2005:325 CIV

EXAMENSARBETE

Visualiseringsverktyg för rotordynamiska beräkningar

LINA JOHANSSON

CIVILINGENJÖRSPROGRAMMET
Maskinteknik

Luleå tekniska universitet
Institutionen för Tillämpad fysik • Maskin- och materialteknik
Avdelningen för Datorstödd maskinkonstruktion

LULEÅ TEKNISKA UNIVERSITET

2005:325 CIV • ISSN: 1402 - 1617 • ISRN: LTU - EX - - 05/325 - - SE

Visualiseringsverktyg för rotordynamiska beräkningar

Lina Johansson

22 december 2005

Förord

Detta arbete är utfört vid Avdelningen för datorstödd maskinkonstruktion, Luleå tekniska universitet, på uppdrag av SwedPower AB. Arbetet är utfört under perioden juni till december 2005. Handledare har varit Rolf Gustavsson, SwedPower AB och Peter Jeppsson, Avdelningen för datorstödd maskinkonstruktion vid Luleå tekniska universitet.

Jag vill tacka Rolf Gustavsson, SwedPower AB för möjligheten att få göra detta arbete och för all hjälp med att förstå vad en rotorsträng är. Även ett stort tack till Peter Jeppsson, handledare/examinator, för allt stöd, hjälp och framför allt inspiration.

Och Carro och Sawo, mina två rumskompisar jag haft under tiden för arbetet, utan er hade det aldrig ”gått”.

Tack!

Abstract

This work has been carried out at the Division of Computer Aided Design at Luleå University of Technology and has been an assignment from SwedPower AB. SwedPower is a consultancy in the energy and power sector and 100% owned by Vattenfall. The company carries out consulting assignments all over the world and it has about 500 consultants.

In this work a tool to visualize rotordynamic calculations has been developed in the CAE-tool I-DEAS. A model of a shaft and components has been built and a macro has been written to read the geometry of the shaft from a text file. The macro also makes changes of the shaft and visualizes the movement of the shaft as it is rotating. At each time step a picture is produced, these pictures is then put together into a movie.

The reason to use a CAE-tool is that the solid can be used not only for the visualization but also for analyses, for example FEA. In this case I-DEAS was chosen because it is easy to program, it is used by Vattenfall and I am already familiar with the software.

Sammanfattning

Detta examensarbete är utfört vid Avdelningen för datorstödd maskinkonstruktion, Luleå tekniska univeritet, på uppdrag av SwedPower AB. SwedPower AB är ett konsultföretag inom energisektorn och företaget är helägt av Vattenfall. Företaget finns representerat i stora delar av världen, med över 500 anställda inom energi och kraftsektorn.

I arbetet har ett visualiseringsverktyg för rotordynamiska beräkningar utvecklats i CAE-programmet I-DEAS. Ett makro har skrivits som läser in geometri och förskjutningar i en rotorsträng, ritar upp rotorsträngen och producerar bilder på förskjutningarna. Dessa bilder sätts sedan samman till en film som visar hur hela rotorsträngen rör sig.

Anledningen till att göra visualiseringen i ett CAE-verktyg är att samma modell i ett senare skede kan användas för analys, till exempel av spänningarna i axeln. I-DEAS valdes därför att det är ett program som redan finns på Vattenfall, det är lätt att programmera och har även använts i undervisningen på Luleå tekniska univeristet.

Innehåll

Kapitel 1

Inledning

1.1 Bakgrund

En rotorsträng i ett vattenkraftverk består av axel, matare, generator, turbin, lager och armkors. Figur 1.1 visar en enkel modell av en rotorsträng. När rotorsträngen roterar vid drift förskjuts axeln i xy-planet på grund av obalans från icke symmetrisk geometri, magnetisk dragningskraft och vattnets laster på turbinen. Hur deformationsplanen är definierade syns i figur 1.2(a).

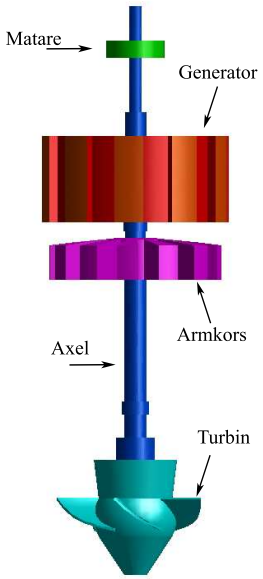
Axeln är uppdelad i ett antal element där början och slutet av elementet representeras av en nod. Bild 1.2(a) visar ett element i axeln och noderna symboliseras av koordinatsystemen.

Hur förskjutningarna ser ut i noderna kan beräknas med hjälp av beräkningsprogram. Med beräkningsprogrammen är det ofta svårt att visualisera förskjutningarna, beräkningsresultaten presenteras ofta i grafer eller tabeller. Genom att enbart titta i grafer och tabeller kan det vara svårt att få en bild av hur saker egentligen beter sig och därför behövs ett verktyg som på ett tydligare sätt visualiserar vad som händer i en rotorsträng.

1.1.1 Vattenkraft

Vattenkraften i Sverige står idag för ungefär hälften av den elproduktion som förekommer i landet. Tillsammans med kärnkraften är det den som ger Sverige grundproduktionen av elektricitet.

Vattenkraft är en förnyelsebar energikälla som aldrig tar slut utan förnyas genom det naturliga kretsloppet. Det är även en kraftkälla som är lätt att reglera, på några sekunder kan turbinens bladvinklar ändras eller ledskenor öppnas eller stängas beroende av behovet. Elektricitet måste produceras i samma ögonblick som det konsumeras. Vattenkraften används därför för att reglera toppar och dalar i energibehovet medan kärnkraften ligger på en jämn nivå över en längre



Figur 1.1: En generell rotorsträng.

tid.

