Referat

Ett flervariabelt feldetekteringssystem för övervakning av bärlagertemperaturen i vattenkraftturbiner

Henrik Fredlund

Syftet med examensarbetet var att utveckla ett automatiskt feldetekteringssystem för övervakning av bärlagertemperaturen i vattenkraftturbiner. De ingående parametrarna förutom bärlagertemperaturen var kylvattentemperaturen och kylvattenflödet. En enkel statisk modell baserad på data samplat en gång per minut togs fram för att estimera bärlagertemperaturen. Därefter utvecklades en detektor för att upptäcka avvikelser i bärlagertemperaturen baserad på CUSUM-algoritmen. På grund av en för liten mängd data var den framtagna modellen alltför osäker för att kunna implementeras i ett fungerande system.

Det framtagna feldetekteringssystemet visade sig fungera bra för de data som fanns tillgängliga. Det är däremot rekommenderat att utvärdera systemets prestanda med längre dataserier. En ytterligare modell baserad på minutdata över ett år måste tas fram innan systemet kan fungera på riktigt. De resultat som erhölls var:

- Feldetekteringssystemet klarar av att upptäcka abrupta och långsamma avvikelser av bärlagertemperaturen.
- Inga falsklarm ges då det är enstaka mätfel eller givarfel av sådan typ som tagits upp i arbetet. Pågår ett mätfel alltför länge ges dock ett larm.

Feldetekteringsalgoritmen implementerades även i Delphi för att kunna användas i ett fungerande system över Internet där t.ex. trendkurvor och larmsignaler skall kunna presenteras.

Nyckelord: Feldetektering, bärlager, Conwide, systemidentifiering, black-box, grey-box, fysikalisk modellering, larm

Uppsala universitet, Institutionen för informationsteknologi, Box 337, 751 05 Uppsala

ISSN 1401-5765

Abstract

A multivariable fault detection system for surveillance of bearing temperature in hydropower turbines.

Henrik Fredlund

The purpose of this thesis work was to develop an automatic fault detection system for surveillance of bearing temperature in hydropower turbines. The parameters used except the bearing temperature were cooling water temperature and cooling water flow. A simple static model based on data sampled every minute was developed to estimate the bearing temperature. Then a detector for detection of change in bearing temperature based on the CUSUM-algorithm was designed. Since the amount of data was very small the developed model was too uncertain to be used in a working system.

The designed fault detection system showed to work well for the available data. It is, however, recommended that the performance of the system should be evaluated using more data. Another model based on data sampled once every minute for at least a year has to be developed before the system can be fully evaluated. The results shown were:

- The fault detection system can discover fast and slow changes in bearing temperature.
- No false alarms were given for measuring faults and sensor faults of the types used in this thesis. If a measuring fault occurs for too long there will be an alarm.

The fault detection algorithm was also implemented in Delphi to be used in a working system over the Internet where for example trends and alarms will be presented.

Keyword: Faultdetection, bearing, Conwide, system identification, black-box, grey-box, physical modelling, alarm

Uppsala University, Department of Information Technology, Box 337,
SE-751 05 Uppsala, Sweden

ISSN 1401-5765

Förord

Ett stort gemensamt tack till de personer som varit med och hjälpt mig i mitt arbete. Några vill jag dessutom tacka speciellt. Jag vill tacka min handledare Magnus Eriksson på Vattenfall Utveckling som har stöttat mig och hjälpt mig med det mesta rörande praktiska detaljer och programmering i *Delphi*, Christian Bernstone på Vattenfall Utveckling för stöd och synpunkter och till sist min ämnesgranskare Bengt Carlsson, institutionen för systemteknik vid Uppsala universitet, för all hjälp och stöd jag fått från planering och utförande till färdig rapport.

Ett stort tack även till personalen på Älvkarleby kraftverk som ställt upp på frågor och visat mig runt i Söderfors. Ett speciellt tack även till Dag Lindroth som installerat de givare i Söderfors som saknades.

UPTEC W 04 004, ISSN 1401-5765
Tryckt hos Institutionen för geovetenskaper, Uppsala universitet, Uppsala 2004

Innehållsförteckning

Sida

Appendixförteckning

1 Introduktion

1.1 Inledning

Detta examensarbete bygger på två tidigare examensarbeten ”Utveckling av larm för tekniska system i vattenkraftstationer” utfört av Minna Glemme vid Vattenfall Utveckling 2001 samt ”Ett feldetekteringssystem för bärlager i en vattenkraftstation” utfört av Mattias Sjölund vid Vattenfall Utveckling 2002 (för sammanfattningar se appendix 1 och 2). Minna Glemme påvisade att vid konstanta kylvattenflöden är det endast kylvattentemperaturen som bärlagertemperaturen är beroende av samt att långtidsdata som är sparade på dygnsnivå ej är representativa för användande vid modellframtagning. Mattias Sjölund använde sig av minutdata insamlade under ett år. Ansatsen att använda en rekursiv algoritm gjorde att långsamma förändringar i bärlagertemperaturen är svårare att upptäcka.

I det här examensarbetet ingår ytterligare en variabel nämligen kylvattenflödet. Den varierar beroende på kylvattentemperaturen och påverkar bärlagertemperaturen. Istället för att använda en rekursiv algoritm kan en enkel linjär modell upptäcka både abrupta och långsamma förändringar i bärlagertemperaturen. För att få en bättre anpassad modell kan en fysikalisk modell användas som har ett större giltighetsområde och större tillförlitlighet än en enkel linjär modell.

1.2 Syfte och mål

Syftet med examensarbetet är att ta fram ett feldetekteringssystem som kan identifiera abrupta och långsamma fel i bärlagertemperaturen i kraftstationer med reglerat kylvattenflöde. Den undersökta kraftstationen är Söderfors kraftstation. Målet är att det framtagna systemet även skall kunna användas på andra kraftverk med samma uppbyggnad som Söderfors utan några större förändringar.

1.3 Begränsningar

När examensarbetet inleddes visade det sig att givare saknades i Söderfors kraftverk på grund av förseningar av installationsarbetet. Det inledande arbetet gick därför ut på att se till att givare installerades så snart som möjligt och att insamlingen av data kom igång. Tyvärr drog detta ut på tiden och datainsamling kom igång först efter 13 veckor. Detta gjorde att examensarbetet blev mer inriktat på teorin och de praktiska resultaten får ses som preliminära.

2 Bakgrund

2.1 Vattenkraft

Vattenkraftverken utnyttjar älvarnas fallhöjd och vattenflöde för att alstra energi. Det är alltså vattnets lägesenergi mellan två nivåer som utnyttjas. Vattnet som strömmar från en högre till en lägre nivå passerar en turbin och får turbinaxeln att rotera (fig. 1). Turbinen driver en generator som omvandlar turbinens roterande rörelse till elektrisk energi (Vattenfall, 2003). Transformatorn anpassar spänningen till en nivå som är lämplig för ledningsnätet. För att öka fallhöjden och för att kunna lagra vatten byggs dammar. Dammarna skapar stora vattenmagasin som gör det möjligt att anpassa elproduktionen efter säsong och förbrukning. Elförbrukningen varierar över dygnet. Det går åt mer el på dagen än på natten och mer el på vardagar än på helgdagar. I Sverige används också mer el på vintern än på sommaren. Hur mycket el som produceras i vattenkraftverken beror också på vädret. Har det snöat och regnat mycket är produktionen högre än år då det är ont om vatten i sjöar och älvar.

Vattenkraften är och kommer även i framtiden att vara en viktig energikälla. Fördelarna är många. Den är dels billig, uthållig och ger nästan ingen påverkan på miljön vid drift. Däremot blir den lokala miljön påverkad när det byggs kraftverk, dammar och regleringsmagasin.

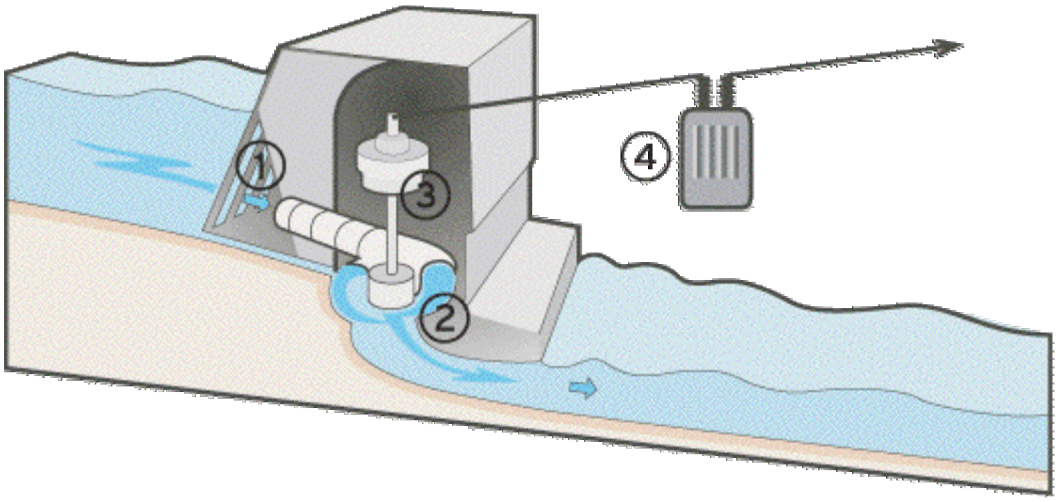


Fig. 1 Ett vattenkraftverk i genomskärning, 1. Vattenmagasin, 2. Turbin, 3. Generator, 4. Transformator (Vattenfall, 2003).

