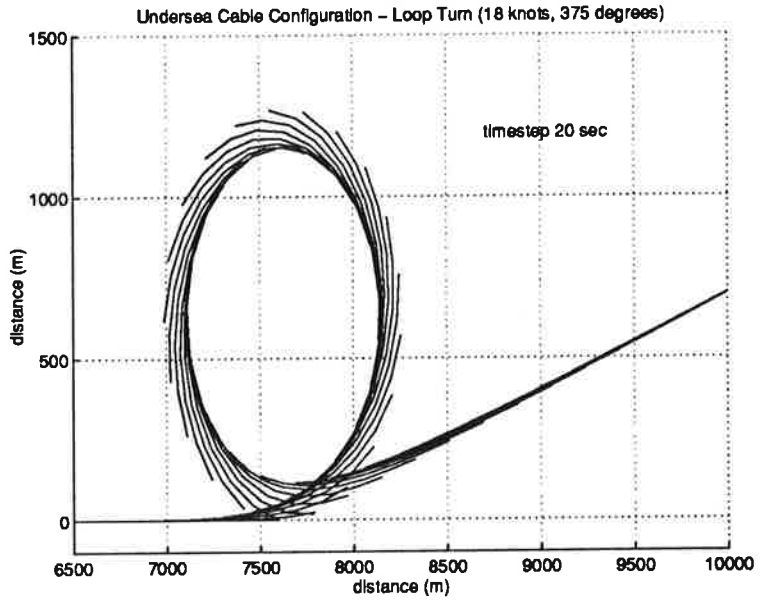
Avvikelsen gentemot Ablow och Schechters simulering kan antas bero på att olika numeriska metoder har använts. Dessutom nämner Ablow och Schechter ingenting i sin artikel om kabelns elastiska egenskaper eller vattnets strömning.

Resultatet av den validerade manövern visar att modellen med god överensstämmelse beskriver kabelns verkliga uppförande.

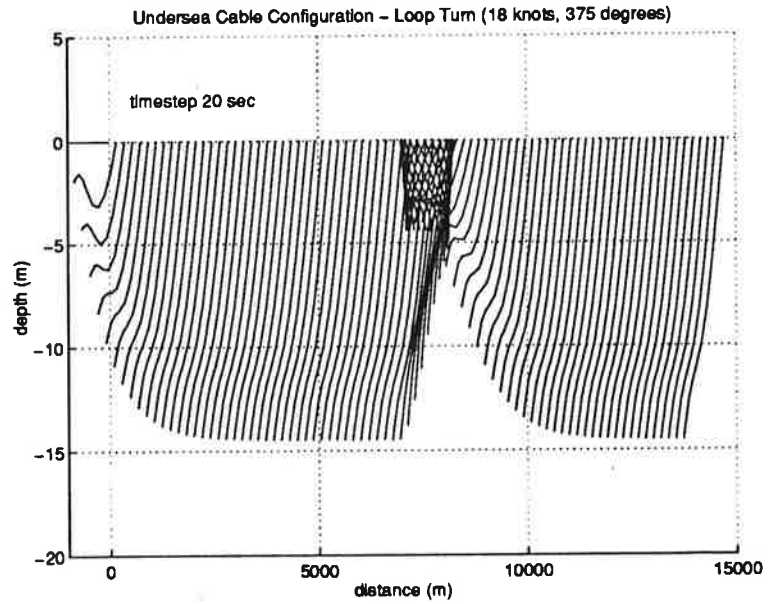
På nästa sida visas i diagram kabeln form under manövern.



3-dimensionell vy



vy uppifrån



vy från sidan

8. Simuleringsresultat

8.1 Konfigurering av kabeln

I det följande beskrivs några simuleringsresultat.

Kabeln är uppdelad i en *Towcable*-del samt en *Array*-del avsedd för sonarutrustning. Båda dessa delar är viktlösa i vattnet. De fysikaliska data som beskriver den simulerade kabeln finns angivna i nedanstående tabell:

	Diameter (m)	Längd (m)	Vikt i vattnet (N/m)	C_n	C_t
Array	0.032	200	0.0	1.2	0.01
Towcable	0.025	400	0.0	1.2	0.02

Under simuleringen har 9 kabelelement använts enligt följande fördelning:

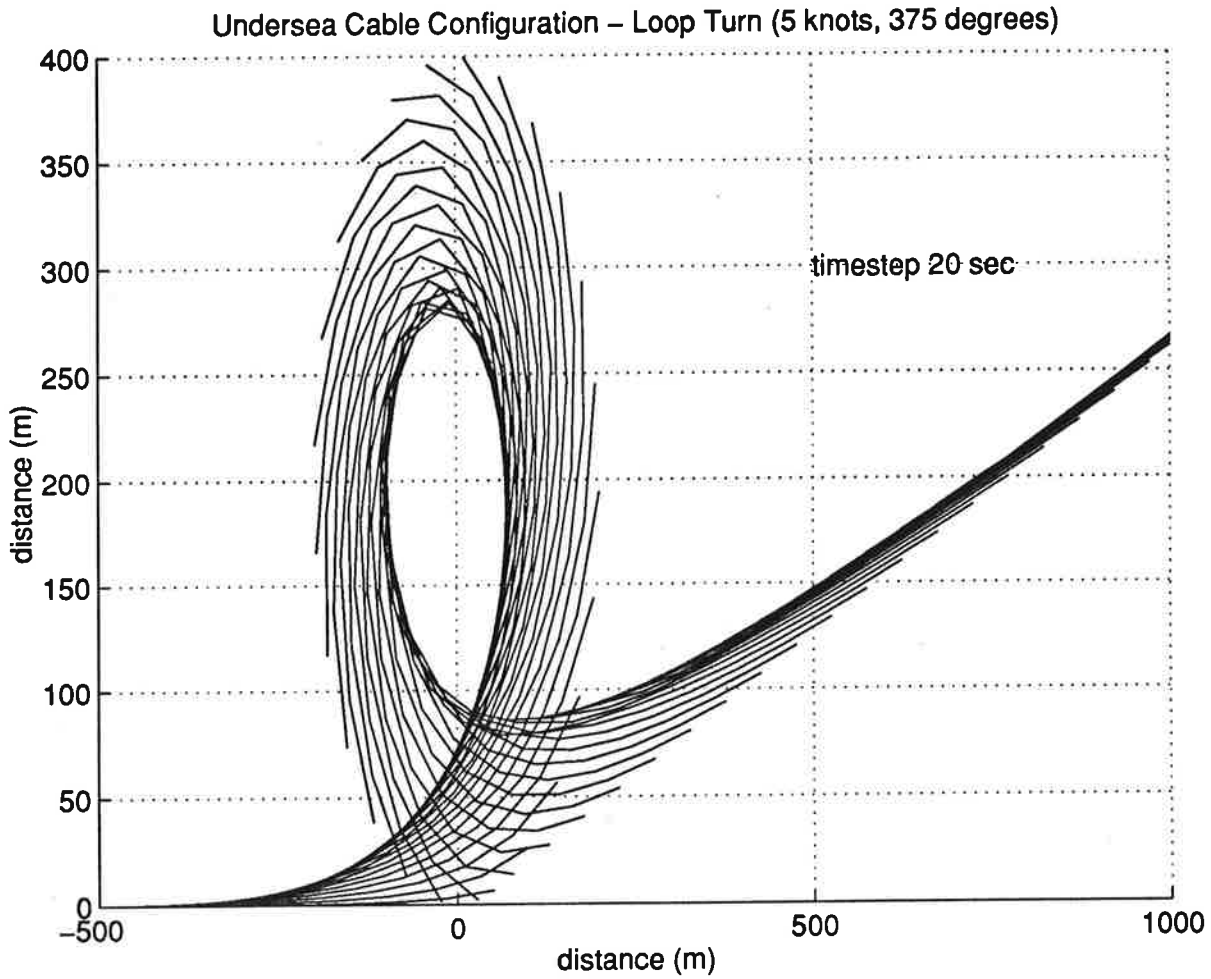
	Längd (m)	Antal kabelelement	Δs	Representerad längd (m)
Array	200	3	67	201
Towcable	400	6	67	402

Ubåtens hastighet i samtliga simuleringar är 5 knop. Kabelns form presenteras var 20:e sekund.

Den inre kraften i kabeln presenteras inte här. Under simuleringarna varierade denna från maximalt ca 3000 N i båtändan ner till ca 1000 N under en gir. Den inre kraften i kabeländen var hela tiden noll.