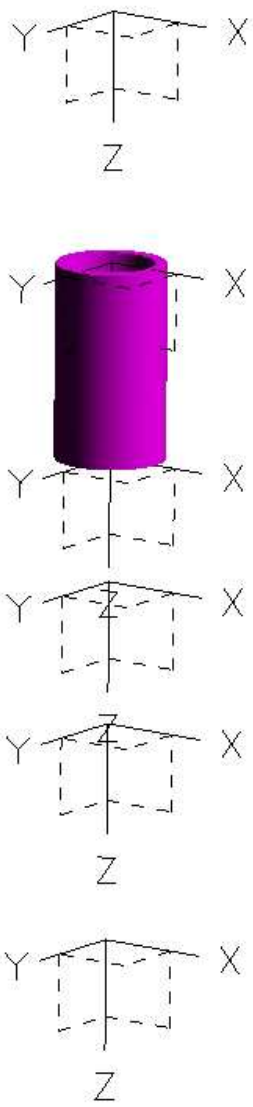
I vattenkraften utnyttjas skillnaden i lägesenergi för att framställa elenergi. Vattnet strömmar från en högre nivå till en lägre och på vägen passeras en turbin, turbinen sätter snurr på en generator som alstrar elektrisk energi. När fallhöjden är mellan ungefär 5-40 meter används oftast en Kaplanturbin, figur 1.3, och när fallhöjden är omkring 40 meter är det vanligare med Francisturbin, figur 1.3 [1].

1.1.2 SwedPower AB

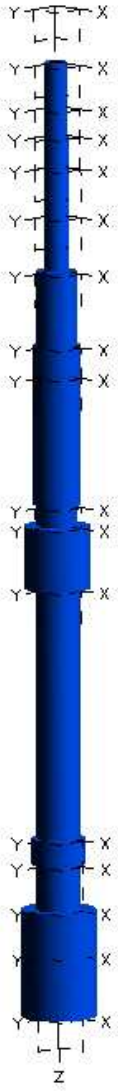
SwedPower är ett konsultföretag inom energisektorn där Vattenfall är huvudägare. SwedPower bildades 1976 som ett oberoende konsultbolag med syfte att möta den internationella utvecklingen inom energiområdet. En tredjedel av företagets försäljning sker utomlands och SwedPower finns representerade i bland annat Central- och Östeuropa, Asien och Afrika. Med över 500 anställda är de ett ledande företag inom energi och kraftsektorn [2].

1.2 Problembeskrivning

Uppgiften bestod i att utveckla ett visualiseringsverktyg för rotordynamiska beräkningar i CAE-verktyget I-DEAS. Indata ska ges från beräkningsprogram

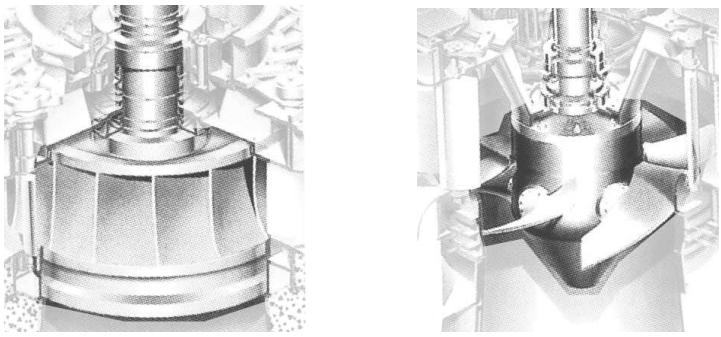


(a) Bilden visar ett godtyckligt element i axeln, noderna representeras av ett koordinatsystem.



(b) Här visas en gotycklig axel där noderna är markerade med koordinatsystem.

Figur 1.2: Figur 1.2(a) visar ett element i axeln, figur 1.2(b) visar en hel axel och dess noder som representaras av koordinatsystem.



(a) Francisturbin, används vi fallhöjder runt 40 meter.

(b) Kaplanturbin, används vid fallhöjder mellan ca 5 - 40 meter.

Figur 1.3: Två olika turbiner, Francis och Kaplan.

program för rotordynamik. Användaren av verktyget ska kunna välja att visa delar av rotorsystemet eller hela systemet med generator, turbin och lager. Användaren ska också ha möjligheten att välja olika konfigurationer och konstruktioner av generator och turbin.

Viss analys med CAD-programmets FEM-lösare ska kunna utföras, exempelvis spänningar i kopplingar mellan komponenter och spänningar i axlar.

1.3 Målgrupp

Arbetet riktar sig till de personer som arbetar med rotordynamiska beräkningar och som vill visualisera sina beräkningsresultat på ett enkelt sätt.

Kapitel 2

Solidmodellering

Visualiseringsverktyget har gjorts i CAE-programmet I-DEAS. Anledning till att använda ett CAE-program är att kunna få ut mer av modellen än bara visualiseringen, modellen kan enkelt användas för att beräkna volym, massa, spänningar med mera.

2.1 Val av program

Det finns många olika program att välja bland för att skapa visualiseringen av rotorsträngen. I det här fallet valdes I-DEAS.

I-DEAS är ett program som redan finns inom Vattenfall, det behöver alltså inte köpas in något program för att kunna använda verktyget. Fördelen med att göra visualiseringen i ett CAE-program är att solidmodellen även går att använda till att beräkna spänningar, massa med mera. I-DEAS är även det program som har använts i undervisning vid Luleå tekniska universitet.

2.2 I-DEAS

I-DEAS, Integrated Design Engineering Analysis Software är ett modernt CAE-program (Computer Aided Engineering) där solidmodellering, beräkning och simulering kan göras inom samma program. CAE-program är designade för att förenkla och förkorta en utvecklingsprocess, samma geometri kan användas till konstruktion, beräkning och tillverkning.

I-DEAS består av olika applikationer, Design, Simulation, Test och Manufacturing. Dessa applikationer har i sin tur olika underfunktioner. När dessa applikationer används tillsammans är I-DEAS som mest kraftfullt, det ger en kortare produktutvecklingsprocess [3].

Design är den applikation som använts i detta arbete och den beskrivs mer ingående i avsnitt 2.2.1 på sidan 6.