2.2 Om Vattenfall AB

Norge, Island och Sverige är de största användarna av vattenkraft i världen. I Sverige finns det ungefär 1200 vattenkraftverk, där de hundra största vattenkraftverken tillsammans producerar nästan hälften av den vattenkraftsel som används i landet (Vattenfall, 2003). Vattenfall är en av de fem största elproducenterna i Europa och producerar och levererar ungefär hälften av den el som används i Sverige. Vattenfall arbetar idag med nästan alla tillgängliga energikällor, vattenkraft, kärnkraft och vindkraft samt fossila bränslen, biobränslen och avfall till el- och värmeproduktion. Vattenfall driver tre kärnkraftverk med tillsammans åtta reaktorer och ca hundra vattenkraftverk varav 28 har en effekt över 100 MW. Vattenfall driver också 40 vindkraftverk och är i och med det den största vindelproducenten i Sverige. Vattenfallkoncernen består av moderbolaget Vattenfall AB samt ett nittiotal direkt eller indirekt ägda rörelsedrivande dotterbolag. Vattenfall AB är helägt av staten. Koncernens totala produktionskapacitet år 2002 var 32 000 MW el och 13 000 MW värme.

Fakta och siffror om Vattenfall

- Vattenfall producerar och distribuerar elektricitet och värme till ca 6 miljoner kunder i norra Europa. De största elkunderna är industrier och energiföretag.
- Vattenfall säljer energitjänster och energilösningar.
- 33 900 anställda vid årsskiftet 2002/2003.
- Nettoomsättning (miljoner SEK) 101 025.
- Investeringar (miljoner SEK) 39 932.
- Elproduktion cirka 160 TWh/år, varav vattenkraft 64 TWh/år.
- Elförsäljning cirka 180 TWh/år.
- Värmeproduktion och försäljning 34 TWh/år.

Siffrorna är hämtade från Vattenfalls årsredovisning år 2002.

2.3 Söderfors kraftverk

Kraftverket i Söderfors är en så kallad pilotanläggning där det är tänkt att ny teknik skall prövas och utvärderas inom projektet som kallas UHC (UnderHållsCentral) Dalälven och drivs av Vattenfall. I det projektet ingår även detta examensarbete.

Söderfors kraftverk ligger vid Dalälven ca 30 km sydväst om Älvkarleby. Det färdigställdes och togs i drift 1978-79. Det består av två stycken aggregat med en effekt på 10 MW vardera. Bruttofallhöjden uppgår till 4,5 meter och den normala vattenföringen är 790 m^3/s. Den årliga energiproduktionen uppgår till ca 100 GWh/år.

2.3.1 Kaplanturbiner

Aggregaten i Söderfors är av typen Kaplanturbiner med horisontella axlar och med ett varvtal på 62,5 rpm. Eftersom turbinerna är liggande är bärlagren annorlunda jämfört med vertikala turbiner där bärlagret sitter längst ned och bär upp hela generatoraxeln. Här sitter istället bärlagret centralt i Kaplanturbinen (fig. 2 eller appendix 3). Det finns

även två styrlager, ett främre och ett bakre som förhindrar rörelse i sidled. Lagren är oljesmorda glidlager. Det är viktigt att turbinens rotation sker så friktionsfritt som möjligt varvid uttagbar effekt kan maximeras och att lagren slits så lite som möjligt.

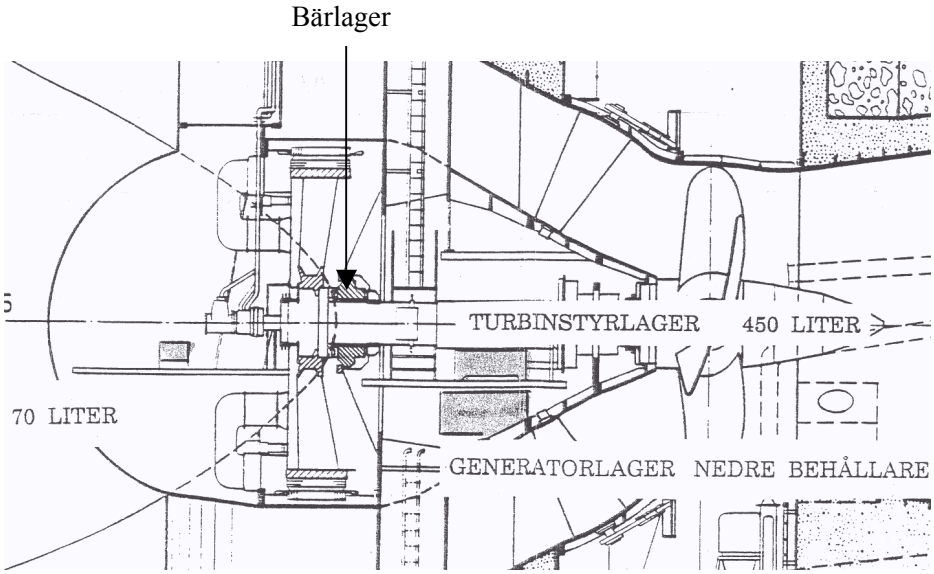


Fig. 2 Söderfors kraftverk i genomskärning.

2.3.2 Bärlager

Bärlagren är glidlager med en skyddande oljefilm som byggs upp med centrifugalkraft då axeln roterar. Oljan tillförs genom att den pumpas till en behållare, där den genom egentyngd transporteras via rör till lagret. Behållaren hålls alltid fylld, då den ska kunna tillföra olja även vid strömavbrott. Vid start och stopp, då varvtalet är mycket lägre än det normala, räcker inte rotationen till för att bygga upp oljefilmen. Då byggs oljefilmen upp med hjälp av tryckluft som tvingar olja in i lagret (Ljung, 2003).

2.3.3 Kylning

Kylningen av bärlagret sker genom att älvvatten leds in och kyler lageroljan på vägen upp till behållaren (fig. 3). Flödet på kylvattnet regleras beroende på temperaturen på oljan (fig. 4). Kylvattenflödet mäts med en flödesgivare innan regleringen. På grund av att oljan som används är en syntetolja (VL 46) så skall oljetemperaturen hållas på 23ºC, annars kan oljefilmen bli för tunn, så att rotationen inte sker tillräckligt friktionsfritt. Oljetemperaturen får endast höjas några grader, men kan sänkas något. Kylvattentemperaturen mäts på kylvattenledningen strax efter flödesgivaren med en PT 100-givare (fig. 4). Kylvattnet kyler lageroljan som i sin tur kyler bärlagret. Bärlagertemperaturen mäts segmentvis med både en PT 100-givare och en vanlig kvicksilvertermometer som kan avläsas i maskinhallen (Ljung, 2003). Temperaturstegring i ett lager kan vara en indikation på ökad friktion vilket är ett allvarligt fel i en turbins bär- och styrlager. Den ökade temperaturen orsakar i sin tur en

materialexpansion i lagrets massiva delar vilket påskyndar förloppet som får en exponentiellt växande hastighet. Därför är det viktigt att ha en effektiv kylning av lagren.

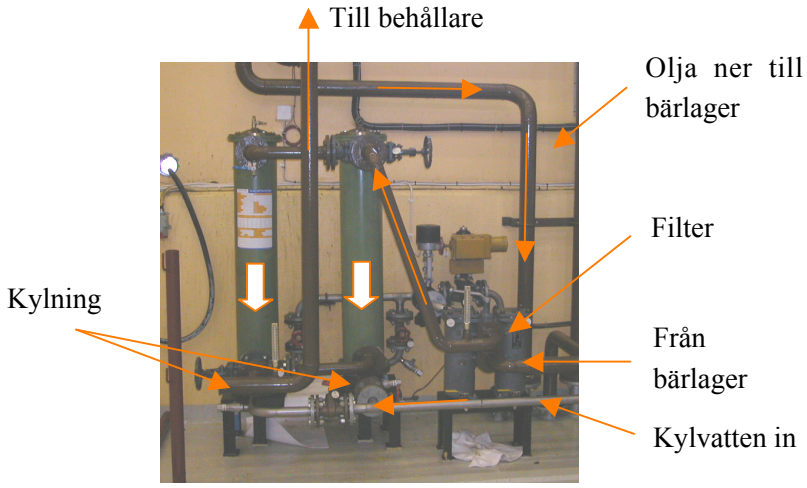


Fig. 3 Kylning av bärlageroljan.

Fig. 4 Temperaturgivare (PT 100), flödesgivare (Promag 50) och flödesregulator i Söderfors.

2.4 Tillståndskontrollsystemet Conwide

Conwide III är ett datasystem för tillståndskontroll av vattenkraft skapat och vidareutvecklat av Conwide AB i samarbete med Vattenfall AB (Kallin, 2003). Systemet är framtaget för att underlätta insamling och behandling av data från mätningar, inspektioner (rondningar) och besiktningar av utrustning. Det första rondsystemet, PC-TK, togs fram 1988 på Vattenfalls initiativ och bekostnad. År 1990 övertog Conwide AB helt utvecklingen av systemet och skapade Conwide II, vilket var DOS-baserat. Den senaste versionen Conwide III är Windows-baserad (Glemme, 2001).

Målet med Conwidesystemet är att anläggningens tekniska status skall samlas i ett system som skall vara utvecklingsbart, lätt att underhålla och endast kräva en minimal

mängd programvara i användarens persondator. Programvaran skall också kunna köras på enkla servrar. Den information som lagras i systemet skall kunna användas som beslutsunderlag vid uppföljning av arbeten och underhåll. Systemet skall också förenkla rapportering av fel och åtgärder.

Conwidesystemet består oftast av en så kallad standararbetsplats (fig. 5) det vill säga den standarddator som används inom Vattenfallkoncernen (i fortsättningen kallad SA-dator), handdator, koncentrator och kopplingsplint som är uppkopplad mot befintlig instrumentering inom anläggningen och särskilda givare. SA-datorn är en dator av PC-typ försedd med skrivare och systemprogram för sammanställning och utvärdering av driftdata. Koncentratorn utgör det som brukar kallas datalogger och är försedd med en larmskrivare. Figur 6 visar en principskiss av ett Conwidesystem med koncentrator.