8.2 Cirkulär rörelse

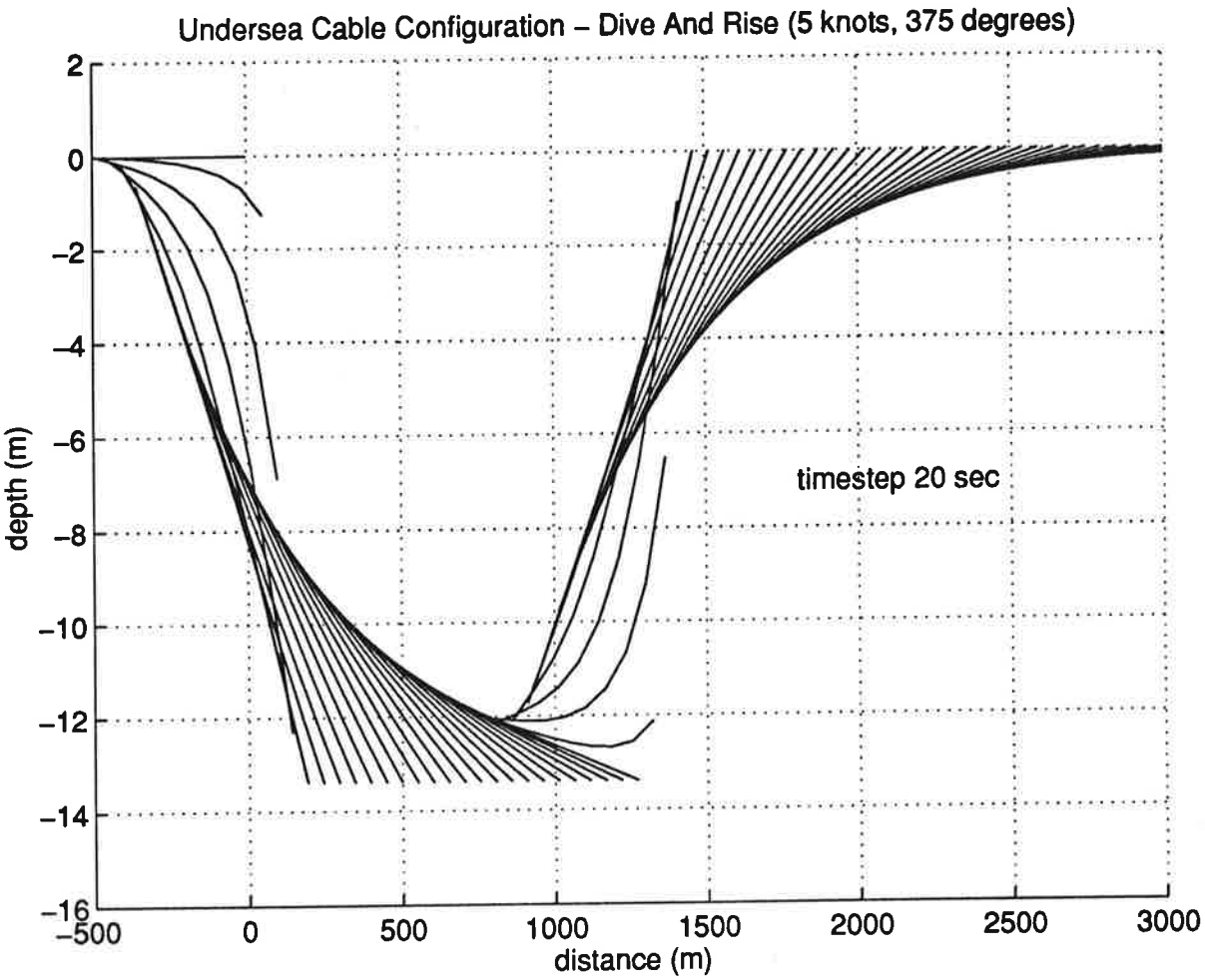
Den första manövern som presenteras är en cirkulär rörelse av samma typ som i kapitel 7 (den validerade manövern). Radien i den cirkulära rörelsen är ca 200 m.



Med en sådan här simulering kan man t ex studera hur lång tid det tar innan kabeln ligger rak igen efter giren. Dessutom kan man studera den minsta svängradien som är möjlig utan att ubåten kolliderar med kabeln

8.3 Djupändring

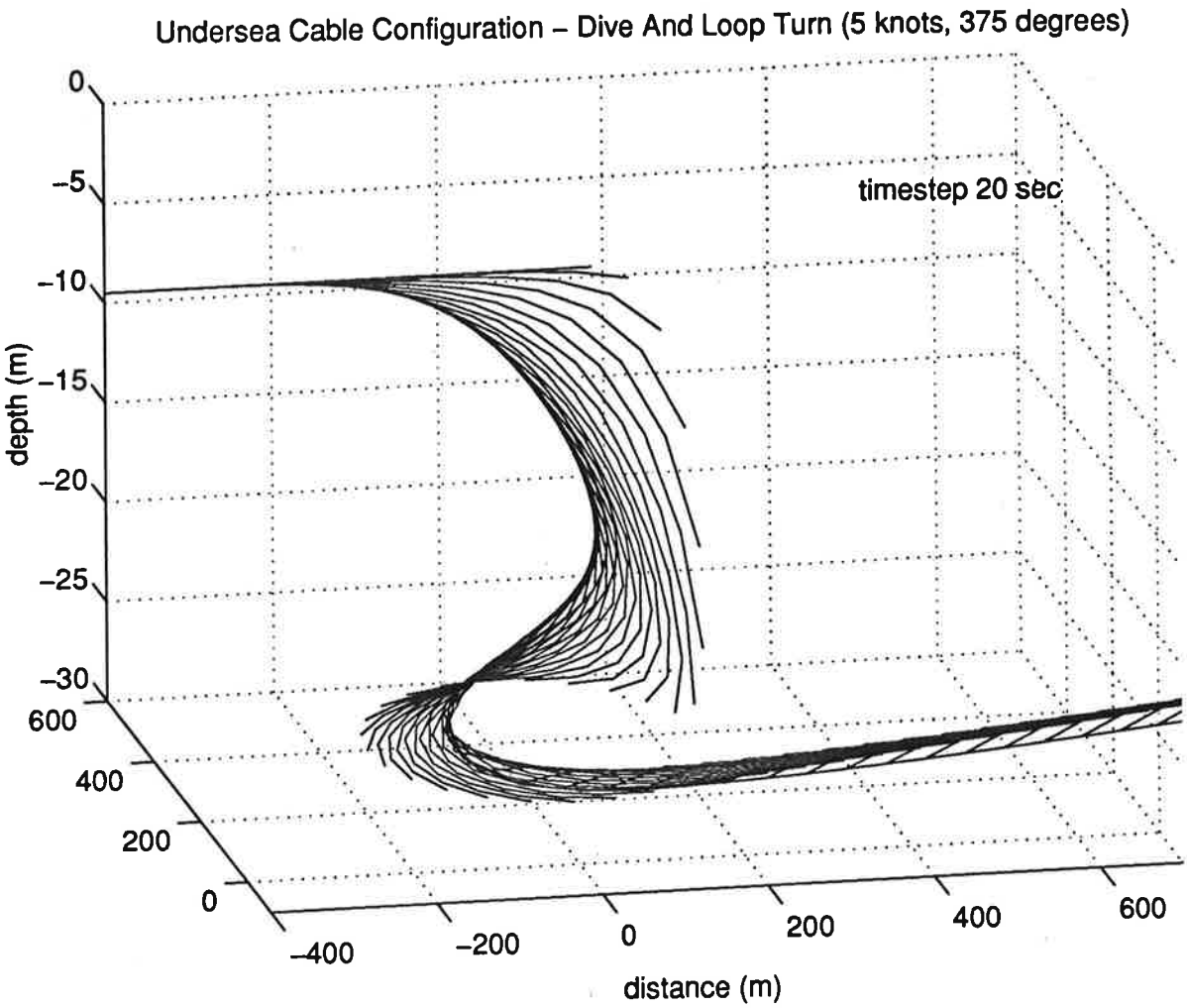
I följande manöver gör ubåten en djupändring på ca 13 m. Därefter återgår ubåten till det ursprungliga djupet.



Även vid djupändringen kan det vara lämpligt att studera den tid det tar innan kabeln ligger rak igen.

8.4 Cirkulär rörelse och djupändring

Den sista manövern som presenteras visar hur ubåten gör en djupändring samtidigt som den följer en cirkulär rörelse. Denna manöver kan sägas vara en kombination av de två föregående.



9. Slutsats

9.1 Simuleringsmiljöerna

Matlab och Omsim används för att lösa de differentialekvationer som beskriver kabelns form och inre kraft. I det följande beskrivs skillnader och likheter mellan dessa båda simuleringsmiljöer.

Kabelns egenskaper beskrivs av en *partiell* differentialekvation med tids- och rumsderivator. I både Matlab och Omsim krävs att man diskretiserar en av dessa två variabler eftersom endast *ordinära* differentialekvationer kan lösas direkt.

Omsim använder ett flertal numeriska algoritmer för att lösa ordinära differentialekvationer, bl a Runge-Kutta. Även i Matlab finns färdiga numeriska lösare av denna typ. Det krävs dock att ekvationerna skrivs på ODE-form (se avsnitt **6.1**), vilket inte behövs i Omsim som själv löser ut tidsderivatorna algebraiskt.

Den stora fördelen med Omsim är naturligtvis snabbheten. Simuleringstiderna blir avsevärt kortare än i Matlab. Detta beror delvis på att implementation i Matlab gjorts med m-filer som direktinterpreteras under exekvering. Uppsnabbning av Matlab-programmet kan därför göras genom omskrivning av m-filerna till s k MEX-filer, skrivna i C.

En annan fördel med Omsim är att man inte behöver skriva om några ekvationer utan kan implementera problemet "som det står". Det krävs dock att man formulerar om problemet till en hanterbar modell och beskrivande komponenter.

En fördel med Matlab är att man har en bättre överblick över variabler och ekvationer under simuleringen. Om någon oklarhet skulle uppstå är det enklare att hitta felet i Matlab än i Omsim, där det mesta händer "bakom kulissen".