2.2.1 Design

I designapplikationen skapas solidmodellens geometri. Det är i designapplikationen allt arbetet börjar. Den modell som byggs upp här är den som senare används i de andra applikationerna för bland annat beräkning, tillverkning och simulering. Viss simulering kan utföra i designapplikationen men då endast simulering och analys av mekaniskt beteende som inte påverkas av yttre krafter [3].

2.2.2 Simulation

I simuleringsapplikationen kan krafter analyseras och det finns funktioner för att hantera data mellan stelkroppsanalys och finita elementberäkningar. Applikationen består av flera olika verktyg för att analysera modellen som byggts upp i designapplikationen. I verktyget för finita element kan till exempel spänningsanalys genomföras. Simuleringsapplikationen kan även hantera icke linjära dynamiska analyser, utmattningsanalyser och simulera värmeöverföring. Det finns även verktyg för att optimera konstuktionen [3].

2.2.3 Manufacturing

Tillverkningsapplikationen är ett verktyg som förenklar utvecklingsprocessen genom att man tidigt kan simulera och generera verktygsbanor för tillverkning och vilket produktionssätt som är det bästa. Den här applikationen använder också den ursprungliga geometrin skapad i designapplikationen [3].

2.2.4 Test

Testapplikationen innehåller olika testverktyg, till exempel för utmattning och modalanalys. Applikationen använder sig av grafisk representation vilket gör det enkelt att se resultaten. Med hjälp av testapplikationen minskar behovet av verkliga test och utvecklingsprocessen blir därmed billigare [3].

2.2.5 Modeler Task

Modeler Task är en undermeny som finns i de flesta applikationer, det är i den menyn själva geometrin skapas. Sketcher ritas upp som sedan till exempel extruderas eller roteras för att skapa en solid eller yta. Ändringar kan lätt göras på geometrin då historiken över uppbyggnaden hela tiden sparas.

Att historiken sparas över hur solidmodellen byggs upp är en förutsättning för att visualiseringsverktyget ska gå att göra i I-DEAS [3].

2.3 Programmering

I-DEAS är enkelt att använda med sitt grafiska gränssnitt, det finns enkla menyer och ikoner att klicka på. Alla menyer och ikoner har också ett motsvarande kortkommando som kan skrivas in via textprompten. Dessa kortkommandon är en förutsättning för att I-DEAS ska vara programmerbart.

Programmerbarheten i I-DEAS är ett kraftfullt verktyg som gör att användaren kan effektivisera arbetet genom att skapa makron/programfiler som till exempel automatiserar repetitiva uppgifter eller skapar specialapplikationer.

2.4 Programfiler

En programfil är en textfil som innehåller kommandon och knapptryckningar som kan utföras i I-DEAS. För att skapa en programfil krävs det att användaren känner till grundläggande programmering och har tillgång till enkelt textredigeringsprogram, helst inte en ordbehandlare som ofta lägger till dolda tecken.

Nackdelarna med programfiler är att de kan vara långsamma att köra och att de inte är garanterade att fungera i nyare versioner av I-DEAS [4].

2.4.1 Makro

Ett makro är en typ av programfil där användaren gör en inspelning av olika moment som utförs, till exempel en inspelning av en serie knapptryckningar. Makrot kan sedan spelas upp och utför då dessa inspelade kommandon.

Ett makro kan i många avseenden snabba upp och effektivisera sekvenser som återkommer regelbundet eller förenkla komplicerade moment. Att göra ett makro kräver ingen kunskap om programmering [5].

2.5 Programmeringsspråk

Det finns hundratals olika programmeringsspråk som är utvecklade runt om i världen. En stor anledning till att det finns så många programspråk beror på att olika människor uppskattar olika språk och att programmeringsspråken är bra på olika saker.

Programspråken är uppdelade i två olika programgrupper, interpreterande programspråk och kompilerande programspråk [6].

2.5.1 Interpreterande programspråk

Makrospråket i I-DEAS är ett interpreterande programspråk vilket betyder att programmet läses rad för rad och kommandona utförs därefter. Programmet

Tabell 2.1: För och nackdelar med olika programspråk

Programspråk	**Fördelar**	**Nackdelar**
Interpreterande	Lätt och snabb programmering, lätt att felsöka	Långsam att köra, kompilerar varje gång
Kompilerande	Snabbt att köra	Svårare att programmera än interpreterande språk, svårare att felsöka

kompileras under tiden det körs och även varje gång programmet körs. Ett interpreterande programspråk är ofta lätt att använda och lätt att felsöka[6].

2.5.2 Kompilerande programspråk

Ett kompilerande programspråk kompilerar koden när den är färdigskriven och sen behöver koden inte kompileras någon fler gång. Open API, Open Application Program Interface är en samling av mjukvaror för att skriva program till I-DEAS. Open API skrivs ofta i programspråket C++ som är ett kompilerande programspråk. Kompilerande programspråk är ofta svårare att lära sig skriva än interpreterande språk [7].

2.5.3 Jämförelse mellan interpreterande och kompilerande programspråk

Ett program som skrivs med kompilerande programspråk kan ta längre att utveckla än ett interpreterande program på grund av att programmeringsspråket är svårare och att programmet måste kompileras innan varje testkörning. När programmet är färdigt är det dock snabbare att köra än det interpreterande programmet som kompileras under körningen, se tabell 2.1 sidan 8.

2.5.4 Val av programspråk

I detta arbete har det interpreterande programspråket valts som arbetsspråk. Detta på grund av att det är enkelt att arbeta med, ingen större erfarenhet av programmering krävs och det behövs endast en texteditor för att skriva programmen.

2.6 Makrofilen

Visualiseringsverktyget är skrivet i I-DEAS makrospråk och filen består av sex olika huvuddelar. Variabeldeklaration, utplacering av noder, skapande av axeln, justering av geometri i sammanställningen, förskjutning av rotorsträngen och produktion av stillbilder.