Fig. 5 SA-dator och koncentrator i Söderfors.

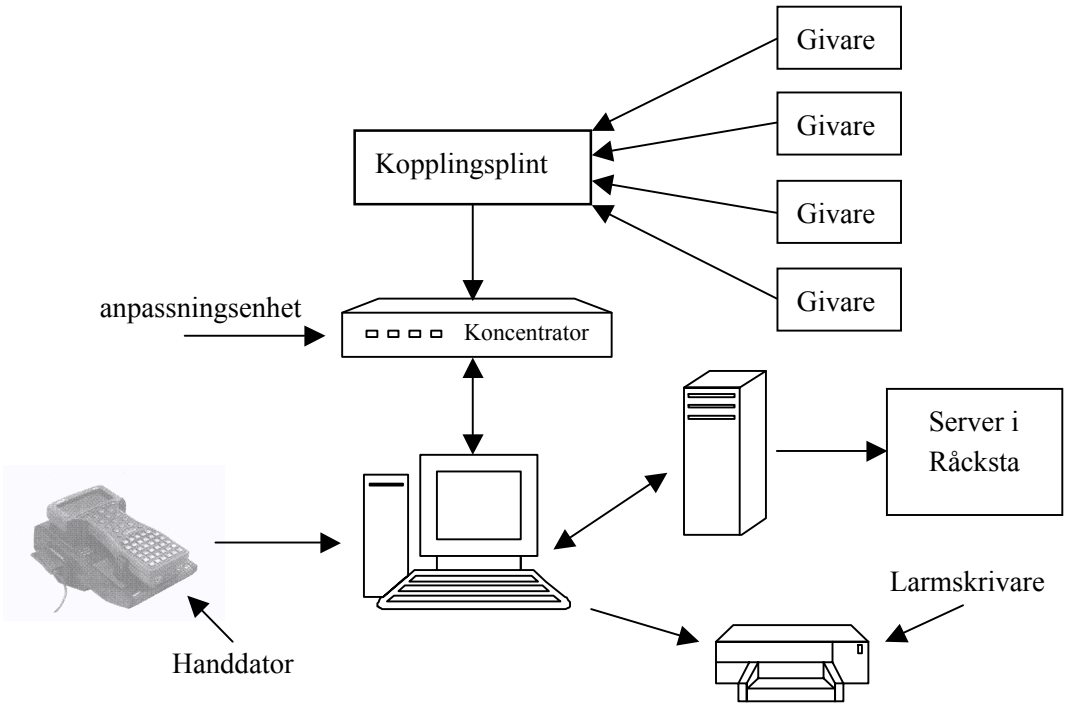


Fig. 6 Principskiss för tillståndskontrollsystemet Conwide.

2.4.1 Koncentrator

Koncentratorn (fig. 5) är i första hand tänkt för de anläggningar som saknar andra styr- och övervakningsdatorer (Kallin, 2003). Finns däremot moderna kontrollutrustningar kan Conwide även kopplas mot och hämta värden från dessa med en så kallad anpassningsenhet. Koncentratorn kan också kopplas mot en kontrollutrustning som till exempel styr utskovsluckor. Med koncentratorns hjälp kan, för viss utrustning, antalet starter räknas eller drifttiden mätas. Detta görs med hjälp av koncentratorns digitala ingångar. Koncentratorn kan också sampla analoga mätvärden på temperatur, tryck, vibration, effekt, vinklar med mera, högst en gång per sekund. Om ingen förändring av en viss parameters mätvärden sker är det maximala intervallet mellan mätvärdena en minut. Sekundmätvärdena finns lagrade i koncentratorn under minst fyra veckor, om intervallet mellan mätvärdena är större räcker emellertid det allokerade minnet för att lagra mätvärden för en tidsperiod upp mot ett år**.** Ur lagrade värden tas ett representativt dygnsvärde ut och överförs till en central databas lokaliserad i Råcksta. Vad som anses vara det representativa dygnsvärdet väljs för varje enskild analog signal i databasen och kan vara till exempel maximum-, minimum- eller medelvärdet. Koncentratorn kan även överföra värden för givna ”tidsögonblick” kopplade mot annan signal. Standardmässigt sätts den valda signalen som huvudsignal och det högsta värdet överförs som dygnsvärde. I en databas i Råcksta lagras dygnsvärden från samtliga av Vattenfalls stationer som har Conwide III-systemet. I slutet av november år 2002 var 20 av Vattenfalls 54 storskaliga vattenkraftverk försedda med koncentratorfunktionen.

2.4.2 Användningsområden

Conwide används av driftpersonalen i huvudsak som ett stöd vid tillsynsarbeten i kraftverken (Kallin, 2003). Till sitt förfogande vid tillsynsarbeten, eller rondning som det allmänt kallas, har operatörerna en handdator. Bestämda rondningslistor används för att underlätta strukturerade observationer. Listorna skapas av driftpersonalen vid respektive anläggning och bygger på respektive stations underhållsplan varför de skiljer sig mellan olika stationer.

Inför rondning överför driftpersonalen aktuell rondlista till handdatorn från Conwidesystemet. Under rondningen matas mätdata och andra observationer enligt rondprotokollet in i handdatorn. Om ett visst mätvärde överskrids larmar handdatorn och nytt värde kan matas in. Genom larmet undviks inmatning av orimliga värden. Kommentarer kan läsas och läggas till i rondningsprotokollet. Efter avslutat tillsynsarbete överförs observationerna till stationsdatorn vilket möjliggör analys av dessa.

De huvudkomponenter som används är förutom handdatorn händelsehanteraren, driftjournalen och trendanalys. Händelsehanteraren är stommen i programmet, där det mesta rörande det analytiska arbetet av driftdata kan utföras. Larmgränser för enskilda sensorer i rondlistan kan ändras, resultat och trender kan visas, driftjournalposter kan skapas etc.

Driftjournalen är en så kallad ”loggbok” vars syfte är att dokumentera onormala händelser och hur dessa hanteras samt fungera som en kunskapsåterförening mellan teknikerna. Det går också att använda den inbyggda statistikmodulen i syfte att skapa underlag för att kunna bedöma antalet störningar och deras omfattning samt kostnader för olika system och apparater i en anläggning. Journalen kan integreras med rondsystemet. Driftpersonalen kan läsa, ändra och radera den information som finns vid behov. En inbyggd sökfunktion i systemet som kan söka på systemnummer, apparat, specifika ord i en fritext, specifika ord i en rubrik eller underrubrik möjliggör att driftpersonalen kan hitta en händelse. Driftjournalen skall också kunna läsas av bland annat extern personal vid felsökning i anläggningen.

Vid införande av en ny händelse skall följande fyra (obligatoriska) poster fyllas i:

- Datum
- Tid
- Station (kan tas från organisationstabell, t.ex. PK006)
- Signatur (på den som utför arbetet)

Utöver dessa bör följande anges:

- Aggregat (t.ex. 03)
- Systemnummer
- Arbetstyp
- Driftläge
- Apparat

Systemnummer anges vanligen och skall motsvara den så kallade VAST-dokumentationen. Varje del i stationen har ett systemnummer på tre siffror där den första kan vara tomt, byggnad, turbin m.m. Den andra siffran kan motsvara någon del i en tomt, byggnad och så vidare. och den tredje en specifik utrustning. Bromsutrustning för en Kaplanturbin kan till exempel motsvara 432 i detta system. En sökfunktion finns för att underlätta valet av nummer. Vid en fritextsökning som kan göras på ordkombinationen ”pump, olja” kan det emellertid visas att ett visst fel ibland är noterat på flera olika systemnummer. Det kan eventuellt bero på att ett visst fel först kan se ut som ett annat fel till exempel elektriskt fel och därför noteras som ett sådant men sedan visa sig vara av en annan typ, som mekaniskt fel.

”Apparat” skall också följa ett visst system men anges inte lika ofta. Om systemnumret är 432 som i exemplet ovan, betecknar B belägg till denna bromsutrustning. Beteckningen P står för pump. Arbetstyp kan betecknas med UTR för utryckning. Kod för driftläge kan vara ID.

”Visa trend” åstadkommer trendkurvor vilka bygger på historiska data. Trendkurvorna kan skapas i SA-datorn eller på PC som är ansluten till den centrala databasen i Råcksta. Trendkurvor som skapas i SA-datorn har högre upplösning än de som kan skapas ur databasen i Råcksta. Anledningen till att den högre upplösningen erhålls är att de

bygger på sekund- eller minutvärden istället för dygnsrepresentativa värden som finns i den centrala databasen.

Trendningen kan innehålla en eller upp till maximalt fyra komponenters data och valfri tidsperiod. Med flera komponenters data underlättas jämförelsearbetet. Ett problem vid en trendlinje är att bildskärmens kapacitet inte alltid räcker för att kunna nyttja alla mätdata. Kurvan interpoleras då mellan mätvärdena varvid intressanta händelser ibland därför inte framträder. Det finns dock en zoom-funktion som kan användas för att zooma in intressanta tidsintervall.

2.4.3 Systemuppbyggnad för Conwide III

Det windowsbaserade systemet Conwide III är skrivet för Windows NT 32 bit med fullt nätverksstöd. Conwide III är kompatibelt med programvaran Word och har möjlighet att exportera data till databaser som till exempel Excel. Systemet vidareutvecklas fortfarande och används vid fler än 180 anläggningar, inklusive anläggningar som inte ägs av Vattenfall Vattenkraft AB.