Det finns naturligtvis stora likheter mellan Matlab och Omsim. Båda simuleringsmiljöerna stödjer operationer på matriser och vektorer. Dessutom finns möjligheter att plotta värden på variabler under simulering i båda miljöerna. Matlab erbjuder dock bättre möjligheter för grafisk presentation av t ex kabelns form i ett xyz-rum än Omsim där man endast kan plotta tidsvariabler.

9.2 Simuleringsresultaten

I kapitel **7** beskrivs hur modellen har validerats. En utvald manöver har simulerats och resultatet har jämförts med ett verkligt experiment. Resultaten har visat sig ligga nära varandra och avvikelserna kan förklaras med att manövern i experimentet inte varit exakt cirkulär.

Övriga simuleringsresultat, som redovisas i kapitel **8**, visar hur kabeln uppför sig i vattnet vid manövrar i djup- och sidled. För detaljer hänvisas till diagrammen i kapitel **8** eller de implementerade Omsim- eller Matlabsimulatorerna.

Genom att studera kraften i kabeln kan man avgöra om en monterad släphydrofon påverkar ubåtens dynamik vid olika manövrar. Med den konfigurering av kabeln som används i kapitel **8** så blir kabelns inre kraft i fästpunkten i ubåten typiskt några 1000-tal N vilket ska jämföras med ubåtens egen vikt i vattnet.

En annan intressant aspekt är att studera hur lång tid det tar för kabelns form att åter bli rak efter en gir i djup- eller sidled.

Kabelns form ligger till grund för analys av sonarprestanda, d v s hur signal-behandlingsalgoritmer ska anpassas och förbättras då sonararrayen inte är snörrät.

Referenser

1. Ablow C. M. och Schechter S. 1983

   Numerical simulation of undersea cable dynamics.
   Ocean Engineering vol 10 nr 6
   sid 443-457

2. Unger D. 1990

   Matematisk modell av släpad lina.
   Examensarbete i uppdrag av Kockums AB i samarbete med institutionen för hydrodynamik, KTH

3. Strang G. 1986

   Introduction to applied mathematics
   Wellesley-Cambridge Press

4. Andersson M. 1994

   Object-Oriented Modeling and Simulation of Hybrid Systems
   Department of Automatic Control, LTH
   Ph. D. Thesis

5. Andersson M. 1995

   Omsim and Omola, Tutorial and User's Manual
   Department of Automatic Control, LTH
   Updated from ISRN LUTFD2/TFRT-7504-SE

6. Artiklar i bilaga till Ny Teknik 1995

   Collins - A submarine downunder
   Marconi skiljer mink från Whisky

Appendix A - Omsim/Omola

Ett kabelelements variabler

Variabel	*Storhet*	*Enhet*
T	Inre kraft i kabel	N
Vt	Kabelhastighet i t-riktning	m/s
Vn	Kabelhastighet i n-riktning	m/s
Vb	Kabelhastighet i b-riktning	m/s
Theta	Kursvinkel	rad
Fi	Trimvinkel	rad
mx	Kabelposition x-led	m
my	Kabelposition y-led	m
mz	Kabelposition z-led	m
w	Kabelns vikt i vattnet	N/m
rho	Vattnets densitet	kg/m^3
d	Kabelns diameter	m
e	Elasticitetskoefficient	1/N
Ct	Hydrodynamisk motståndskoefficient (t-led)	-
Cn	Hydrodynamisk motståndskoefficient (n-led)	-
Jt	Vattnets relativa hastighet (t-led)	m/s
Jn	Vattnets relativa hastighet (n-led)	m/s
Jb	Vattnets relativa hastighet (b-led)	m/s
A	Kabelns tvärsnittsarea	m^2
m	Kabelns massa per längdenhet	kg/m
Ut	Ubåtens hastighet i kabelns koordinatsystem (t-led)	m/s
Un	Ubåtens hastighet i kabelns koordinatsystem (n-led)	m/s
Ub	Ubåtens hastighet i kabelns koordinatsystem (b-led)	m/s
m1	Effektiv massa (kabel + vatten) per längdenhet	kg/m
V1	Ubåtens hastighet i jordfast koordinatsystem (i-led)	m/s
V2	Ubåtens hastighet i jordfast koordinatsystem (j-led)	m/s
V3	Ubåtens hastighet i jordfast koordinatsystem (k-led)	m/s
delta_s	Längd av kabelelement	m