2.6.1 Variabeldeklaration

I början av programmet deklareras de variabler som används i programmet. Här har användaren möjlighet att påverka antalet noder som ska användas, dimensioner på de delar som ingår i rotorsträngen, val av olika delar, val av katalog där bilderna ska sparas med mera.

2.6.2 Placering av noder

Axelns noder placeras ut efter dimensioner i en textfil som användaren anger. Noderna består av koordinatsystem som alla refererar från det lokala koordinatsystemet i "parten". Alla koordinatsystem placeras ut innan axeln börjar byggas upp, detta för att koordinatsystemen ska ligga efter varandra i historieträdet och därmed underlätta automatiseringen.

2.6.3 Axelns geometri

Axelns geometri byggs upp med en loop som läser in ytter- och innerdiameter från geometrifilen och sedan "loftas" geometrin fram mellan sektionerna som ligger i koordinatsystemens xy-plan.

2.6.4 Sammanställning av delar

Sammanställningen av delarna är redan gjord i modellfilen och det är inget som programfilen gör. Däremot positionerar programfilen de olika delarna i rätt nod på axeln efter i vilken nod användaren vill att de ska sitta. Här anpassas också bärlagrets dimensioner efter var på axeln användaren väljer att placera det. Val av turbin genomförs också, den turbin som inte används göms i sammanställningen.

I sammanställningen är armkorset låst i rymden. Axelns lokala koordinatsystem är låst till armkorset med en fri frihetsgrad, för att få rotataion runt axelns z-axeln. Resterande delar är fullständigt låsta till axelns lokala koordinatsystem och roterar med axeln i rörelsen runt z-led.

2.6.5 Deformation av rotorsträngen

Programfilen som skrivits ska förändra axelns utseende. Det som förändrar axelns utseende är en loop i programmet som läser in axelns deformation från en indatafil. Dessa indata förflyttar varje nod, det vill säga varje koordinatsystem som axlen består av, i förhållande till axelns lokala koordinatsystem. När varje nod fått en ny position uppdateras modellen av axeln och dess utseende förändras. Efter det roteras axelns runt sitt lokala koordinatsystems z-axel.

2.6.6 Produktion av stillbilder

För att få en film som visar rotorsträngens deformation och rotation produceras stillbilder som sedan sätts samman till en film. I varje tidssteg förändras axelns utseende och när modellen är uppdaterad tas en bild på hela strängen utseende. Bilden sparas och numreras, den bild som skapades först får lägst nummer.

Kapitel 3

CAD-modellen

Visualiseringen består av sju olika delar som tillsammans bygger upp en rotorsträng med axel, matare, generator, bärlager, armkors och turbin (Francis eller Kaplan). Dessa delar är parametriserade så att modellens utseende kan lätt ändras av användaren.

Alla sju delar är är modellerade och sammansatta i designapplikationen.

3.1 Axeln

Axeln är den del som överför vridmoment från turbinen till generatorn.

Axeln är uppbyggd av koordinatsystem som var och ett representerar en nod i rotorsträngen, se bild 1.2(b) sidan 3. Koordinatsystemen som axeln är uppbyggd av refererar alla till det lokala koordinatsystemet i axeln. Det lokala koordinatsystemet används sedan vid sammanställningen av rotorsträngen för att låsa axeln i förhållande till de andra delarna, det här koordinatsystemet är stilla i modellen och förskjuts inget. Resterande koordinatsystem kan förskjutas i x-, y- och z-led och roteras runt x-, y-, och z-axlarna. I varje nod definieras axelns geometri i form av ett tvärsnitt, så kallad sektion. Axelns geometri sveps sedan fram med hjälp av kommandot loft. Geometrin på axeln läses in från en textfil. För att loftningen ska fungera på ett bra sätt får dimensionsövergångarna inte var för snäva då problem med fria kanter kan uppstå. I detta fall får övergångarna inte vara mindre än 0,01 m. Med fria kanter menas att modellen har kanter som inte gränsar till någon annan kant, modellen är alltså en yta och ingen sammanslutande volym.

3.2 Matare

Mataren sitter i modellen högst upp på axeln. Det är mataren som bygger upp magnetfältet i generatorn, vilket gör att rörelseenergin från generatorn kan om-

vandlas till elektrisk energi i statorn.

Mataren är uppbyggd på samma sätt som axeln. Det lokala koordinatsystemet i mataren är låst till axeln och förskjuts inget i förhållande till axelns lokala koordinatsystem. Utifrån det lokala koordinatsystemet i mataren är ett nytt koordinatsystem skapat som matarens geometri är uppbyggd runt. Det här koordinatsystemet förskjuts och roteras runt sina axlar och soliden som skapats följer med i rörelsen.

Mataren är fullständigt parametriserad och innan visualiseringsverktyget startas får diameter och höjd på mataren och i vilken nod den ska sitta anges i programfilen.

3.3 Generator

Generatorn producerar rörelseenergi och med hjälp av matarens magnetfält omvandlas rörelseenergi till elektrisk energi.

Generatorn är uppbyggd på samma sätt som axeln och mataren. Det lokala koordinatsystemet i generatorn är låst till axeln och förskjuts inget i förhållande till axelns lokala koordinatsystem. Utifrån det lokala koordinatsystemet i generatorn är ett nytt koordinatsystem skapat som generatorns geometri är uppbyggd runt. Det här koordinatsystemet förskjuts och roteras runt sina axlar och soliden som skapats följer med i rörelsen.

Generatorn är fullständigt parametriserad och innan visualiseringsverktyget startas får diameter och höjd på generatorn anges i programfilen. Även antal poler och i vilken nod generatorn ska sitta anges.

3.4 Bärlager

Bärlagrets uppgift är att bära upp tyngden av de roterande delarna, det vill säga turbin, axel, generator och matare. När vattner passerar genom turbinen utsätts lagret för ytterligare beslastning genom att tryckfallet över turbinen ger en kraft som trycker ner turbinen.