Överföringen till databasen sker normalt genom att koncentratorn en gång per dygn kopplar upp sig och överför data. Det är emellertid inte möjligt att koppla upp sig mot koncentratorn utifrån om denna inte är uppkopplad. Detta på grund av ökad säkerhet mot dataintrång. En fast IP-anslutning ökar i omfattning och 2003 är det 6 anläggningar som har fast anslutning.

Koncentratorn kan bestyckas utifrån kundens (anläggningens) behov. Den kan utrustas med 32, 64, 96 eller 128 analoga ingångar och 0, 32 eller 64 digitala ingångar plus en extra virtuell räknare för varje digital ingång. De analoga och digitala ingångarna tål överspänningar. De digitala ingångarna delas upp i två kategorier, räknare och drifttidräknare. Räknare används för att räkna antal starter, men starter kortare än en sekund filtreras bort. Drifttid räknas per dygn och nollställs varje dygn.

Definitionen av ingångar, larmgränser och kalibrering är helt mjukvarustyrd vilket innebär att programvara används istället för byglar, motstånd etc.

Handdatorn som används av operatörerna vid rondning är en Husky FS/2 vilken är av PC-typ och kan utrustas med en så kallad AD-mättillsats. Med mättillsatsen kan till exempel batterimätningar och mätningar med vibrations- och tryckgivare göras.

2.5 Larmhantering

2.5.1 Beskrivning av TK-larm

TK-larm (Tillståndskontroll larm) i till exempel en vattenkraftanläggning kan sammanfattas som en åtgärd baserad på ett beslut om en oacceptabel tillståndsförändring. Undantag av TK-larm klassas som falsklarm och beror bland annat på hur larmgränser är inställda och vilken larmhanteringsstrategi som ligger till grund för inställda kriterier samt villkor på den övervakade enheten. En avvikelse är ett samlingsbegrepp som står för en tillståndsförändring från det ”normala” i processen. Förändringen behöver inte betyda att något fel inträffat utan det kan vara ett så kallat ”potentiellt” fel, se P i figur 7. Beroende på felutvecklingstiden tar det olika lång tid tills det potentiella felet övergått till ett feltillstånd. Alla larm genererade i TK-systemet Conwide kallas för TK-larm. Syftet med TK-larm är att upptäcka potentiella fel i anläggningen. Larm från TK-systemet används endast lokalt och personalen informeras normalt under den ordinarie arbetstiden. Hur larmen tolkas och vilka åtgärder som behöver vidtas kan variera beroende på flera olika kriterier som exempelvis den tillgänglighet eller den säkerhet som krävs i den aktuella driftsituationen (Sjödin, 2003).

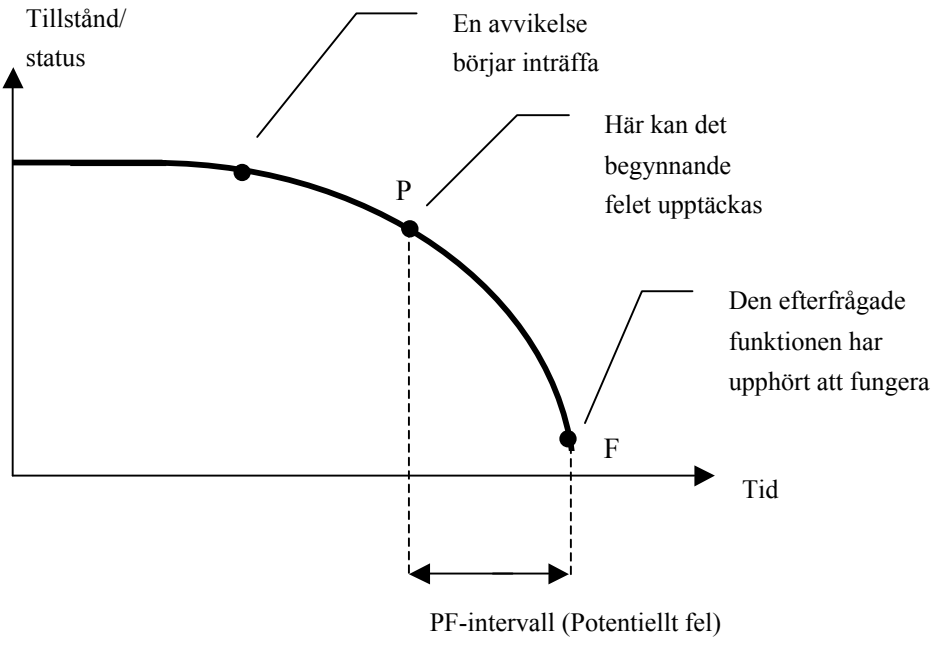


Fig. 7 Felutveckling i tiden.

2.5.2 Larmhantering i en kraftstation

För larmhanteringen vid en vattenkraftstation finns det två olika typfall (Glemme, 2001).

Det första och enklaste fallet är då driftpersonal befinner sig på stationen. Larmet är inställt så att en ljudsignal hörs då ett fel inträffar. Samtidigt fås en utskrift på signalföljdskrivaren som visar vad det är som larmar. Någon ur arbetsgruppen som befinner sig lämpligt till för tillfället kvitterar då larmsignalen så att den blir tyst och sedan läses det av vad som orsakat larmet på signalföljdskrivaren. Om larmet är av övergående natur behöver inget göras, en kommentar om vad som hänt sparas i händelseskrivaren. Om något behöver åtgärdas kan det ofta göras på en gång. Beroende på vilket fel som inträffat ringer driftpersonalen ibland upp driftcentralen och hör vad de anser innan felet åtgärdas. I vissa fall tas aggregatet ur drift. Personalen på kraftverket ska dokumentera alla fel och vilka åtgärder som eventuellt vidtagits i driftjournalen, men detta görs inte alltid.

Det andra fallet är då stationen är obemannad. Detta gäller nätter och helger samt då hela personalen befinner sig ute på andra kraftstationer. Vid dessa tillfällen är larmet inställt på fjärrsignal och ingen signal hörs på den aktuella kraftstationen. I Söderfors fall innebär det att larmet går i Älvkarlebys kontrollrum. Den jourhavande personalen kontaktas och åker ut till den larmande kraftstationen för att kvittera larmet och se vad som orsakat det. Efter kvittering är arbetsgången densamma som i det första fallet.

3 TEORI

3.1 Inledning

Feldetektering spelar en allt större roll i processindustrin. Dynamisk modellering av system eller komponenter är en väl utvecklad del av processindustrin och modellbaserade metoder kan därför vara effektiva och relativt enkla att tillämpa. I vattenkraftproduktionen är processen mindre ”stabil” eftersom produktionen varierar efter elbehovet. Detta gör att aggregaten inte går i ”konstant läge”. När ett feldetekteringssystem utvecklas delas förfarandet upp i olika delar. Framtagandet består framförallt av följande tre steg.

1. Modellkonstruktion av det aktuella systemet.
2. Implementering av feldetekteringsalgortitm.
3. Hantering av felen, det vill säga larmhantering.

Hur varje del behandlas beror på vad som är av intresse för det aktuella problemet. I detta kapitel presenteras bakomliggande teori till de använda metoderna.

3.2 MODELLKONSTRUKTION

3.2.1 Introduktion

En modell är ett verktyg för att kunna besvara frågor om ett system. Modeller används för att bland annat göra prognoser, konstruktioner, regulatordesign, forskning för att utveckla och testa hypoteser, feldiagnos och processövervakning (Sjölund, 2002). Det finns flera olika modelltyper, till exempel mentala modeller, verbala modeller, fysiska modeller och matematiska modeller. För att konstruera en matematisk modell finns det två grundprinciper, det är fysikaliskt modellbygge och systemidentifiering, se figur 8 (Glad & Ljung, 1991).

Fysikaliskt modellbygge (white-box identifiering) innebär att återföra systemets egenskaper på delsystem vilkas uppförande är kända. För tekniska system innebär detta i regel att de ”naturlagar” (massbalans, energibalans, Newtons lagar etc.) som beskriver delsystemen används. För icke-tekniska system (ekonomiska, biologiska etc.) finns i regel inga säkra ”naturlagar” ens för enkla delsystem. Då används istället hypoteser eller allmänt vedertagna samband (Glad & Ljung, 1991). Fördelarna med fysikalisk modellering är att den ofta har stort giltighetsområde, ger fysikalisk insyn och modeller där parametrarna har fysikalisk mening. Det negativa är att det ofta är svårt att finna de fysikaliska sambanden som ligger bakom.

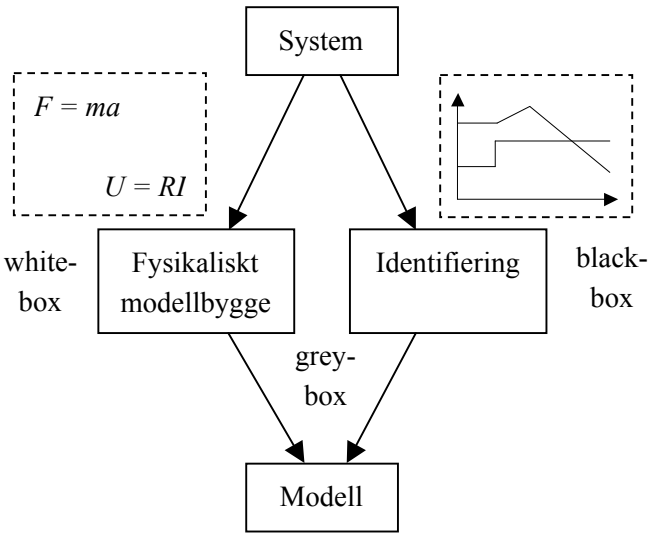


Fig. 8 Modellkonstruktion (Glad & Ljung, 1991).