Programlistor

Följande sidor innehåller programlistor till Omsim/Omola-implementeringen.

```
LIBRARY Cable;
% Library with components describing the cable.
%%-------------------------------------------------------------
%% Model of sea.
%%-------------------------------------------------------------

Sea ISA Model;

%%-------------------------------------------------------------
%% Vector terminal.
%%-------------------------------------------------------------

VectorTerm ISA SimpleTerminal WITH
 n TYPE Integer;
 value TYPE Column [n];
 default TYPE STATIC Column [n];
END;

%%-------------------------------------------------------------
%% Column and matrix terminals.
%%-------------------------------------------------------------

SixColumnTerm ISA VectorTerm WITH
 n := 6;
END;

SixSqMatrixTerm ISA SimpleTerminal WITH
 value TYPE Matrix[6,6];
 default TYPE STATIC Matrix[6,6];
END;

%%-------------------------------------------------------------
%% A node which is the interface between two cableelements.
%%-------------------------------------------------------------

Node ISA RecordTerminal WITH
 y       ISA SixColumnTerm;
 Mmatrix  ISA SixSqMatrixTerm;
 N       ISA SixSqMatrixTerm;
 q       ISA SixColumnTerm;
END;

%%-------------------------------------------------------------
%% Node with matrixes Minvers, N and q.
%%-------------------------------------------------------------

MatrixNode ISA Node WITH
 T, Vt, Vn, Vb, Theta, Fi TYPE Real;
 mx,my,mz                 TYPE Real;

 % PARAMETERS

 % The cable's weight in the water.                    (N/m)

 w ISA Parameter WITH default := 2;
 END;

 % Density of the water.                               (kg/m^3)

 rho ISA Parameter WITH default := 1000;
```

```
END;

% Diameter of streched cable.                    (m)

d ISA Parameter WITH default := 0.1;
END;

% Elastic coefficient.                           (1/N)
% e = 1/(E*A) where E is Young's module.

e ISA Parameter WITH default := 0.0000001;
END;

% Hydrodynamic coefficient in direction t.       (-)

Ct ISA Parameter WITH default := 0.02;
END;

% Hydrodynamic coefficient in direvtion n.       (-)

Cn ISA Parameter WITH default := 2;
END;

% Velocity of the sea in direction t.            (m/s)

Jt ISA Parameter WITH default := 0;
END;

% Velocity of the sea in direction n.            (m/s)

Jn ISA Parameter WITH default := 0;
END;

% Velocity of the sea in direction b.            (m/s)

Jb ISA Parameter WITH default := 0;
END;


% Cross-sectional area of unstreched cable.      (m^2)

A TYPE Real := 3.14159265*d*d/4;

% The mass of the cable per meter.               (kg/m)

m TYPE Real:= w/9.81+rho*A;


Ut TYPE Real;
Un TYPE Real;
Ub TYPE Real;
m1 TYPE Real;

Ut    = Vt - Jt;
Un    = Vn - Jn;
Ub    = Vb - Jb;
m1    = m  + rho*A;
```

```
% Cablepositions in boat coordinate system:

mx' = Vt*cos(Theta)*cos(Fi)-Vn*cos(Theta)*sin(Fi)+Vb*sin(Theta);
my' = -Vt*sin(Theta)*cos(Fi)+Vn*sin(Theta)*sin(Fi)+Vb*cos(Theta);
mz' = -Vt*sin(Fi)-Vn*cos(Fi);

Mmatrix =

 [1, 0, 0, 0,   0,                      0;
  0, 1, 0, 0,   Vb*cos(Fi),             -Vn;
  0, 0, 1, 0,   -Vb*sin(Fi),            Vt;
  0, 0, 0, 1,   Vn*sin(Fi)-Vt*cos(Fi),  0;
  0, 0, 0, 0,   -T*cos(Fi),             0;
  0, 0, 0, 0,   0,                      T];

q     =

 [w*sin(Fi)+(1/2)*rho*d*sqrt(1+e*T)*3.14156295*Ct*Ut*abs(Ut);
  0;
  0;
  0;
  (1/2)*rho*d*sqrt(1+e*T)*Cn*Ub*sqrt(Un*Un+Ub*Ub)-rho*A*Jb';
  w*cos(Fi)+(1/2)*rho*d*sqrt(1+e*T)*Cn*Un*sqrt(Un*Un+Ub*Ub)-rho*A*Jn'];

N     =

[-m*e*Vt/(1+e*T), m, 0, 0, (m1*Vb-rho*A*Jb)*cos(Fi),-(m1*Vn-rho*A*Jn);
e, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 1+e*T;
0, 0, 0, 0, -(1+e*T)*cos(Fi), 0;
-e*(m1*Vb-rho*A*Jb)/(1+e*T), 0, 0, m1,(m1*Vn-rho*A*Jn)*sin(Fi)-m*Vt*cos(Fi), 0;
-e*(m1*Vn-rho*A*Jn)/(1+e*T), 0, m1,0, -(m1*Vb-rho*A*Jb)*sin(Fi), m*Vt];

y     =

   [T;
    Vt;
    Vn;
    Vb;
    Theta;
    Fi];
END;

%%-------------------------------------------------------------
%% Model of undersea cabel.
%%-------------------------------------------------------------

CableModel ISA Model WITH
  delta_s ISA Parameter WITH default := 50; END;
END;
```

```
%%-------------------------------------------------------------
%% Model of an element (not end) of an undersea cabel.
%%-------------------------------------------------------------

CableElement ISA CableModel WITH
 N1 ISA MatrixNode WITH
  Graphic ISA Base::Layout WITH
   x_pos := 0.0;
   y_pos := 150.0;
  END;
 END;

 N2 ISA Node WITH
  Graphic ISA Base::Layout WITH
   x_pos := 400.0;
   y_pos := 150.0;
  END;
 END;

 % Six equations describing the conditions between the variables
 % of the two nodes of the cableelement.

 (N1.Mmatrix+N2.Mmatrix)*(N1.y-N2.y)/delta_s = N1.N*N1.y'+N1.q + N2.N*N2.y'+N2.q;
END;

%%-------------------------------------------------------------
%% Model of one end of the cable, attached to the boat.
%% The interface between the boat and the cable is a boatnode.
%%-------------------------------------------------------------

BoatEnd ISA CableElement WITH
 % Velocity of the boat in three directions (should be set here).

 V1,V2,V3 TYPE Real;

 % Three boundary conditions:

 N1.Vt-V1*cos(N1.Theta)*cos(N1.Fi)+V2*sin(N1.Theta)*cos(N1.Fi)+V3*sin(N1.Fi)=0;
 N1.Vn+V1*cos(N1.Theta)*sin(N1.Fi)-V2*sin(N1.Theta)*sin(N1.Fi)+V3*cos(N1.Fi)=0;
 N1.Vb-V1*sin(N1.Theta)-V2*cos(N1.Theta)=0;
END;

%%-------------------------------------------------------------
%% Model of the end of the cable, not attached.
%%-------------------------------------------------------------

CableEnd ISA CableElement WITH
 N2 ISA MatrixNode WITH
  Graphic ISA Base::Layout WITH
   x_pos := 400.0;
   y_pos := 150.0;
  END;
 END;

 %% The three boundary conditions (with no body at the end of the cable).

 N2.T-(N2.w*sin(N2.Fi)+(1/2)*N2.rho*N2.d*N2.Ct*N2.Ut*abs(N2.Ut))*delta_s = 0;
 N2.Theta-N1.Theta= 0;
 N2.Fi-N1.Fi= 0;
END;
```

```
LIBRARY Long;
% Model used in solving the cableequations.
Sea ISA Model WITH
 Graphic ISA super::Graphic;
 BoatEnd ISA Cable::BoatEnd WITH
  Graphic ISA super::Graphic WITH
   x_pos := 76.0;
   y_pos := 251.0;
  END;
 END;
 CableElement2 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 201.0;
   y_pos := 251.0;
  END;
 END;
 CableElement3 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 326.0;
   y_pos := 251.0;
  END;
 END;
 CableElement4 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 75.0;
   y_pos := 150.0;
  END;
 END;
 CableElement5 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 200.0;
   y_pos := 150.0;
  END;
 END;
 CableElement6 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 326.0;
   y_pos := 151.0;
  END;
 END;
 CableElement7 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 76.0;
   y_pos := 51.0;
  END;
 END;
 CableElement8 ISA Cable::CableElement WITH
  Graphic ISA super::Graphic WITH
   x_pos := 200.0;
   y_pos := 50.0;
  END;
 END;
 CableEnd ISA Cable::CableEnd WITH
  Graphic ISA super::Graphic WITH
   x_pos := 326.0;
   y_pos := 51.0;
  END;
 END;
 C1 ISA Base::Connection WITH
```

```
 BoatEnd.N2 AT CableElement2.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [124.0, 250.0; 150.0, 250.0];
END;
C2 ISA Base::Connection WITH
 CableElement2.N2 AT CableElement3.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [249.0, 250.0; 275.0, 250.0];
END;
C3 ISA Base::Connection WITH
 CableElement3.N2 AT CableElement4.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [374.0, 250.0; 25.0, 149.0];
END;
C4 ISA Base::Connection WITH
 CableElement4.N2 AT CableElement5.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [124.0, 149.0; 150.0, 149.0];
END;
C5 ISA Base::Connection WITH
 CableElement5.N2 AT CableElement6.N1;
 bpoints TYPE STATIC Matrix[3, 2] := [249.0, 149.0; 268.0, 150.0; 275.0, 150.0];
END;
C6 ISA Base::Connection WITH
 CableElement6.N2 AT CableElement7.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [374.0, 150.0; 25.0, 50.0];
END;
C7 ISA Base::Connection WITH
 CableElement7.N2 AT CableElement8.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [124.0, 50.0; 150.0, 49.0];
END;
C8 ISA Base::Connection WITH
 CableElement8.N2 AT CableEnd.N1;
 bpoints TYPE STATIC Matrix[2, 2] := [249.0, 49.0; 275.0, 50.0];