Bärlagret är parametriserat att anpassa sig efter var på axeln det placeras och vilken dimension som väljs på armkorset.

3.5 Armkors

Armkorsets uppgift är att hålla bärlagret på plats och därmed bära upp lasten från rotorsträngen och vattnet.

Ett armkors är modellerat och innan visualiseringsverktyget startas anges diameter på armkorset och i vilken nod det ska sitta. Armkorset styr dimensioner och position på bärlagret.

3.6 Turbin

Turbinen är det som överför vattnets rörelse till en mekanisk rotationsrörelse av axeln. Beroende på fallhjöd och vattenmängd används olika typer av turbiner. Vid låga och medelhöga fallhjöder, ca 5 - 40 meter används Kaplanturbinen som även är den vanligaste turbintypen i moderna svenska kraftsationer. Francisturbinen används i kraftstationer med större fallhöjd, vid ungefär 40 meter och uppåt. Bild på Kaplan- och Francisturbin syns i figur 1.3.

Två turbiner är modellerade, en Francis- och en Kaplanturbin. Båda är fullständigt parametriserade, diameter och höjd på turbinen, även antal turbinblad anges i programfilen innan programmet startas. Turbinen placeras automatiskt i den nedersta noden på axeln.

Kapitel 4

Manual

Visualiseringsverktyget består av flera olika delar. Modellfilen som innehåller solidmodellen, programkoden som läser in geometri och förskjutningar till modellfilen och indatafiler som innehåller värden på geometri och förskjutningar. Även andra program än I-DEAS har använts, Ideastoppm och Imtools för att konvertera bilderna till lämpligt format som passar mpeg-encodern och en mpeg-encoder för att skapa filmen från stillbilderna.

4.1 Modellfilen

Modellfilen från I-DEAS behövs för att köra visualiseringsverktyget. I modellfilen finns alla information om de olika delarnas geometri och hur de är skapade. De olika delarna i rotorsträngen är fördefinierade och användaren av verktyget bestämmer de olika delarnas dimensioner i programfilen innan den startas.

4.2 Koden

Programkoden finns i bilaga A. Längst upp i programkoden kan olika inställningar göras. Här definieras hur många noder axeln ska bestå av, i vilka noder generator och dylikt ska sitta och även geometri för generator, turbin och de andra delarna bestäms här. Axelns geometri läses dock in från en textfil.

4.3 Textfilerna

För att textfilerna ska vara läsliga av I-DEAS måste varje kolumn vara avgränsad med ett kommatecken. Antalet kolumner i textfilen måste också stämma överens med antalet kolumner definierade i programkoden. Stämmer inte detta fås programfel vid körning.

4.4 Programvaror

För att producera filmen som visualiserar rotorsträngen behövs ett antal programvaror. I detta arbete har I-DEAS, Imtools, mpeg-encode och Ideastopmm använts.

4.4.1 I-DEAS

I-DEAS har använts för att producera bilder till visualiseringen. Utförligare beskrivning av I-DEAS finns i kapitel 2.2 på sida 5.

4.4.2 Ideastoppm

Bilderna som produceras i I-DEAS sparas i formatet .pff. Ideastoppm är ett script som konverterar bilderna från .pff till formatet .ppm. Formatet .ppm stöds av Imtools som används för att konvetera bilderna ytterligare en gång till format som passar mpeg-encodern.

4.4.3 Imtools

Imtools är ett gratisprogram och har använts för att konvertera bildfilerna från .ppm till .jpg. Jpeg-bilderna kan sedan användas i mpeg-encodern för att tillverka en film [8].

4.4.4 mpeg-encode

MSC.Marc är ett finita element-program. Med MSC.Marc följer en mpeg-encoder som sätter ihop jpeg-bilder till film. Denna encoder har använts för att få en film av bilderna som producerats i I-DEAS.

4.4.5 Script

För att arbetet utanför I-DEAS ska vara enkelt, gå snabbt och automatiskt, har några shellscript skrivits. Ett av scripten gör om textfilerna med indata till att passa I-DEAS, kolumnerna med indata ändras från att vara semikolonavgränsade till avgränsas med komma, se bilaga B.1. Det andra scriptet konverterar bilderna med hjälp av Ideastoppm och Imtools, se bilaga B.2.

Kapitel 5

Slutsats

5.1 I-DEAS begränsningar

I I-DEAS finns det vissa begränsningar för den här typen av applikationer. Det tar lång tid att producera bilderna, det är svårt att hantera små dimensionsövergångar och det är svårt att automatisera meshningen i modeller där topologin varierar.

5.1.1 Tid att producera bilder

Att producera bilderna i I-DEAS tar lång tid, varje bild behöver ca fem minuter för att bli färdig. Det som händer under dessa fem minuter är att modellen läser in nya förskjutningar, uppdaterar och tar en bild. Ju färre noder modellen består av desto snabbare går det att producera bilderna.

5.1.2 Dimensioner

För att loftfunktionen som använts för att svetpa fram axelns geometri ska fungera på ett tillfredställande sätt är det en fördel att använda så långa element som möjligt, minst 0.1 m. Med långa element fås en finare böjning av axeln när den deformeras. När axelns element loftas fram skapas även tangentvillkor som gör att deformationen i ett element påverkar intilliggande element. Funktionen med tangetvillkor är dåligt utvecklad i I-DEAS och fungerar därför inte på ett tillfredställande sätt när elementlängden blir korta. Det resulterar i ”kantig” övergång mellan elementen.

Dimensionsövergångarna mellan de olika axeldiametrarna får inte vara för små. I-DEAS kan då få problem med fria kanter när modellen uppdateras efter att nya förskjutningar lästs in. Med fria kanter menas att modellen har kanter som inte gränsar till någon annan kant, modellen är alltså en yta och ingen sammanslutande volym.