Systemidentifiering (black-box identifiering) innebär att observationer används (in- och utsignaler) från systemet för att anpassa modellens egenskaper till systemets. Att ta fram en modell genom systemidentifiering är relativt enkelt men modellen har ofta ett begränsat giltighetsområde. Ofta används denna princip som komplement till den första. En vettig ansats är att modellera det som ”går” med fysikalisk insikt och sedan använda systemidentifiering för att bestämma okända parametrar, så kallad Grey-Box identifiering.

För att ta fram en modell genom systemidentifiering används data från systemet. Data består utav mätningar av variabler i systemet, insignaler, utsignaler och eventuella störsignaler. I princip finns det tre sätt att utnyttja identifieringsmetodik för modellbyggande:

1. Genomföra enkla experiment som skall underlätta strukturering av problemet.

2. Bygga modeller som förmår beskriva hur utsignalerna beror av insignalerna, men som inte är baserad på någon fysikalisk insikt om vad som händer inne i systemet. Modellerna är oftast linjära och det finns två angreppssätt.
   a) Dels kan modeller av godtyckliga linjära system byggas genom att deras impulssvar eller frekvensfunktion skattas.
   b) Dels kan linjära konfektionsmodeller användas.

3. Data kan användas för att bestämma okända systemparametrar i en fysikalisk modell som tagits fram tidigare och där okända samband finns, så kallad Grey-Box identifiering.

Punkterna 1 och 2 a) ovan är så kallade icke-parametriska identifieringsmetoder eftersom de inte direkt skattar några modellparametrar. Huvudverktyget för punkt 1 är transientanalys (se avsnitt 3.2.6) och för steg 2 a) korrelationsanalys, frekvensanalys eller spektralanalys. Punkterna 2 b) och 3 innebär skattning av parametrar i dynamiska modeller.

Svårigheten med att bygga en modell ligger i att göra den bra och tillförlitlig. Genom att verifiera och/eller validera en modell skaffas tilltro till de förutsägelser och resultat den ger. Modellverifiering innebär att modellens uppförande jämförs med systemets och skillnaden utvärderas. Alla modeller har ett begränsat giltighetsområde som det kan vara vanskligt att hamna utanför. Giltighetsområdet kan avse noggrannhetskraven, vilka värden på systemvariabler och systemparametrar som modellen gäller för (Glad & Ljung, 1991).

Som nämnts tidigare inkluderar grey-box modellering både fysikaliskt modellbygge och systemidentifiering. Modellen är baserad på tillgängliga fysikaliska kunskaper. Kunskaperna är dock ofullständiga och osäkra. De data som används för att identifiera de okända och osäkra delarna är också influerade av osäkra störningar. Därför måste modellen analyseras med uppmätta data. Identifieringen och analysen kan ge mer insikt i hur de okända delarna skall utformas. Detta ger en möjlighet att få mera kunskaper om funktionaliteten och fysikalisk förståelse om processen. Tyvärr är det både dyrare och det tar längre tid att konstruera en grey-box modell än en vanlig black-box modell. Hur som helst är det syftet med modellen som avgör vilken typ av modell som bör konstrueras (Sohlberg, 1998).

3.2.2 Fysikaliskt modellbygge

Som tidigare nämnts innebär fysikaliskt modellbygge att det utifrån kunskap om de grundläggande mekanismerna i systemet ställs upp en matematisk modell. Termen fysikaliskt modellbygge används eftersom det för de flesta fall är kunskaper i fysik som är relevanta. Givet ett system (som kan vara fysiskt, tekniskt, biologiskt, ekonomiskt och så vidare) skall systemet visas på formen (Glad & Ljung, 1991)

$$\begin{aligned} \frac{d}{dt}x(t) &= f(x(t),u(t)) \\ y(t) &= h(x(t),u(t)) \end{aligned} \tag{3.1}$$

Tre faser kan urskiljas i arbetet med att komma fram till en matematisk modell:

1. Problemet struktureras
2. Basekvationer ställs upp
3. Tillståndsmodellen formas

Fas 1 består av att försöka dela upp systemet på delsystem, bestämma de huvudsakliga orsakssambanden, vilka variabler som är viktiga och hur de påverkar varandra. I detta arbete är det viktigt att ha klart för sig vad syftet med modellen är. Det är också här som

grunden för modellen läggs vad gäller komplexitet och approximationsgrad. Det som fås ut av fas 1 är ett blockschema, eller ett diagram av motsvarande slag.

Fas 1 kan sammanfattas på följande sätt:

- Bestäm utsignaler och externa signaler för modellen. Bestäm vilka interna variabler som är av vikt för att beskriva systemets uppförande.
- Illustrera sambanden mellan externa signaler, interna variabler och utsignaler i ett blockschema, se figur 9.

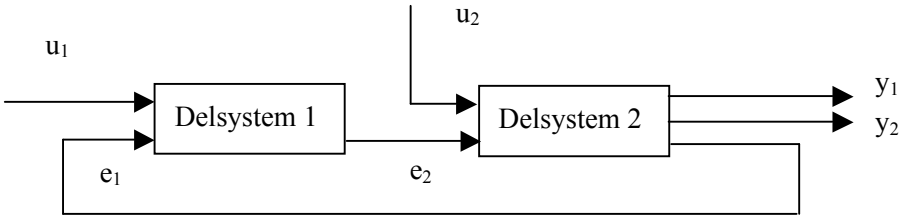


Fig. 9 Exempel på blockschema från fas 1.

Fas 2 innebär att de delsystem, ”block”, som struktureringen i fas 1 ledde till betraktas. Således ställs de samband som råder mellan variabler och konstanter i delsystemen upp. Där användes, för fysikaliska system, de ”naturlagar” och fysikaliska basekvationer som antas gälla. Basekvationerna kan delas upp i två grupper; balansekvationer och konstitutiva relationer. Balansekvationer relaterar storheter av samma slag, till exempel flöde in – flöde ut = upplagrad volym per tidsenhet. Konstitutiva relationer relaterar storheter av olika slag, som Ohms lag. Här införs en del approximationer och idealiseringar.

Fas 2 kan sammanfattas med:

- Ställ upp den balansekvation som gäller för blocket – delsystemet i fråga.
- Använd tillämpliga konstitutiva relationer för att uttrycka balansekvationen i termer av modellens variabler. Gör dimensionsanalys av storheterna som kontroll.

Fas 3 är till skillnad från de övriga ett mer formellt steg som har till uppgift att på lämpligt sätt organisera de många ekvationer och uttryck som fas 2 lämnar efter sig. Även om modellen i och för sig är given redan i fas 2 är detta steg nödvändigt för att ge en modell som kan användas för analys eller simulering. Det som erhålls från fas 3 kan till exempel vara tillståndsmodeller.

Detta sammanfattas i följande tre steg:

1. Välj en uppsättning tillståndsvariabler.
2. Uttryck tidsderivatan av var och en av dessa med hjälp av tillståndsvariablerna och insignalerna.
3. Uttryck utsignalerna som funktion av tillståndsvariablerna och insignalerna.

Lyckas steg 2 och 3 har en representation av formen (3.1) erhållits. Det svåra ligger i hur tillståndsvariablerna ska väljas. Det kan ses som att alla interna variabler som svarar mot upplagring av olika storheter, som upplagrad volym eller upplagrad termisk energi, är kandidater för att väljas som tillståndsvariabler (Glad & Ljung, 1991).

3.2.3 Kalmanfilter

Ur föregående avsnitt fås systemet på tillståndsform enligt ekvation (3.1). Tillstånden $x(t)$ är ofta omätbara och måste därför skattas. Det kan göras med ett Kalmanfilter. Introducera följande observatör (Glad & Ljung, 1991):

$$\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + K(t)(y(t) - C\hat{x}(t)) \tag{3.2}$$

För att kunna skatta observatören måste processbrus och mätbrus läggas på modellen:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + e(t) \end{aligned} \tag{3.3}$$

där $Ew(t)w^T(t) = R_1$ och $Ee(t)e^T(t) = R_2$. Observatörens förstärkning fås sedan ur

$$K(t) = AP(t)C^T(CP(t)C^T + R_2)^{-1} \tag{3.4}$$

där P är lösningen till Riccatiekvationen

$$P(t+1) = AP(t)A^T + R_1 - AP(t)C^T(CP(t)C^T + R_2)^{-1}CP(t)A^T. \tag{3.5}$$

Detta ger en ny tillståndsekvation på formen

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + Ke(t) \\ y(t) &= C\hat{x}(t) + e(t) \end{aligned} \tag{3.6}$$

3.2.4 Systemidentifiering

3.2.4.1 Skräddarsydda modeller

Då det i fas 2 av modellbygget ställs upp relationer mellan systemets variabler ses i regel att en del systemkonstanter har värden som inte är kända. Den resulterande tillståndsmodellen (3.1) blir då i princip av formen (Glad & Ljung, 1991)

$$\begin{aligned} \frac{d}{dt}x(t) &= f(x(t), u(t), \theta) \\ y(t) &= h(x(t), u(t), \theta) \end{aligned} \tag{3.7}$$

där parametervektorn θ innehåller de okända systemparametrarna. Med *d* antal sådana parametrar blir

$$\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} \tag{3.8}$$

Dessa kan ofta bestämmas med hjälp av mätningar, vilket hör till det fysikaliska modellbygget. I de fall de inte går att mäta kan dessa estimeras utifrån systemidentifiering.