END;

variables:

T0,T1,T2,T3,T4,T5,T6,T7,T8,T9                    TYPE Real;
Vt0,Vt1,Vt2,Vt3,Vt4,Vt5,Vt6,Vt7,Vt8,Vt9           TYPE Real;
Vn0,Vn1,Vn2,Vn3,Vn4,Vn5,Vn6,Vn7,Vn8,Vn9           TYPE Real;
Fi0,Fi1,Fi2,Fi3,Fi4,Fi5,Fi6,Fi7,Fi8,Fi9           TYPE Real;
mx0,mx1,mx2,mx3,mx4,mx5,mx6,mx7,mx8,mx9          TYPE Real;
my0,my1,my2,my3,my4,my5,my6,my7,my8,my9          TYPE Real;
mz0,mz1,mz2,mz3,mz4,mz5,mz6,mz7,mz8,mz9          TYPE Real;

T0 := BoatEnd.N1.T;
T1 := CableElement2.N1.T;
T2 := CableElement3.N1.T;
T3 := CableElement4.N1.T;
T4 := CableElement5.N1.T;
T5 := CableElement6.N1.T;
T6 := CableElement7.N1.T;
T7 := CableElement8.N1.T;
T8 := CableEnd.N1.T;
T9 := Cableend.N2.T;

Vt0 := BoatEnd.N1.Vt;
Vt1 := CableElement2.N1.Vt;
Vt2 := CableElement3.N1.Vt;
Vt3 := CableElement4.N1.Vt;
Vt4 := CableElement5.N1.Vt;
Vt5 := CableElement6.N1.Vt;
Vt6 := CableElement7.N1.Vt;
```

```
Vt7 := CableElement8.N1.Vt;
Vt8 := CableEnd.N1.Vt;
Vt9 := Cableend.N2.Vt;

Vn0 := BoatEnd.N1.Vn;
Vn1 := CableElement2.N1.Vn;
Vn2 := CableElement3.N1.Vn;
Vn3 := CableElement4.N1.Vn;
Vn4 := CableElement5.N1.Vn;
Vn5 := CableElement6.N1.Vn;
Vn6 := CableElement7.N1.Vn;
Vn7 := CableElement8.N1.Vn;
Vn8 := CableEnd.N1.Vn;
Vn9 := Cableend.N2.Vn;

Fi0 := BoatEnd.N1.Fi;
Fi1 := CableElement2.N1.Fi;
Fi2 := CableElement3.N1.Fi;
Fi3 := CableElement4.N1.Fi;
Fi4 := CableElement5.N1.Fi;
Fi5 := CableElement6.N1.Fi;
Fi6 := CableElement7.N1.Fi;
Fi7 := CableElement8.N1.Fi;
Fi8 := CableEnd.N1.Fi;
Fi9 := Cableend.N2.Fi;

mx0 := BoatEnd.N1.mx;
mx1 := CableElement2.N1.mx;
mx2 := CableElement3.N1.mx;
mx3 := CableElement4.N1.mx;
mx4 := CableElement5.N1.mx;
mx5 := CableElement6.N1.mx;
mx6 := CableElement7.N1.mx;
mx7 := CableElement8.N1.mx;
mx8 := CableEnd.N1.mx;
mx9 := Cableend.N2.mx;

my0 := BoatEnd.N1.my;
my1 := CableElement2.N1.my;
my2 := CableElement3.N1.my;
my3 := CableElement4.N1.my;
my4 := CableElement5.N1.my;
my5 := CableElement6.N1.my;
my6 := CableElement7.N1.my;
my7 := CableElement8.N1.my;
my8 := CableEnd.N1.my;
my9 := Cableend.N2.my;

mz0 := BoatEnd.N1.mz;
mz1 := CableElement2.N1.mz;
mz2 := CableElement3.N1.mz;
mz3 := CableElement4.N1.mz;
mz4 := CableElement5.N1.mz;
mz5 := CableElement6.N1.mz;
mz6 := CableElement7.N1.mz;
mz7 := CableElement8.N1.mz;
mz8 := CableEnd.N1.mz;
mz9 := Cableend.N2.mz;

END;
```

```
BEGIN
 Long::Sea mod;                  % Create the model

 Simulator sim(mod);             % Create the simulator
 sim.display(200,200);
 sim.stoptime := 100;
 sim.title := "Simulation of undersea cable";

 Plotter Tplot(mod);             % Create the T plotter
 Plotter Vtplot(mod);            % Create the Vt plotter
 Plotter Vnplot(mod);            % Create the Vn plotter
 Plotter Fiplot(mod);            % Create the Fi plotter

 Tplot.display(300,800);
 Vtplot.display(600,800);
 Vnplot.display(300,400);
 Fiplot.display(600,400);

 Storefile stfile(mod);          % Create a store file

 % Initial values for variables

 mod.BoatEnd.N1.T                := 3000;    % --- Towcable ---
 mod.BoatEnd.N1.Theta            := 0;
 mod.BoatEnd.N1.Fi               := 0;
 mod.BoatEnd.N1.d                := 0.025;
 mod.BoatEnd.delta_s             := 67;
 mod.BoatEnd.N1.w                := 0;
 mod.BoatEnd.N1.Cn               := 1.2;
 mod.BoatEnd.N1.Ct               := 0.02;

 mod.CableElement2.N1.T          := 2700;
 mod.CableElement2.N1.Vb         := 0;
 mod.CableElement2.N1.Vt         := 2.7;
 mod.CableElement2.N1.Vn         := 0;
 mod.CableElement2.N1.Theta      := 0;
 mod.CableElement2.N1.Fi         := 0;
 mod.CableElement2.N1.d          := 0.025;
 mod.CableElement2.delta_s       := 67;
 mod.CableElement2.N1.w          := 0;
 mod.CableElement2.N1.Cn         := 1.2;
 mod.CableElement2.N1.Ct         := 0.02;

 mod.CableElement3.N1.T          := 2000;
 mod.CableElement3.N1.Vb         := 0;
 mod.CableElement3.N1.Vt         := 2.7;
 mod.CableElement3.N1.Vn         := 0;
 mod.CableElement3.N1.Theta      := 0;
 mod.CableElement3.N1.Fi         := 0;
 mod.CableElement3.N1.d          := 0.025;
 mod.CableElement3.delta_s       := 67;
 mod.CableElement3.N1.w          := 0;
 mod.CableElement3.N1.Cn         := 1.2;
 mod.CableElement3.N1.Ct         := 0.02;

 mod.CableElement4.N1.T          := 1600;
 mod.CableElement4.N1.Vb         := 0;
 mod.CableElement4.N1.Vt         := 2.7;
 mod.CableElement4.N1.Vn         := 0;
```

```
mod.CableElement4.N1.Theta := 0;
mod.CableElement4.N1.Fi    := 0;
mod.CableElement4.N1.d     := 0.025;
mod.CableElement4.delta_s  := 67;
mod.CableElement4.N1.w     := 0;
mod.CableElement4.N1.Cn    := 1.2;
mod.CableElement4.N1.Ct    := 0.02;

mod.CableElement5.N1.T     := 1200;
mod.CableElement5.N1.Vb    := 0;
mod.CableElement5.N1.Vt    := 2.7;
mod.CableElement5.N1.Vn    := 0;
mod.CableElement5.N1.Theta := 0;
mod.CableElement5.N1.Fi    := 0;
mod.CableElement5.N1.d     := 0.025;
mod.CableElement5.delta_s  := 67;
mod.CableElement5.N1.w     := 0;
mod.CableElement5.N1.Cn    := 1.2;
mod.CableElement5.N1.Ct    := 0.02;

mod.CableElement6.N1.T     := 800;
mod.CableElement6.N1.Vb    := 0;
mod.CableElement6.N1.Vt    := 2.7;
mod.CableElement6.N1.Vn    := 0;
mod.CableElement6.N1.Theta := 0;
mod.CableElement6.N1.Fi    := 0;
mod.CableElement6.N1.d     := 0.025;
mod.CableElement6.delta_s  := 67;
mod.CableElement6.N1.w     := 0;
mod.CableElement6.N1.Cn    := 1.2;
mod.CableElement6.N1.Ct    := 0.02;

mod.CableElement7.N1.T     := 600;    % ---- Array ------
mod.CableElement7.N1.Vb    := 0;
mod.CableElement7.N1.Vt    := 2.7;
mod.CableElement7.N1.Vn    := 0;
mod.CableElement7.N1.Theta := 0;
mod.CableElement7.N1.Fi    := 0;
mod.CableElement7.N1.d     := 0.032;
mod.CableElement7.delta_s  := 67;
mod.CableElement7.N1.w     := 0;
mod.CableElement7.N1.Cn    := 1.2;
mod.CableElement7.N1.Ct    := 0.01;

mod.CableElement8.N1.T     := 400;
mod.CableElement8.N1.Vb    := 0;
mod.CableElement8.N1.Vt    := 2.7;
mod.CableElement8.N1.Vn    := 0;
mod.CableElement8.N1.Theta := 0;
mod.CableElement8.N1.Fi    := 0;
mod.CableElement8.N1.d     := 0.032;
mod.CableElement8.delta_s  := 67;
mod.CableElement8.N1.w     := 0;
mod.CableElement8.N1.Cn    := 1.2;
mod.CableElement8.N1.Ct    := 0.01;

 mod.CableEnd.N1.T         := 200;
mod.CableEnd.N1.Vb         := 0;
mod.CableEnd.N1.Vt         := 2.7;
mod.CableEnd.N1.Vn         := 0;
```

```
 mod.CableEnd.N1.Theta      := 0;
 mod.CableEnd.N1.Fi         := 0;
 mod.CableEnd.N1.d          := 0.032;
 mod.CableEnd.delta_s       := 67;
 mod.CableEnd.N1.w          := 0;
 mod.CableEnd.N1.Cn         := 1.2;
 mod.CableEnd.N1.Ct         := 0.01;

 mod.CableEnd.N2.Vb         := 0;
 mod.CableEnd.N2.Vt         := 2.7;
 mod.CableEnd.N2.Vn         := 0;
 mod.CableEnd.N2.d          := 0.032;
 mod.CableEnd.delta_s       := 67;
 mod.CableEnd.N2.w          := 0;
 mod.CableEnd.N2.Cn         := 1.2;
 mod.CableEnd.N2.Ct         := 0.01;

 Tplot.y(T0,T1,T2,T3,T4,T5,T6,T7,T8,T9);
 Vtplot.y(Vt0,Vt1,Vt2,Vt3,Vt4,Vt5,Vt6,Vt7,Vt8,Vt9);
 Vnplot.y(Vn0,Vn1,Vn2,Vn3,Vn4,Vn5,Vn6,Vn7,Vn8,Vn9);
 Fiplot.y(Fi0,Fi1,Fi2,Fi3,Fi4,Fi5,Fi6,Fi7,Fi8,Fi9);

 stfile.name := "output.dat";
 stfile.store(mx0,mx1,mx2,mx3,mx4,mx5,mx6,mx7,mx8,mx9,
          my0,my1,my2,my3,my4,my5,my6,my7,my8,my9,
          mz0,mz1,mz2,mz3,mz4,mz5,mz6,mz7,mz8,mz9);
 stfile.interval := 0.05;
 stfile.append := 1;
END;
```

```
% Matlab-file that reads output data from Omsim/Omola and plots cable shape in Matlab.

matrix=om2mat('output.dat');

x=matrix(:,2:11);                % mx
y=matrix(:,12:21);               % my
z=matrix(:,22:31);               % mz

% Plot cable shape for each time step:

for t=1:10*2:size(matrix,1);   % t=1:10*2:size(matrix,1);
 %clf;
 hold on;
 axis([-1000 9000 -100 1500 -20 2]);
 view(-5,20);
 newx = x;
 [nrows,ncols] =  size(newx);

 bias(1:nrows,1)=zeros(nrows,1);
 for k=2:6;
  bias(1:nrows,k)=-144.6*ones(nrows,1)+bias(1:nrows,k-1);
 end;
 for k=7:9;
  bias(1:nrows,k)=-91.5*ones(nrows,1)+bias(1:nrows,k-1);
 end;
 bias(1:nrows,10)=-30.5*ones(nrows,1)+bias(1:nrows,9);

 newx=x+bias;

 plot3(newx(t,1:10),y(t,1:10),-z(t,1:10)); % y(t,1:10));  %z(t,1)-z(t,1:10)
 pause(0.005);
end;
grid;

title('Undersea Cable Configuration - Loop Turn (18 knots, 375 degrees)');
text(5100,-16,'timestep 20 sec');
xlabel('distance (m)');
ylabel('depth (m)');
```