5.1.3 FEM-analys

I programmet staplas cylindrar med olika dimensioner ovanpå varandra. När två cylindrar med lika dimensioner staplas på varandra smälts gemensamma kanter samman och två kanter ersätts med en kant. Detta gör att numreringen av kanterna blir olika från fall till fall, det finns ingen möjlighet att förutse från modell till modell hur kanterna numreras. För att kunna automatisera meshningen i detta fall krävs det att man vet vad kanterna har för nummer. Eftersom detta är svårt att beräkna och hantera bestämdes att någon spänningsanalys inte görs i detta examensarbete.

5.2 Fortsatt arbete

För att få en visalisering med bättre grafik kan ett CAD-program med bättre visualiseringsmöjligheter användas, till exempel NX3. I NX3 går det även att namnge kanterna på solidmodellen vilket gör att meshning på ett enklare sätt går att automatisera.

För att generera filmen har en mpeg1-encoder använts, encodern komprimerar filmen ganska mycket vilket gör att filen blir mindre och att grafiken försämras. En bättre mpeg-encoder kan användas för att få en film med högre kvalitet.

I framtiden kan man tänka sig att verktyget kan användas för att se hur en rotorsträng i drift rör sig. Mätvärden skickas direkt från kraftstationen och rotorsträngens förskjutningar visas i realtid direkt på skärmen hos användaren av programmet.

Litteraturförteckning

[1] Beckeman & Wenner AB (2001). *Vattenkraft* Utgåva 7. www.svenskenergiiskolan.nu

[2] SwedPower AB. 21 september 2005 www.swedpower.se

[3] Mark H. Lawry (2004). *I-DEAS Student Guide* 2:a uppl. New York, NY 10020 : McGraw-Hill. ISBN 0-07-252544-4

[4] I-DEAS help library (2004). 20 oktober 2005 http://www.mt.luth.se/software/ideas11/SDRCHelp /LANG/English/int_ug/program_files.htm#gGDTudoek

[5] Jeppsson, Peter, lektor vid Avdelningen för datorstödd maskinkonstruktion, Luleå tekniska universitet, Luleå. Föreläsning 6, MTM153 höstterminen 2004.

[6] Rejås, Marcus och Määttä, Magnus. *Programmering i PHP* 18 oktober 2005 http://www.rejas.se/fritis/programmeringab/c60.html

[7] I-DEAS help library (2004). 4 oktober 2005 http://www.mt.luth.se/software/ideas11/SDRCHelp /LANG/English/oar_ug/api_over.htm