3.2.4.2 Linjära konfektionsmodeller

Linjära konfektionsmodeller är en så kallad standardmodell som av erfarenhet är kapabel att kunna hantera en mängd olika fall av systemdynamik. Parametrarna har i princip ingen fysikalisk tolkning utan används för att kunna beskriva egenskaper hos systemets insignal-utsignal-samband. En allmän linjär, tidsdiskret modell kan skrivas (Glad & Ljung, 1991)

$$y(t) = \eta(t) + w(t). \tag{3.9}$$

Här är $w(t)$ en störterm och $\eta(t)$ den störningsfria utsignalen, som kan skrivas

$$\eta(t) = G(q^{-1}, \theta)u(t) \tag{3.10}$$

där $G(q^{-1}, \theta)$ är en rationell funktion av förskjutningsoperatorn q^{-1},

$$G(q^{-1}, \theta) = \frac{B(q^{-1})}{F(q^{-1})} = \frac{b_1 q^{-nk} + b_2 q^{-nk-1} + \ldots + b_{nb} q^{-nk-nb-1}}{1 + f_1 q^{-1} + \ldots + f_{nf} q^{-nf}}, \tag{3.11}$$

där insignalen är tidsfördröjd *nk* sampel. På samma sätt kan störtermen skrivas

$$w(t) = H(q^{-1}, \theta)e(t) \tag{3.12}$$

med

$$H(q^{-1}, \theta) = \frac{C(q^{-1})}{D(q^{-1})} = \frac{1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc}}{1 + d_1 q^{-1} + \ldots + d_{nd} q^{-nd}} \tag{3.13}$$

där $e(t)$ är vitt brus. Detta kan nu sammanfattas som

$$y(t) = G(q^{-1}, \theta)u(t) + H(q^{-1}, \theta)e(t) \tag{3.14}$$

Konfektionsmodellen (3.14) är känd som *Box-Jenkins* (*BJ*)-modellen, efter statistikerna Box och Jenkins. Ett viktigt specialfall är att strunta i att modellera störsignalens egenskaper och sätta $H(q^{-1}) \equiv 1$, alltså $nc = nd = 0$. Detta specialfall kallas *Output-Error* (*OE*)-modellen

$$y(t) = G(q^{-1},\theta)u(t) + e(t) \tag{3.15}$$

En ofta använd variant är att låta nämnarna till G och H sammanfalla, $F(q^{-1}) = D(q^{-1}) = A(q^{-1}) = 1 + a_1q^{-1} + \ldots + a_{na}a^{-na}$. Genom att förlänga med $A(q^{-1})$ kan då (3.14) skrivas som

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t) \tag{3.16}$$

Denna konfektionsmodell kallas *ARMAX* (AutoRegression Moving Average med en eXtra insignal)-modellen. Slutligen fås specialfallet av (3.15) att $C(q^{-1}) \equiv 1$, alltså $nc = 0$

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t) \tag{3.17}$$

vilket med samma taxonomi kallas en *ARX*-modell.

3.2.4.3 Linjär regression

Både de skräddarsydda modellerna och konfektionsmodellerna ger en framställning av hur det predikterade värdet hos y beror på gamla värden på y och u och av parametrarna θ (Glad & Ljung, 1991). I allmänhet kan detta vara en ganska komplicerad funktion av θ. Skattningsarbetet underlättas emellertid betydligt om prediktionen är en linjär funktion av θ:

$$\hat{y}(t) = \theta^T\varphi(t) \tag{3.18}$$

Här är θ en kolonnvektor som innehåller de okända parametrarna, medan $\varphi(t)$ är en kolonnvektor formad av gamla insignaler och utsignaler. Vektorn $\varphi(t)$ kallas regressionsvektorn och dess komponenter kallas regressorer. ARX-modellen (3.17) utgör det vanligaste exemplet på (3.18). ARX-modellen (3.17) kan skrivas på formen (3.18) genom att definiera

$$\theta = \begin{bmatrix} a_1 & \ldots & a_{na} & b_1 & \ldots & b_{nb} \end{bmatrix}^T \tag{3.19}$$

$$\varphi(t) = \begin{bmatrix} -y(t-1) & \ldots & -y(t-na) & u(t-nk) & \ldots & u(t-nk-nb+1) \end{bmatrix}^T \tag{3.20}$$

För det flervariabla fallet betrakta ARX-modellen med *nu* insignaler och *ny* utsignaler

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t) \tag{3.21}$$

där nu $A(q^{-1})$ och $B(q^{-1})$ är följande matrispolynom med dimensionen ($ny|ny$) och ($ny|nu$) respektive:

$$\begin{aligned} A(q^{-1}) &= I + A_1 q^{-1} + \ldots + A_{na} q^{-na} \\ B(q^{-1}) &= B_1 q^{-1} + \ldots + B_{nb} q^{-nb} \end{aligned} \tag{3.22}$$

Antag att alla element i matriserna A_1, …, A_{na}, B_1, …, B_{nb} är okända. De kommer därför att ingå i parametervektorn. Modellen (3.21) kan alternativt skrivas som (Söderström & Stoica, 1989)

$$y(t) = \Phi^T(t)\Theta + e(t) \tag{3.23}$$

där

$$\Phi^T(t) = \begin{pmatrix} \varphi^T(t) & & 0 \\ & \ddots & \\ 0 & & \varphi^T(t) \end{pmatrix} \tag{3.24}$$

$$\varphi^T(t) = \begin{bmatrix} -y(t-1) & \ldots & -y(t-na) & u(t-1) & \ldots & u(t-nb) \end{bmatrix}^T \tag{3.25}$$

$$\Theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{ny} \end{pmatrix} \qquad \begin{pmatrix} \theta_1^T \\ \vdots \\ \theta_{ny}^T \end{pmatrix} = \begin{pmatrix} A_1 & \ldots & A_{na} & B_1 & \ldots & B_{nb} \end{pmatrix} \tag{3.26}$$

3.2.4.4 Anpassning av modeller till data

Givet en parametrisk modell och gamla mätdata kan värdet på utsignalen vid tiden t predikteras. Prediktionen betecknas $\hat{y}(t,\theta)$ och prediktionsfelet definieras som

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t,\theta) \tag{3.27}$$

Det är naturligt att välja den modell θ som minimerar prediktionsfelets varians. Givet mätdata $\{y(t),u(t)\}_{t=1,\ldots,N}$ bildas förlustfunktionen

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon^2(t,\theta) \tag{3.28}$$

och bestäms det θ som minimerar $V_N(\theta)$:

$$\theta_N = \arg\min_{\theta} V_N(\theta) \tag{3.29}$$

Då prediktionen kan skrivas som en linjär funktion av θ är minimeringen av förlustfunktionen speciellt enkel. För en ARX-modell kan prediktionen skrivas som (3.18) eller för flervariabla fallet som (3.23). Motsvarande prediktionsfel blir då

$$\varepsilon(t,\theta) = y(t) - \theta^T \varphi(t) \tag{3.30}$$

Förlustfunktionen $V_N(\theta)$ minimeras i detta fall av

$$\theta_N = R_N^{-1} f_N \tag{3.31}$$

där

$$f_N = \frac{1}{n} \sum_{t=1}^{N} \varphi(t) y(t) \tag{3.32}$$

$$R_N = \frac{1}{n} \sum_{t=1}^{N} \varphi(t) \varphi^T(t) \tag{3.33}$$

För ARX-modellen består elementen i f_N och R_N av skattningar av olika kovariansfunktioner för y och u. Denna metod att minimera förlustfunktionen kallas för minstakvadratskattningen eller *Least Square* (*LS*) och gäller då R_N^{-1} excisterar.

3.2.4.5 Modellens egenskaper

När modellkvalitet diskuteras, menas att modellen ligger så nära en ”sann” beskrivning som möjligt. Men vad är en ”sann” beskrivning? Modellkvalitet är relaterat till modellanvändning. En modell kan till exempel vara utmärkt för ett visst ändamål men otillräcklig för ett annat. Modellkvalitet är också relaterat till modellens ”stabilitet”, det vill säga hur reproducerbar själva modellen är från uppmätta data. De modellofullkomligheter som finns tar sig i uttryck på, i princip, två olika sätt. Det ena är de modellfel som uppstår på grund av att mätningarna och systemet påverkas av mätbrus (variansfel) och det andra är brister i själva modellstrukturen (biasfel). Variansfelen kan typiskt reduceras genom att använda längre mätsekvenser. Biasfel ger sig tillkänna som variationer i modellen då den anpassas till datamängder som samlats in under olika betingelser (även med obetydligt variansfel). En bra modell är alltså en som har både litet variansfel och litet biasfel (Glad & Ljung, 1991).

3.2.5 Validering

Innan en modell kan användas måste den valideras, det vill säga undersökas om den duger för syftet. Valideringen sker ofta löpande med framtagandet av modeller. Olika modellstrukturer prövas och undersöks vilka som ger bäst resultat. Flera metoder för validering finns för detta.

Korsvalidering
Ett enkelt och i vissa fall avgörande sätt för att validera en modell är att simulera modellen från insignalen enbart och jämföra den simulerade utsignalen med den uppmätta. Detta skall helst göras på en datamängd som inte har använts för identifieringen.

Residualanalys
Prediktionsfelen som modellen ger analyseras. Dessa skall hålla sig innanför ett visst konfidensintervall. Gör de inte det har modellen inte klarat av att beskriva systemet på ett korrekt sätt. Det går också att titta på summan av prediktionsfelen för att avgöra vilken som ger bäst passning.

Akaikes Information Criteria (AIC)
AIC beräknas utifrån kalibreringsdata.