Appendix B - Matlab

Ett kabelelements variabler

Variabel	*Storhet*	*Enhet*
T	Inre kraft i kabel	N
Vt	Kabelhastighet i t-riktning	m/s
Vn	Kabelhastighet i n-riktning	m/s
Vb	Kabelhastighet i b-riktning	m/s
Theta	Kursvinkel	rad
Fi	Trimvinkel	rad
pos	Kabelpositioner (x,y,z)	m
w	Kabelns vikt i vattnet	N/m
rho	Vattnets densitet	kg/m^3
d	Kabelns diameter	m
e	Elasticitetskoefficient	1/N
Ct	Hydrodynamisk motståndskoefficient (t-led)	-
Cn	Hydrodynamisk motståndskoefficient (n-led)	-
Jt	Vattnets relativa hastighet (t-led)	m/s
Jn	Vattnets relativa hastighet (n-led)	m/s
Jb	Vattnets relativa hastighet (b-led)	m/s
A	Kabelns tvärsnittsarea	m^2
m	Kabelns massa per längdenhet	kg/m
Ut	Ubåtens hastighet i kabelns koordinatsystem (t-led)	m/s
Un	Ubåtens hastighet i kabelns koordinatsystem (n-led)	m/s
Ub	Ubåtens hastighet i kabelns koordinatsystem (b-led)	m/s
ml	Effektiv massa (kabel + vatten) per längdenhet	kg/m
V	Ubåtens hastighet (i,j,k) i jordfast koordinatsystem	m/s
delta_s	Längd av kabelelement	m