[8] http://www-vis.lbl.gov/NERSC/Software/imtools/

Bilaga A

Programkod

Den här bilagan visar den programkod som skrivits i I-DEAS för att producera stillbilderna.

```
C : Rotor.prg
C : Program som deformeras en rotorsträng och
    producerar stillbilder
C :
C : Examensarbete - Visualiseringsverktyg för
    rotordynamiska beräkningar
C : Lina Johansson, Luleå tekniska universitet, 2005
C :
K : $ return
K : $ mpos :; /O N
K : $ return
C : # ON_ERROR PAUSE
K : # echo none
C :
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : x                                        x
C : x    1) Välj i vilka noder prylarna ska  x
C : x       sitta.                           x
C : x    2) Välj totala antalet noder.       x
C : x    3) Välj antal tidssteg              x
C : x    4) Välj turbin och dimensioner på   x
C : x       prylarna i [m].                  x
C : x                                        x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C :
C : Deklarering av variabler
C : xxx Nod för armkorset xxxxxxxxxxxxxxxxxxx
```

```
C :                                                x
K : # Armk = 10
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Nod för generatorn xxxxxxxxxxxxxxxxxxxx
C :                                                x
K : # Gen = 7
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Nod för mataren xxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : # Matare = 4
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Antalnoder xxxxxxxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : # AntalKoord = 16
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Antal tissteg xxxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : # steg = 200
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Val av turbin xxxxxxxxxxxxxxxxxxxxxxxx
C :     1 = Kaplan                                 x
C :     2 = Francis                                x
C :                                                x
K : #        val = 1
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Antal Francis-blad xxxxxxxxxxxxxxxxxx
C :                                                x
K : #   FranBlad = 15
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Antal Kaplan-blad xxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   KaplBlad =  6
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Matardiameter xxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   MatDia   = 2
```

```
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Matarhöjd xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   MatHojd = 0.5
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Generatordiameter xxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   GenDia   = 6.0
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Generatorhöjd xxxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   GenHojd = 1.5
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Antal poler, jämnt nr. xxxxxxxxxxxxxxx
C :                                                x
K : #   Poler    = 24
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Armkorsdiameter xxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   ArmDia   = 5.0
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Turbindiameter xxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   FranDia   = 1.5
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Turbinhöjd xxxxxxxxxxxxxxxxxxxxxxxxxx
C :                                                x
K : #   KapHojd = 3
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C : xxx Val av katalog där bilder sparas xxxxx
C :                                                x
K : #   DIR = "/scratch/Lina/"
C :                                                x
C : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
C :
K : #     Turbin =  15
```

```
K : #      AvMat = 0.9
K : #      AvGen = 0.8
K : #      AvArm = 0.8
K : #      AvTur = 0.8
K : #   koordNum = 1
K : #          j = koordNum
K : #   Distance = 1
K : #          m = 1
K : #          k = 3
K : #        EXT = ".pff"
K : #        IDX = 1
K : #   COMMAND2 = "mv bild.pff "
K : #   variabel = 1
C :
C :
C :
C : --- Städar undan och plockar fram axeln ---
K : / ma pu lab all ge et sr Axeln; d Appl okay
C : --- Ändrar till SI-enheter och byter till Master Modeler---
K : / o u u si
K : / ta mm
C : ---Skapar det koordinatsystem som noderna refererar ifrån---
K : / cr ref cs lab Axeln; cs cs2; tri tra 0,0,Distance rot 0,0,0
C :
C :
C : ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
C : o                                                                 o
C : o    Placerar ut koordinatsystemen                                o
C : o                                                                 o
C : ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
C :
K : # OPEN hp "geometri.dat"
K : # LOOP_KOORD:
K :     # READ hp FORMAT="(1F20,5F20.4)" Ax Ay Az Kx Ky Kz
K :     # Distance = (Distance + Kx)
K :       / cr ref cs lab Axeln; cs cs2; tri tra 0,0,Distance
            rot 0,0,0
K :     # IF ( (Ax) eq Matare ) THEN # AvMat = Distance
K :     # IF ( (Ax) eq Gen ) THEN # AvGen = Distance
K :     # IF ( (Ax) eq Armk ) THEN # AvArm = Distance
K :     # IF ( (Ax+1) eq (AntalKoord) ) THEN # AvTur = Distance
K :     # j = (j + 1)
K :     # IF ( j le (AntalKoord-1) ) THEN GOTO loop_koord
K : # CLOSE hp
```

```
C :
C : oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
C : o                                                                    o
C : o    Loftar fram axeln                                               o
C : o                                                                    o
C : oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
C :
K : # jo = 1
K : # OPEN hop "geometri.dat"
K : # LOOP_CIRKEL:
K :     # Influ = 0.05
K :     # READ hop FORMAT="(1F20,5F20.4)" Gx Gy Gz Vx Vy Vz
K :     # if ( Vx ge 1.0 ) THEN # Influ = 0.5
K :     # if ( Vx le 0.3 ) THEN # Influ = 0.005
K :     # if ( Vx le 0.2 ) THEN # Influ = 0.001
C :
K :     / w at lab Axeln; cs "cs"+k; xy
K :     / cr c ce op cx; 0; cy; 0; d d rd Gz; okay
K :               op cx; 0; cy; 0; d d rd Gy; okay
K :
K :     / w at lab Axeln; cs "cs"+(k+1); xy
K :     / cr c ce op cx; 0; cy; 0; d d rd Gz; okay
K :               op cx; 0; cy; 0; d d rd Gy; okay
K :
K :     / cr lo lab Axeln; c "c"+m c "c"+(m+1) don lab "c"+(m+4)
K :
K :
K :         lab Axeln; c "c"+(m+2) c "c"+(m+3) don lab "c"+(m+6)
K :
K :
K :
K : fe a
C : ***--- Skippar tangentvillkor om elemetlängden
C :        är kortare än 0.05 meter
K :     # if ( Vx le 0.05 ) THEN GOTO UTAN_TAN
C :
C : ***--- Placerar ut tangentvillkor
K : mo
K : ta
K : lab
K : Axeln
K : "ch"+jo
K : t
K : u
```

```
K : i
K : key
K : Influ
K : okay
K : lab
K : Axeln
K : "ch"+(jo+2)
K : t
K : u
K : i
K : key
K : Influ
K : okay
K :
K : okay
K : okay
K : # GOTO ROST
C :
K :     # UTAN_TAN:
K : okay
K : # ROST:
K :     # jo = (jo + 4)
K :     # k = (k + 1)
K :     # m = (m + 8)
K :     # IF ( k le (AntalKoord+1) ) THEN GOTO loop_cirkel
K : # CLOSE hop
C :
C : ***--- Byter till Master Assembly
K : / ta ma
C :
C : ***--- Städar och plockar fram assemblyt ---***
K : / ma pu all
K :