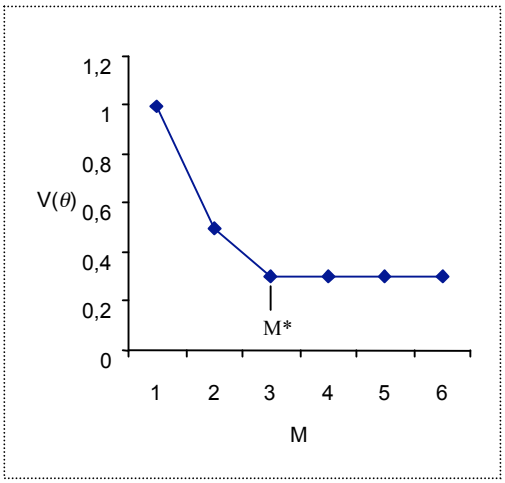
3.2.6 Transientanalys

Innan fysikalisk modellering eller systemidentifiering kan göras måste de storheter och variabler som är viktiga för att beskriva vad som händer bestämmas. En vanlig form av experiment som ger insyn i hur variabler påverkar varandra och i vilken tidsskala detta sker är så kallad stegsvarsanalys eller transientanalys. Metoden innebär att insignalerna varieras, en i taget, som ett steg: $u(t) = u_0,\ t < t_0;\ u(t) = u_1,\ t \geq t_0$. Övriga mätbara variabler i systemet registreras under tiden. Alternativt kan systemets impulssvar studeras genom att låta insignalen vara en kortvarig puls (Dirac puls). Från dessa mätningar kan följande information erhållas (Glad & Ljung, 1991):

- Vilka variabler som påverkas av insignalen. Detta gör det lättare att rita blockschema för systemet.
- Vilka tidskonstanter som systemet har.
- Vilken karaktär (oscillativ, svagt dämpad, monoton etc.) som stegsvaren har samt nivån på den statiska förstärkningen.

3.2.7 Val av modelldimension

Att välja lämplig modelldimension är inte självklart. En avvägning mellan modellordning och anpassning av modellen till valideringsdata måste ske. Med ökad modellordning (antalet fria parametrar ökar) blir modellen bättre kalibrerad till observerade data, men modellen anpassas också alltmer till störningar och brus. Detta resulterar i att det är viktigt att undersöka om förbättringen i modellen är signifikant. I figur 10a visas förhållandet mellan förlustfunktionen och modellordningen M för ett idealt fall. Antag att data är brusfritt eller att $N = \infty$ och att det finns en modellstruktur M^* som kan beskriva systemet exakt. I det ideala fallet kommer förlustfunktionen $V_N(\theta)$ vara konstant när $M \geq M^*$ (se figur 10a). När systemet inte är idealt (brus, $N < \infty$) kommer förlustfunktionen $V_N(\theta)$ minska med ökat M. Problemet ligger då i att avgöra när skillnaden $V = V_1 - V_2$ är litet eller ej, se figur 10b. Om skillnaden är liten ska modellstrukturen M_1 väljas, annars bör M_2 väljas, se figur 10b (Söderström & Stoica, 1989).



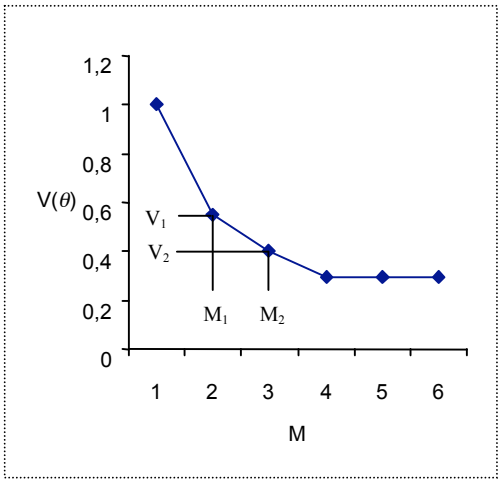


Fig. 10a Minimivärdet av förlustfunktionen som funktion av modellordningen för det ideala fallet.

Fig. 10b Minimivärdet av förlustfunktionen som funktion av modellordningen för det icke ideala fallet.

3.3 NÅGRA PRAKTISKA STEG VID SYSTEMIDENTIFIERING

Identifieringsprocessen går ut på att upprepade gånger välja en modellstruktur, beräkna den bästa modellen i modellstrukturen och evaluera dess egenskaper för att se om den är tillfredsställande. Den iterativa proceduren kan summeras på följande vis (Ljung, 2002):

1. Design av experiment. Bestäm intressanta parametrar.
2. Samla in- och utdata från processen som ska identifieras.
3. Undersök data. Polera dataserierna genom att ta bort trender och avvikande (outliers) värden. Välj användbara delar av dataserierna som innehåller mycket information om systemet. Filtrera vid behov för att förstärka viktiga frekvensområden.
4. Välj och definiera en modellstruktur.
5. Beräkna den bästa modellen i modellstrukturen från in- och utdata och ett givet minimeringskriterie.
6. Undersök den erhållna modellens egenskaper.
7. Om modellen är tillräckligt bra avbryts förfarandet, annars börja om från steg 3 för att pröva en ny modellstruktur. Eventuellt försök med en annan estimeringsmetod (steg 4) eller arbeta mer med in- och utdata (steg 1 och 2).

Ett bra hjälpmedel för att utföra dessa steg är *Matlabs System Identification Toolbox*.

3.4 FELDETEKTERING

3.4.1 Introduktion

Tillsyn av industriella processer blir mer och mer viktigt med den ökade automatiseringen i industrin. Processövervakning och tillståndsbeskrivning är ett tekniskt frambrytande område på grund av ökade krav på produktkvalitet, låg produktionskostnad, reducerad underhållskostnad av maskiner och förlust av produktionstid på grund av produktionsstopp. Många av faktorerna är inte bara reglerproblem utan också problem på en högre nivå av processövervakning. Det övervägande syftet av processövervakning är att förbättra hela systemets prestanda.

Vanligtvis sker övervakning genom att värden på viktiga processparametrar kontrolleras. När en process ska övervakas installeras ofta fler givare och dess värden presenteras för processoperatörerna. Eftersom antalet sensorer och givare ökar och således informationsmängden till operatören blir det svårt att analysera informationen. Ett sätt att lösa detta på är att använda sig av processmodeller och med datorhjälpmedel estimera processparametrarna eller genom att använda sig av statistiska metoder för att ge underlag till beslut som påverkar processen.

En vanlig typ av fel uppstår från förslitningar genom att objektets funktion sakta försämras. Framförallt mekaniska system har rörliga delar som utsätts för förslitningar. Många processer är också utsatta för andra fel än förslitningar. Dessa sker ofta abrupt och måste upptäckas snabbt och skiljas från förslitningar. Typiska abrupta fel är ställdonsfel, sensorfel och fel som beror på produktionsstopp (Sohlberg, 1998).

Design av ett feldetekteringssystem är en avvägning mellan sannolikheten att upptäcka förändringar och risken om att felaktigt besluta att en förändring inträffat (det vill säga falsklarm). Om ett system larmar ofta och det visar sig vara falsklarm kommer operatörer snart att frångå feldetekteringssystemet då förtroendet för systemet har raserats (Sjölund, 2002).

Residualerna ($e = y - \_$) det vill säga differensen mellan det sanna värdet för systemet (y) och det skattade värdet (_) för ett system används ofta för feldetektering. Om systemet är idealt och modellen perfekt kommer residualerna att vara identiskt lika med noll innan en förändring och skilt från noll efter ett fel. I verkligheten finns det processbrus och mätbrus varvid det faktiska värdet på residualen inte kan predikteras exakt. Medelbeteendet utnyttjas i stället. Om det inte sker någon förändring i systemet och antagen modell är korrekt kommer residualerna vara på formen vitt brus om mätbruset är vitt. Medelvärdet eller variansen eller båda förändras efter en förändring, residualerna blir ”stora”. Bestämmandet av vad ”stora” är, är problemet i statistisk feldetektering (Sjölund, 2002).

De typer av fel som kan uppstå är i regel två sorter, abrupta och långsamma fel. Nedan följer två metoder för feldetektering. Dessa metoder är för on-line detektering.

3.4.2 Detektering av långsamma fel

Förslitning är en långsam tidsvarierande process och är ett resultat av att effektiviteten i någon del av processen avtar. En generell beskrivning av situationen visas i figur 11. Systemet har en process och en estimator som kan identifiera långsamma tidsvarierande parametrar. Estimaten används som ett beslutsunderlag för processen.

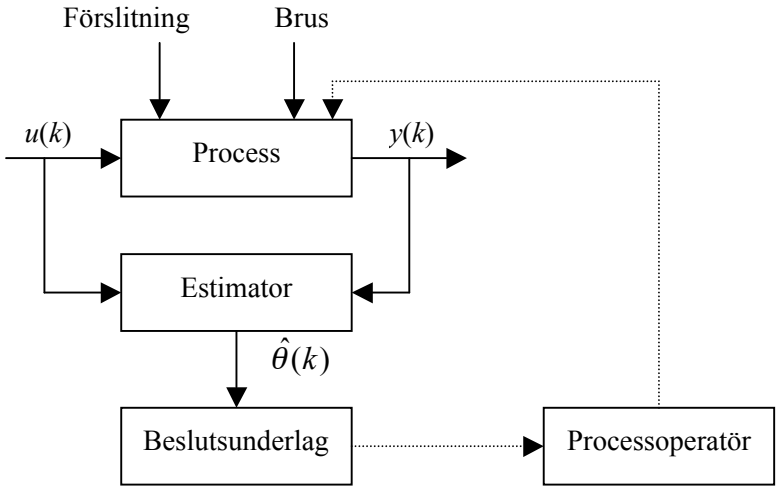


Fig. 11 Process som påverkas av förslitning (Sohlberg, 1998).

Processen påverkas av insignalen $u(k)$, förslitningar och störningar. Estimatorn identifierar modellparametrarna och där lämpligt, estimerar processtillstånden. Estimatorn är baserad på en passande processmodell, vars komplexitet varierar beroende på syftet med övervakningen (Sohlberg, 1998).

Estimatorn kan också vara en modell med fasta modellparametrar. För det fallet jämförs modellens skattade värde med det riktiga på samma sätt som för abrupta fel.