Övriga variabler

delta_t	Längd av tidssteg
y	Vektor med alla tillståndvariabler (alla kabelnoder)
yprev	Värdet av alla variabler i y i förra tidssteget
diffindex	Index till element i vektorn y som skall differentieras
delta	Steglängder för beräkning av partiella derivator
elem_n	Nummer på kabelelement, med start (=1) från båtsidan
n_elem	Antal kabelelement
r	De tre ekvationerna (randvillkoren) i båtänden
g	De sex ekvationerna i ett kabelelement
e	De tre ekvationerna (randvillkoren) i kabeländen
CV	Hastigheter (i,j,k) av (alla) kabelnoder
delta_y	Korrigering av y i varje iterationssteg i Newtons algortim
J	Jacobianmatris (=funktionalmatris)
tol	Feltolerans (stoppvillkor) i Newtons algortim
Vtime	Ubåtens hastighet i varje tidsögonblick
yres	Resultat av simulering - Tillståndsvariablerna i y
posres	Resultat av simulering - Positioner (x,y,z) av kabelnoder
tres	Resultat av simulering - De tidssteg som y och pos lagrats i

Beskrivning av m-filer

BoatEq.m

Beräknar nuvarande värdet av de tre (randvillkors-) ekvationerna i båtänden. Dessa ekvationer skall satisfieras i varje tidssteg. Ekvationerna används även vid numerisk beräkning av de partiella derivatorna som ingår i jacobianmatrisen.

BoatVel.m

Returnerar värdet av ubåtens hastighet i tre riktingar i varje tidssteg.

CableEq.m

Beräknar nuvarande värdet av de sex ekvationerna i varje kabelelement. Dessa ekvationer skall satisfieras i varje tidssteg. Ekvationerna används även vid numerisk beräkning av de partiella derivatorna som ingår i jacobianmatrisen.

CableVel.m

Beräknar kabelnodernas hastigheter i det jordfasta korrdinatsystemet.

Current.m

Beräknar de nuvarande värdena av samtliga ekvationer, d v s de tre randvillkorsekvationerna i båtänden, de sex ekvationerna i varje kabelelement samt de tre randvillkorsekvationerna i kabeländen. Dessa värden används i Newtons algoritm.

EndEq.m

Beräknar nuvarande värdet av de tre (randvillkors-) ekvationerna kabeländen. Dessa ekvationer skall satisfieras i varje tidssteg. Ekvationerna används även vid numerisk beräkning av de partiella derivatorna som ingår i jacobianmatrisen.

Iter.m

Genomför *en* iteration i Newtons algoritm. Beräknar jacobianmatrisen och justeringen av variablerna y i varje iterationsssteg.

Jacobian.m

Beräknar jacobianmatrisen som används i Newtons algortim. De partiella derivatorna som ingår som element i jacobianmatrisen beräknas numeriskt.

Param.m

Innehåller alla parametrar som används av övriga m-filer.

PlotPos.m

Plottar upp kabelnodernas positioner, d v s beskriver kabelns utseende, i varje tidssteg.

Ploty.m

Plottar upp värdet av tillståndsvariablerna $y=(T,V_t,V_n,V_b,\theta,\phi)$ i alla kabelnoder under den simulerade tiden.

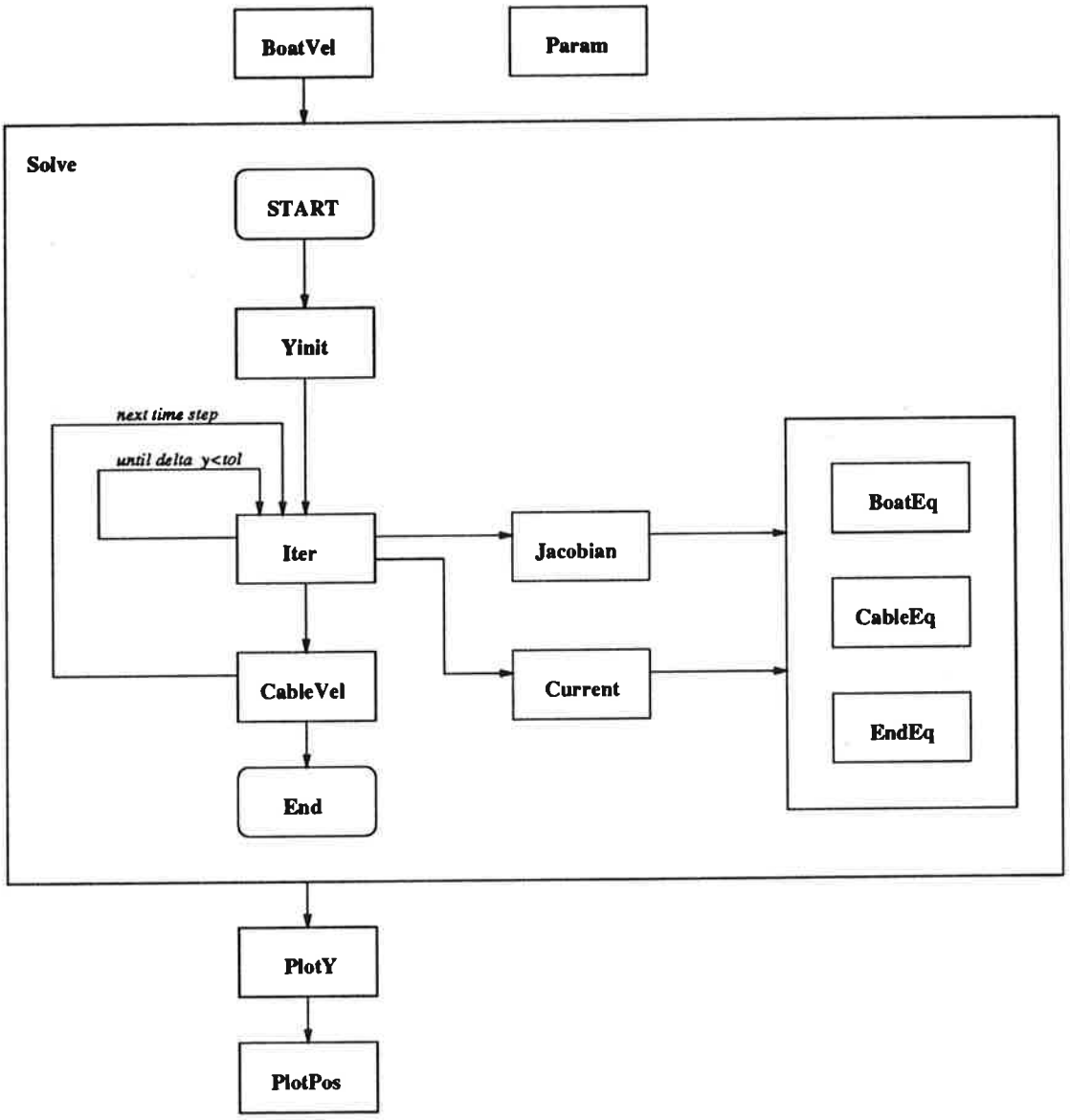
Solve.m

Beräknar kabelns form och inre kraft i varje tidssteg vid vald ubåtsmanöver.

Yinit.m

Initierar alla värden på variablerna i y. Detta görs genom att gissa en lösning som beskriver en rakt liggande kabel med en vald kritisk (lutnings-)vinkel.