K : / ma ge et sd sr Rotorstrang; d appl okay
C :
C : ***--- Positionerar delarna längs axeln
K : / cn ae v 1; d eq AvArm; okay
K : / cn ae v 2; d eq AvMat; okay
K : / cn ae v 3; d eq AvGen; okay
K : / cn ae v 4; d eq AvTur; okay
K : / cn ae v 5; d eq AvTur; okay
C :
C : ***--- Dimensionsändringar på matare, generator
           och armkors genomförs
```

```
K : / cn eq lab Rotorstrang; in hi8 v 11; d eq MatDia; okay
K : / cn eq lab Rotorstrang; in hi8 v 8; d eq MatHojd; okay
K : / cn eq lab Rotorstrang; in hi14 v 7; d eq GenDia; okay
K : / cn eq lab Rotorstrang; in hi14 v 9; d eq GenHojd; okay
K : / cn eq lab Rotorstrang; in hi14 v 19; d eq Poler; okay
K : / cn eq lab Rotorstrang; in hi11 v 1; d eq (ArmDia/2); okay
K : / cn eq lab Rotorstrang; in hi15 v 40; d eq KapHojd; okay
K : / cn eq lab Rotorstrang; in hi18 v 7; d eq FranDia; okay
C :
C : ***--- Val av turbin genomförs
K : # GOTO "Sak_"+val
K : # Sak_1:
K : / a h ph sr Francis; d hi okay !
K : / cn eq lab Rotorstrang; in hi15 v 34; d eq KaplBlad; okay
K : / up p
K : #    GOTO abort
C :
K : # Sak_2:
K : / a h ph sr Kaplan; d hi okay !
K : / cn eq lab Rotorstrang; in hi18 v 23; d eq FranBlad; okay
K : / up p
K : #    GOTO abort
K : # ABORT:
C :
K : / up c lab q se sr Rotorstrang; d okay
C : ***--- Inställningar för bilden. Vy, genomskinlighet mm.
K : / V E 0,1,0,0 AU !
K : / DO SO DT SHH Okay ! do df aa of appl paf aoo okay appl
      asf aoo okay appl okay !
K : / mo ap lab Rotorstrang; > in Hi8 don v tr 90 okay don !
K : / mo ap lab Rotorstrang; > in Hi14 don v tr 90 okay don !
K : / mo ap lab Rotorstrang; > in Hi10 don v tr 90 okay don !
K : / mo ap lab Rotorstrang; > in Hi11 don v tr 90 okay don !
K : / mo ap lab Rotorstrang; in q se sr Axeln; d okay don cs on
      co ? 2; appl okay don
K : / do li e 1 d cu e 2 d cu !
C :
C : ***--- Visar Minors i historieträdet
K : / cr spe ht lab Rotorstrang; in q se sr Axeln; d okay don
K :   mi canc
C :
P :
C : ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
C : o                                                                 o
```

```
C : o    Bilder produceras                                          o
C : o                                                               o
C : oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
C :
C : ***--- Loop som tar bilder och numrerar bildfilerna
C :
K : # OPEN oops "respons.dat"
K : # LOOP_BILD:
K :      # koosys = 3
K :      # IMG_FILE=DIR+"tid_"+IDX+EXT
K :      # IF (IDX LT 1000) THEN #IMG_FILE=DIR+"tid_0"+IDX+EXT
K :      # IF (IDX LT  100) THEN #IMG_FILE=DIR+"tid_00"+IDX+EXT
K :      # IF (IDX LT   10) THEN #IMG_FILE=DIR+"tid_000"+IDX+EXT
C :
K :      / do lh dt li appl; okay
C :
K :      # LOOP_VRIDA:
K :           # READ oops FORMAT="(1F20.4,1F20,5F20.4,)"
                                  Tids Noden Nx Ny Rx Ry Rz
K :           # IF ( Noden eq Matare ) THEN GOTO F_Matare
K :           # IF ( Noden eq Gen ) THEN GOTO F_Gen
K :           # IF ( Noden eq AntalKoord ) THEN GOTO F_Tur
K :           # GOTO F_Axel
C :
C :***--- Deformerar mataren
K :        # F_MATARE:
K :          / cr spe ht lab Rotorstrang; cs  Hi8 cs4; tri don
K :          ha 4; md dv v 1; d eq (Nx*50); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs  Hi8 cs4; tri don
K :          ha 4; md dv v 2; d eq (Ny*50); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs  Hi8 cs4; tri don
K :          ha 4; md dv v 4; d eq (Rx*(180/3.14)*100); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs  Hi8 cs4; tri don
K :          ha 4; md dv v 5; d eq (Ry*(180/3.14)*100); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :        # GOTO F_Axel
C :
```

```
C : ***--- Deformerar generatorn
K :        # F_GEN:
K :          / cr spe ht lab Rotorstrang; cs hi14; cs4; tri don
K :          ha 4; md dv v 1; d eq (Nx*50); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs hi14; cs4; tri don
K :          ha 4; md dv v 2; d eq (Ny*50); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs hi14; cs4; tri don
K :          ha 4; md dv v 4; d eq (Rx*(180/3.14)*100); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs hi14; cs4; tri don
K :          ha 4; md dv v 5; d eq (Ry*(180/3.14)*100); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :        # GOTO F_Axel
C :
C : ***--- Deformerar turbinen
K :        # F_TUR:
K :          / cr spe ht lab Rotorstrang; cs hi15; cs4; tri don
K :          ha 4; md dv v 1; d eq (Nx*50); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs hi15; cs4; tri don
K :          ha 4; md dv v 2; d eq (Ny*50); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs hi15; cs4; tri don
K :          ha 4; md dv v 4; d eq (Rx*(180/3.14)*100); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; cs hi15; cs4; tri don
K :          ha 4; md dv v 5; d eq (Ry*(180/3.14)*100); appl okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :        # GOTO F_Axel
C :
K :        # F_AXEL:
K :          / cr spe ht lab Rotorstrang; in q se sr Axeln;
               d okay don
K :          ha koosys; md dv v 1; d eq (Nx*50); appl; okay
```

```
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
K :          / cr spe ht lab Rotorstrang; in q se sr Axeln;
               d okay don
K :          ha koosys; md dv v 2; d eq (Ny*50); appl; okay
K :          des
K :          / up c lab q se sr Rotorstrang; d okay
C :
K :      # koosys = (koosys + 1)
K :      # if (koosys le (AntalKoord+2)) THEN #GOTO loop_vrida
C :
C :
K :      / do df as on appl okay
K :      / mo ap lab Rotorstrang; in q se sr Axeln; d okay don
K :        sh s ra 40; di 85; gl 60; br 40; id 1;
K :        ss on rs on ds on gs on bs on is on v ts on tv1 on
           appl okay don
K :      / do so dt sr dq f cs bes Appl Okay
K :      / up c lab q se sr Rotorstrang; d okay
K :      / or r lab Rotorstrang; in hi12; don lab Rotorstrang;
           cs hi12; cs2; ori z 8.5; don
C :
C :      ***--- Tar en bild och tilldelar filnamn ---***
K :      / f pi op f f okay; fil bild okay; y
K :      / v asr
K :      # COMMAND=COMMAND2+IMG_FILE
K :      # wait 1
K :      # execute COMMAND
K :      # wait 1
K :      # IDX=IDX+1
K :      # DELETE Tids, Noden, Nx, Ny, Rx, Ry, Rz
K :      / do lh dt li appl; okay
K :      # IF (idx LE steg) THEN #GOTO loop_bild
C :
K : # CLOSE oops
C :
K : $ mpos :; /O N
K : # DELETE ALL
K : $ return
C : / f e n
K : $ mpos :; /F PR E
E : **** END OF SESSION ****
```

Bilaga B

Script

B.1 Ändring av textfilerna

Script för att ändra textfilerna till en form som I-DEAS kan läsa.

```
#!/bin/tcsh
#
# Byter ut ; mot ,
#
dos2unix < respons.skv > respons.ux.dat
tr \; \, < respons.ux.dat > respons.dat
```

B.2 Bilder till film

Script för att konvertera bilderna från .pff till .jpg och för att göra en film av bilderna. Scriptet använder sig av Ideastoppm, Imtools och do_in för att konvertera bilder och skapa filmen.

```
#!/bin/tcsh
#
#Konveterar bilder och sätter ihop
#till film
#
module add imtools30
foreach i ( *.pff )
./ideastoppm $i | imconv -outquality 100 -informat ppm
  -infile - -outfile $i.jpg
end

foreach i (*.pff.jpg)
mv $i ‘basename $i .pff.jpg‘.jpg
```

```
end

module add marc2003
/home/ilioja-1/priv/Script/do_in
```