Flödesdiagram



Programlistor

Följande sidor innehåller programlistor till Matlab-implementeringen.

```
function [r]=BoatEq(y,diffindex,delta,V);
%
% function [r]=BoatEq(y,diffindex,delta,V);
%
% The three equations at the boat end. These equations
% should be satisfied (=0) in each time step.
%
% Input:        y               Current values of first 6 variables in y.
%               diffindex       Index of which variable in y that should be
%                               differentiated. Used in Newton algorithm.
%               V               Velocity of boat.
%
% Output:       r               Current values of the three boat equations.

  % Differentiate one of the variables:

  if diffindex > 0
    y(diffindex)=y(diffindex)+delta;
  end;

  % Set parameters:

  Param;  % m-file with all parameters

  % Variables:

  Vt    =y(2);
  Vn    =y(3);
  Vb    =y(4);
  Theta =y(5);
  Fi    =y(6);

  % Equations:

  r(1)=-Vt+V(1)*cos(Theta)*cos(Fi)-V(2)*sin(Theta)*cos(Fi)-V(3)*sin(Fi);
  r(2)=-Vn-V(1)*cos(Theta)*sin(Fi)+V(2)*sin(Theta)*sin(Fi)-V(3)*cos(Fi);
  r(3)=-Vb+V(1)*sin(Theta)+V(2)*cos(Theta);
```

```
function [V]=BoatVel();
%
% function [V]=BoatVel;
%
% Defines boat velocity in each time step.
%
% Output:       V       Boat velocity in each time step.

  % Set parameters:

  Param;                % m-file with all parameters

  V(1,1:3)=9.5*[1 0 0]; % Initial velocty.

for t=delta_t:delta_t:500;
  V(1+t/delta_t,1:3)=9.5*[1 0 0];
end;

%  for t=delta_t:delta_t:30;
%   if t<440;
%      V(1+t/delta_t,1:3)=9.5*[cos(t/66.85)      sin(t/66.85)     0];
%    else
%      V(1+t/delta_t,1:3)=9.5*[cos(440/66.85)    sin(440/66.85)   0];
%    end;
%  end;
```

```
function [g]=CableEq(y,yprev,diffindex,delta,elem_n)
%
% function [g]=CableEq(y,yprev,diffindex,delta,elem_n);
%
% The six equations in each cable-element node. These equations
% should be satisfied (=0) in each time step.
%
% Input:        y               Current values of the 12 variables in y
%                               that is used in these six equations.
%               yprev           Values of y in prevoius time step.
%               diffindex       Index of which variable in y that should be
%                               differentiated. Used in Jacobi algorithm.
%               delta           Vector of deltas used to evaluate the
%                               partial derivatives of the six equations.
%                               Derivatives needed in the Jacobi algorithm.
%               elem_n          Cable element number (id) counted from
%                               boat, starting with 1 and ending with
%                               the total number of elements (n_elem).
%
% Output:       g               Current values of the six boat equations.

  % Set parameters:

  Param; % m-file with all parameters

  w=w(elem_n);                  % Set weight of this cable element;
  d=d(elem_n);                  % Set diameter of this cable element.
  delta_s=delta_s(elem_n);      % Set length of this cable element.
  Ct=Ct(elem_n);                % Set Ct of this cable element.
  Cn=Cn(elem_n);                % Set Cn of this cable element.

  % Equations of parameters
  A=pi*d*d/4;
  m=w/9.81+rho*A;
  m1=m+rho*A;

  dJndt=0;
  dJbdt=0;

  % Differentiate one of the variables:
  if diffindex > 0
    y(diffindex)=y(diffindex)+delta;
  end;

  % Approximation of s-derivatives:

  dTds     =(y(1)-y(7))/delta_s;
  dVtds    =(y(2)-y(8))/delta_s;
  dVnds    =(y(3)-y(9))/delta_s;
  dVbds    =(y(4)-y(10))/delta_s;
  dThetads =(y(5)-y(11))/delta_s;
  dFids    =(y(6)-y(12))/delta_s;

  % Approximation of t-derivatives:

  dTdt     =(y(7)-yprev(7))/delta_t;
  dVtdt    =(y(8)-yprev(8))/delta_t;
  dVndt    =(y(9)-yprev(9))/delta_t;
  dVbdt    =(y(10)-yprev(10))/delta_t;
  dThetadt =(y(11)-yprev(11))/delta_t;
  dFidt    =(y(12)-yprev(12))/delta_t;

  % Variables:
  T        =y(7);
  Vt       =y(8);
  Vn       =y(9);
  Vb       =y(10);
```

```
  Theta     =y(11);
  Fi        =y(12);

  % Equations:

  g(6)=-dFidt+((1/(1+e*T))*(dVnds+Vt*dFids-Vb*sin(Fi)*dThetads));
  g(5)=-dThetadt+((1/((1+e*T)*cos(Fi)))*(-dVbds+Vt*cos(Fi)*dThetads-Vn*sin(
Fi)*dThetads));
  g(4)=-dVbdt+((1/m1)*((m*Vt*cos(Fi)-(m1*Vn-rho*A*Jn)*sin(Fi))*dThetadt+rho
*A*dJbdt+(m1*Vb-rho*A*Jb)*e*dTdt/(1+e*T)-T*cos(Fi)*dThetads-(1/2)*rho*d*sqr
t(1+e*T)*Cn*(Vb-Jb)*sqrt((Vn-Jn)^2+(Vb-Jb)^2)));
  g(3)=-dVndt+((1/m1)*(-m*Vt*dFidt+rho*A*dJndt-(m1*Vb-rho*A*Jb)*sin(Fi)*dTh
etadt+(m1*Vn-rho*A*Jn)*e*dTdt/(1+e*T)+T*dFids-w*cos(Fi)-(1/2)*rho*d*sqrt(1+
e*T)*Cn*(Vn-Jn)*sqrt((Vn-Jn)^2+(Vb-Jb)^2)));
  g(2)=-dVtdt+((1/m)*((m1*Vn-rho*A*Jn)*dFidt-(m1*Vb-rho*A*Jb)*cos(Fi)*dThet
adt+m*Vt*e*dTdt/(1+e*T)+dTds-w*sin(Fi)-(1/2)*rho*d*sqrt(1+e*T)*pi*Ct*(Vt-Jt
)*abs(Vt-Jt)));
  g(1)=-dTdt+((1/e)*(dVtds-Vn*dFids+Vb*cos(Fi)*dThetads));
```

```
function [CV]=CableVel(y);
%
% function [CV]=CableVel(y);
%
% Evaluate velocity (CV) of cable-nodes in real world coordinate system,
% given the velocities (Vt Vn Vb) and angles (Theta Fi) of cable-nodes
% in cable coordinate system.
%
% Input:        y               Current values of variables in y.
%
% Output:       CV              Velocities (in 3 directions) of cable-
%                               nodes in real world coordinate system.

  cvp=1;                        % Pointer to CV.

  for i=1:6:size(y,1);

    Vt=y(i+1);
    Vn=y(i+2);
    Vb=y(i+3);
    Theta=y(i+4);
    Fi=y(i+5);

    LBKinv=[cos(Theta)*cos(Fi), -sin(Theta)*cos(Fi), -sin(Fi);
           -cos(Theta)*sin(Fi),  sin(Theta)*sin(Fi), -cos(Fi);
            sin(Theta),          cos(Theta),          0];

    CV(cvp:cvp+2)=[Vt Vn Vb]*LBKinv;
    cvp=cvp+3;
  end;
```

```
function [f]= Current(y,yprev,V);
%
% function [f]= Current(y,yprev,V);
%
% Evaluate the current values of all the equations,
% given the current values of the variables in y.
%
% Input:        y       Current values of variables in y.
%               yprev   Values of y in previous time step.
%               V       Velocity of boat.
%
% Output:       f       Current values of all equations.

  ysize=size(y,1);
  f(1:3)=BoatEq(y(1:6),0,0,V);
  i=4;
  elem=1;
  while i < (ysize-2)
    f(i:i+5)=CableEq(y((i-3):(i-3)+11),yprev((i-3):(i-3)+11),0,0,elem);
    i=i+6;
    elem=elem+1;
  end;
  f(i:i+2)=EndEq(y(ysize-11:ysize),0,0);
end;
```

```
function [e]=EndEq(y,diffindex,delta)
%
% function [e]=EndEq(y,diffindex,delta);
%
% The three equations at the boat end. These equations
% should be satisfied (=0) in each time step.
%
% Input:        y               Current values of last 12 variables in y.
%                               These variables is in the three end
%                               equations.
%               diffindex       Index of which variable in y that should be
%                               differentiated. Used in Jacobi algorithm.
%               delta           Vector of deltas used to evaluate the
%                               partial derivatives of the three equations.
%                               Derivatives are needed in the Jacobi
%                               algorithm.
%
%
% Output:       e               Current values of the three end equations.

  % Set parameters:

  Param;   % m-file with all parameters

  w=w(n_elem);                  % Set weight of this cable (end) element.
  d=d(n_elem);                  % Set diameter of this cable (end) element.
  delta_s=delta_s(n_elem);      % Set length of this cable (end) element.
  Ct=Ct(n_elem);                % Set Ct of this (end) cable element.

  if diffindex > 0
    y(diffindex)=y(diffindex)+delta;
  end;

  % Variables:
  T        =y(7);
  Vt       =y(8);
  Fi       =y(12);
  dThetads =(y(5)-y(11))/delta_s;
  dFids    =(y(6)-y(12))/delta_s;

  % Equations:
  e(1)=-T+(w*sin(Fi)+(1/2)*rho*d*Ct*pi*(Vt-Jt)*abs(Vt-Jt))*delta_s;
  e(2)=dThetads;
  e(3)=dFids;
```

```
function [y,delta_y]=Iter(y,yprev,V);
%
% function [y,delta_y]=Iter(y,yprev,V);
%
% Makes one iteration in the Jacobi algorithm. The Jacobi algorithm
% solves the nonlinear system f(y)=0 where f is a system of equations.
% The formula is J*delta_y=-f(y) where J is the Jacobian matrix.
% The variable delta_y is the adjusting of y in each iteration step.
%
% Input:        y       Values of variables in y (previous iteration).
%               yprev   Values of y in previous time step
%                       (previous iteration).
%               V       Velocity of boat.
%
% Output:       y       Values of variables in y after this iteration.
%               delta_y Adjusting of y in this iteration.

  J=Jacobian(y,yprev,V);                    % New equation L*U*delta_y=-f(y).
  [L,U]=lu(J);                              % J=L*U (U upper triangular).
  H=-(inv(L)*Current(y,yprev,V)');          % New equation U*delta_y=H.

  ysize=size(y,1);

  delta_y(ysize)=H(ysize)/U(ysize,ysize); % Backward substitution.
  for i=(ysize-1):-1:1;
    sum=0;
    for j=i+1:ysize;
      sum=sum+U(i,j)*delta_y(j);
    end;
    delta_y(i)=(H(i)-sum)/U(i,i);
  end;

  y=y+delta_y';
```

```
function [J]= Jacobian(y,yprev,V);
%
% function [J]= Jacobian(y,yprev,V);
%
% Evaluate the Jacobian matrix. This matrix is used to solve
% a non-linear system of equations. The partial derivatives in
% the Jacobian matrix is evaluated numericaly with delta-
% approximation.
%
% Input:        y       Current values of variables in y.
%               yprev   Values of y in prevoius time step.
%               V       Velocity of boat.
%
% Output:       J       Jacobian matrix.

  ysize=size(y,1);

  J=zeros(ysize);
  delta=[10;0.1;0.1;0.1;0.05;0.05;10;0.1;0.1;0.1;0.05;0.05];

  % Boundaries in boat (three first):

  [rcurrent]=BoatEq(y(1:6),0,0,V);

  for j=1:6;
    [r]       =BoatEq(y(1:6),j,delta(j),V);
    J(1:3,j)=(r-rcurrent)'./delta(j);
  end;

  % Cable equations:

  i=4;
  elem=1; % Cable element id.

  while i < (ysize-2);
    [gcurrent] = CableEq(y((i-3):(i-3)+11),yprev((i-3):(i-3)+11),0,0,elem);

    for j=(i-3):(i-3)+11;
      [g] = CableEq(y((i-3):(i-3)+11),yprev((i-3):(i-3)+11),j-(i-4),delta(j
-(i-4)),elem);
      J(i:i+5,j)=(g-gcurrent)'./delta(j-(i-4));
    end;
    i=i+6;
    elem=elem+1;
  end;

  % Boundaries in cable-end (three last)

  [ecurrent]=EndEq(y(ysize-11:ysize),0);

  for j=ysize-11:ysize;
    [e]=EndEq(y(ysize-11:ysize),j-(ysize-11)+1,delta(j-(ysize-11)+1));
    J(i:i+2,j)=(e-ecurrent)'./delta(j-(ysize-11)+1);
  end;
```

```
% Parameters used in m-files in directory:
%
% c:\matlab\mydoc\sixeq

  delta_t=10;      % Length of each time step.                    [s]
  n_elem=15;       % Number of cable elements.                    [-]

  e=0.0000001;     % Elastic coefficient.                         [-]
  rho=1000;        % Density of water.                            [kg/m^3]

  % Length of each cable element.                                 [m]

  delta_s(15,1)=30.5;
  delta_s(11:14,1)=ones(4,1)*68.6;
  delta_s(1:10,1)=ones(10,1)*72.3;

  % Weight/meter in water of each cable element.                   [N/m]

  w(15,1)=0.57;
  w(11:14,1)=zeros(4,1);
  w(1:10,1)=ones(10,1)*2.34;

  % Diameter of each cable element.                               [m]

  d(15,1)=0.0254;
  d(11:14,1)=ones(4,1)*0.0794;
  d(1:10,1)=ones(10,1)*0.0406;

  % Hydrodynamic resistance in t-direction of each cable element.[-]

  Ct(15,1)=0.02168;
  Ct(11:14,1)=ones(4,1)*0.00898;
  Ct(1:10,1)=ones(10,1)*0.0150;

  % Hydrodynamic resistance in n-direction of each cable element.[-]

  Cn(15,1)=1.8;
  Cn(11:14,1)=ones(4,1)*1.8;
  Cn(1:10,1)=ones(10,1)*2;

  Jt=0;            % Relative velocity of water stream.           [m/s]
  Jn=0;            %            "                                 "
  Jb=0;            %            "                                 "

  tol=10;          % Error tolerance in Jacobi-algorithm.         [-]

  % Note: When parameter is a vector (w and d), then the index of
  %       each cable element is counted from the boat, starting
  %       with 1 and ending with the total number of cable
  %       elements (=n_elem).
```

```
function PlotPos(posres);
%
% function=PlotPos(posres);
%
% Plot postions of cable-nodes.
%
% Input:        posres  Result of simulation. Positions of cablenodes
%                       in boat koordinate system in each timestep.

  % Set parameters:

  Param;                % m-file with all parameters.

  posplot=posres;       % Make a copy of the result.

  % Evaluate x-bias for each cablenode. X-bias is the distance
  % (along the cable) between two cable elements.

  xbias(1)=0;
  for j=4:3:(n_elem+1)*3;
    xbias(1+(j-1)/3)=xbias((j-1)/3)+delta_s((j-1)/3);
  end;

  axis([-1100 5000 -25 5]);

  for i=1:size(posplot,2);
    hold on;
    for j=4:3:(n_elem+1)*3;
      posplot(j,i)=posplot(j,i)-xbias(1+(j-1)/3);       % Subtract x-bias.
    end;
    plot(posplot(1:3:(n_elem+1)*3,i),-posplot(3:3:(n_elem+1)*3,i));
  end;
```

```
function Ploty(yres,tres);
%
% function Ploty(yres,tres);
%
% Plot time-variables T,Vt,Vn,Vb,Theta,Fi for all nodes:
%
% Input:        yres    Result of simulation. Values of variables in y.
%               tres    Result of simulation. Each time step.

yressize=size(yres,1);

for i=1:6;
  subplot(2,3,i);                                % Plot six diagrams.
  hold on;
  plot(tres,yres(i,:),'y');                      % Yellow is boat-node.
  for j=6:6:(yressize-6);
    plot(tres,yres(i+j,:),'r');                  % Red is middle nodes.
  end;
  plot(tres,yres(i+(yressize-6),:),'w');         % White is end-node.
end;
```

```
function [yres,posres,tres]=solve(Vtime);
%
% function [yres,posres,tres]=solve(Vtime);
%
% Solve the dynamic equations.
% Start with an initial guess of the variables in y.
% Old values (prevoius time step) is kept in yprev.
%
% Input:        Vtime           Velocity of boat in each timestep.
%                               Each row is [Vi Vj Vk].
%
% Output:       yres            Result of simulation. Values of variables
%                               in y in each timestep.
%               posres          Result of simulation. Values of positions
%                               of cablenodes (in real world koordinate
%                               system) in each timestep.
%               tres            Result of simulation. Each timestep.

  % Set parameters:

  Param;                                    % m-file with all parameters.
  y=yinit(6*(n_elem+1),0,Vtime(1,1));       % Start with straight cable
                                            % and critical angle zero.


  % Solve equations:

  pos=zeros(size(y,1)/2,1);                 % Initial position of cable-nodes.
  yprev=y;                                  % Assume steady state, i e yprev=y.

  stoptime=size(Vtime,1)*delta_t;           % Stoptime evaluated from indata V.

  for t=delta_t:delta_t:stoptime;           % Each loop is a time step.
    V=Vtime(t/delta_t,:);                   % Read actual boat velocity.

    [y delta_y]=iter(y,yprev,V);            % Iteration with Newtons method.
    iters=1;
    while max(abs(delta_y))>tol & iters<10;
      [y delta_y]=iter(y,yprev,V);
      iters=iters+1;
    end;                                    % Iteration done. Solution in y.

    yres(:,t/delta_t)=y;                    % Result saved in yres (column t).

    pos=pos+CableVel(y)'*delta_t;           % Integration to evaluate position
    posres(:,(t/delta_t)+1)=pos;            % of cable-nodes from known
                                            % velocities.


    tres(t/delta_t)=t;                      % Save each timestep.
    t

    yprev=y;                                % Save old values before
                                            % next timestep.
  end;
```

```
function [y]=yinit(ysize,fi_c,V1);

% function [y]=yinit(ysize,fi_c,V1);
%
% Evaluate an initial guess of the six variables in
% y=(T,Vt,Vn,Vb,Theta,Fi). Assume a straight cable
% with critical pitch angle Fi=fi_c,and course angle
% Theta=0 and Vb=0.This means that the boat has
% constant velocity V1 straight forward.
%
% Input:        ysize           Size of y.
%               fi_c            Critical angle of cable.
%               V1              Constant velocity of boat.
%
% Output:       y               An initial guess of y.

   % Set parameters:

   Param;  % m-file with all parameters.

   w=mean(w);                      % Approximative weight per cable element.
   d=mean(d);                      % Approximative diameter per cable element.
   delta_s=mean(delta_s);          % Approximative length per cable element.
   Ct=mean(Ct);                    % Approximative Ct per cable element.

   y=zeros(ysize,1);

   Vt=V1*cos(fi_c);
   Vn=V1*sin(fi_c);
   Fi=fi_c;
   dTds=w*sin(Fi)+(1/2)*rho*d*Ct*pi*(Vt-Jt)*abs(Vt-Jt);

   for n=1:6:ysize;
     y(n)=dTds*delta_s*((ysize-n+1)/6-1);
     y(n+1)=Vt;
     y(n+2)=Vn;
     y(n+5)=Fi;
   end;
```