3.4.3 Detektering av abrupta fel

Abrupta fel i ett system delas grovt upp i två typer (Basseville & Nikiforov, 1993). Den första sorten är avvikelse från referensvärdet μ_0 till μ_1 med konstant standardavvikelse på bruset (se figur 12a). Denna avvikelse är ett systematiskt fel. Den andra sorten är ökning i standardavvikelsen från σ_0 till σ_1 med oförändrat medelvärde (se figur 12b). Avvikelsen är ett slumpmässigt fel.

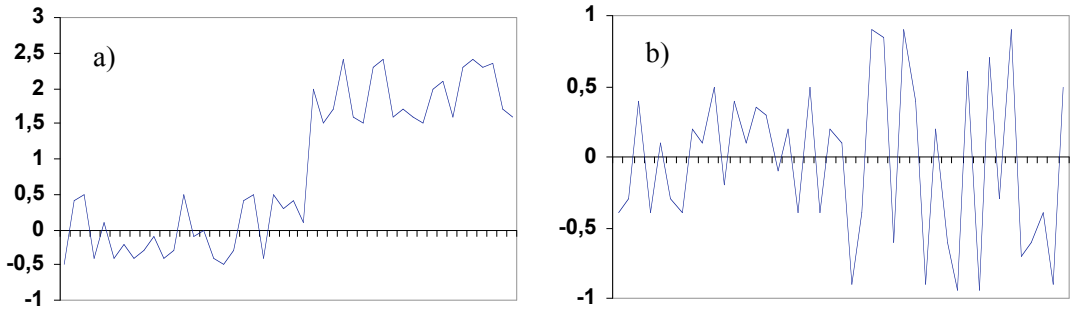


Fig. 12a) Avvikelse från referensvärdet μ_0 till μ_1 med konstant standardavvikelse på bruset och **b)** ökning av standardavvikelsen på bruset från σ_0 till σ_1 med oförändrat medelvärde.

För detektering av abrupta förändringar kan en testvariabel g_t införas, som används för beslut om larm om $g(t)$ överskrider ett gränsvärde h, det vill säga

$$\text{larm då } g(t) > h.$$

De flesta feldetekteringsalgoritmer kan omformuleras till problemet att bestämma vilken av följande två hypoteser som är riktig.

$$H_0: \mathrm{E}(e(t)) = 0$$
$$H_1: \mathrm{E}(e(t)) \neq 0$$

H_0 innebär att ingen förändring har inträffat och H_1 att en förändring har inträffat. Stoppvilkor uppnås genom att lågpassfiltrera $e(t)$ och jämföra det lågpassfiltrerade värdet med ett gränsvärde.

Det finns många olika metoder för att upptäcka om ett fel har uppstått. Nedan följer en icke-linjär metod (SPRT) och dess specialfall (CUSUM).

3.4.3.1 SPRT (Sequential Probability Ratio Test Statistics)

SPRT-algoritmen:

$$g(t) = g(t-1) + e(t) - v$$
$$g(t) = 0, \quad och \quad \hat{k} = t \quad om \quad g(t) < a < 0$$
$$g(t) = 0, \quad och \quad t_a = t \quad och \quad larm \quad om \quad g(t) > h > 0$$

Testvariabeln $g(t)$ summerar indata $e(t)$ och ger ett larm då summan överstiger ett gränsvärde h. Om indata är vitt brus kommer testvariabeln att driva iväg. För att motverka att variabeln driver iväg i positiv riktning, vilket resulterar i ett falskt larm, så subtraheras en liten term v i varje tidssteg. För att motverka en negativ drift, vilket skulle orsaka att tiden för upptäckt efter förändring ökar, så nollställs variabeln $g(t)$ när

den är mindre än en negativ konstant a. Konstanten a skall väljas liten i storlek och ett vettigt val som förklarats väl i litteraturen är $a = 0$. Detta viktiga specialfall har fått benämningen CUSUM (Gustafsson, 2000).

3.4.3.2 CUSUM (Cumulativ SUM)

CUSUM-algoritmen:

$$g(t) = g(t-1) + e(t) - v$$
$$g(t) = 0, \quad och \quad \hat{k} = t \quad om \quad g(t) < 0$$
$$g(t) = 0, \quad och \quad t_a = t \quad och \quad larm \quad om \quad g(t) > h > 0$$

Antagandet här är att residualerna $e(t)$ består av en deterministisk komponent $\gamma(t)$, signalen, och ett vitt brus $w(t)$,

$$e(t) = \gamma(t) + w(t)$$

Driften v skall väljas som hälften av den kritiska nivån som inte får överskridas av den fysiska variabeln $\gamma(t)$. Då $\gamma(t) = 0$ kommer $g(t)$ att sättas till noll vid nästan varje tidpunkt (beror på brusnivån och om $a < -v$ är valt). Efter en förändring till $\gamma(t) > v$, kommer $g(t)$ att växa och kommer inte att nollställas förrän larm beslutas. Genom att kräva att flera $g(t) > h$ kan robustheten öka och antalet falska larm minska. Detta är en kvalitetskontroll som kallas *run test* (Gustafsson, 2000).

Algoritmen ovan antar att $g(t)$ blir positivt vid ett fel. I ett dubbelsidigt test går det även att upptäcka fel då $g(t)$ blir negativt vid ett fel. Då sätts helt enkelt $g(t) > h_1$ och $g(t) < h_2$.

4 BEHANDLING AV MÄTDATA

4.1 INLEDNING

De modelleringsproblem som har studerats är bärlagertemperaturens beroende av kylvattentemperatur och kylvattenflöde. På grund av förseningar i installationen av givare på kraftstationen har endast ett fåtal data kunnat samlas in. Flera av de givare som varit installerade har dessutom visat något helt annat värde, till exempel bärlagertemperaturen visade oljetemperaturen (fig. 13). Detta beroende på att det vid installationen inte angivits rätt portadress i koncentratorn. Dessutom har de givare som installerats i tid inte varit vare sig skalade eller kalibrerade. Skalning och kalibrering gjordes under oktober och november månad, vilket kan ses i figur 13 och figur 14 som hopp eller störningar i mätdata. På grund av alla dessa förseningar har det inte gått att samla in mer mätvärden än över ett par veckor. För att en framtagen modell ska fungera över hela året under alla årstider bör modellen åtminstone baseras på data över minst ett år (Glemme, 2001).

4.2 FÖRBEHANDLING AV DATA

4.2.1 Insamling av data

De data som samlats in är bärlagertemperaturen, kylvattentemperaturen, kylvattenflödet och producerad effekt. Producerad effekt behövs för att undersöka om generatorn är i drift eller inte. Det finns två olika sätt att hämta längre lagrade dataserier. I databasen i Råcksta lagras dygnsdata som är dygnets högsta, lägsta eller medelvärdet under dygnet. Dessa värden är dock ointressanta för modelleringen då det ska vara data samplade med minst en minuts intervall för att få med abrupta variationer. Det andra sättet är att samla in data på plats vid Söderfors kraftstation. Detta görs med ett DOS-program som erhölls från Conwide AB. Inkoppling sker mot koncentratorn och data kopieras till en diskett. Detta måste göras en gång i månaden för att vara säker på att få med alla data. Beroende på hur mycket värdena ändras sparas de med olika intervall. Det kortaste intervallet är en sekund och det längsta en minut. Ändras inte värdena mer än en gång per minut räcker det att hämta värden en eller ett par gånger per år. Med denna metod att spara data på får dataserierna olika samplingsintervall. Det är den här metoden som har använts. Förfarandet är en tillfällig lösning. Vid framtida implementering kommer data från databasen i Råcksta att kunna hämtas. Databasen kommer inom kort även att lagra minutvärden.

4.2.2 Förbehandling

Eftersom de rådata som hämtats ur koncentratorn har varierande samplingsintervall (från några sekunder till en minut) har en förbehandling krävts för att återskapa en dataserie med regelbundet samplingsintervall. För detta skrevs ett program i Matlab, *utfyllnad.m* (se appendix 7). Data behöver även behandlas innan detta program kan köras. För beskrivning se appendix 6. Programmet fyller ut dataserien med de värden som saknas och sparar undan i en fil en gång i minuten. Denna fil kan sedan användas för systemidentifieringen.

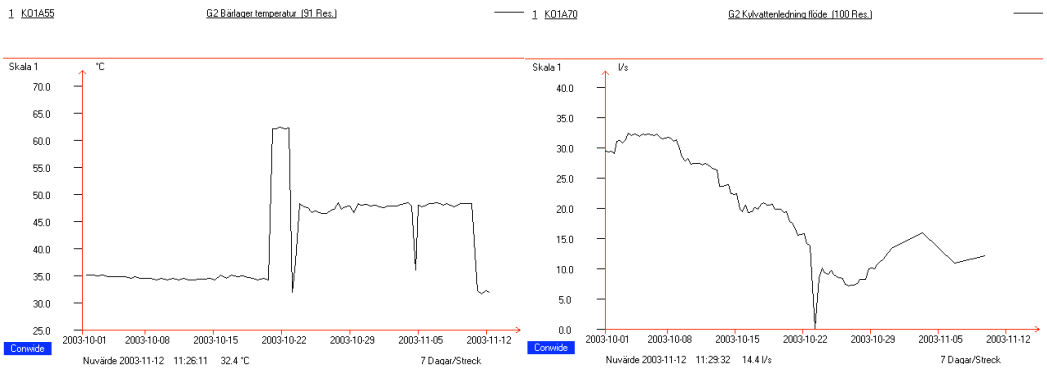


Fig. 13 Bärlagertemperaturen och kylvattenflödet från 1 oktober till 12 november 2003. De stora variationerna i bärlagertemperaturen beror på skalning och kalibrering. Det vänstra diagrammet visar oljetemperaturen till en början för att sedan ändras till bärlagertemperatur, men med fel skala. Omskalningen gjordes i början på november.

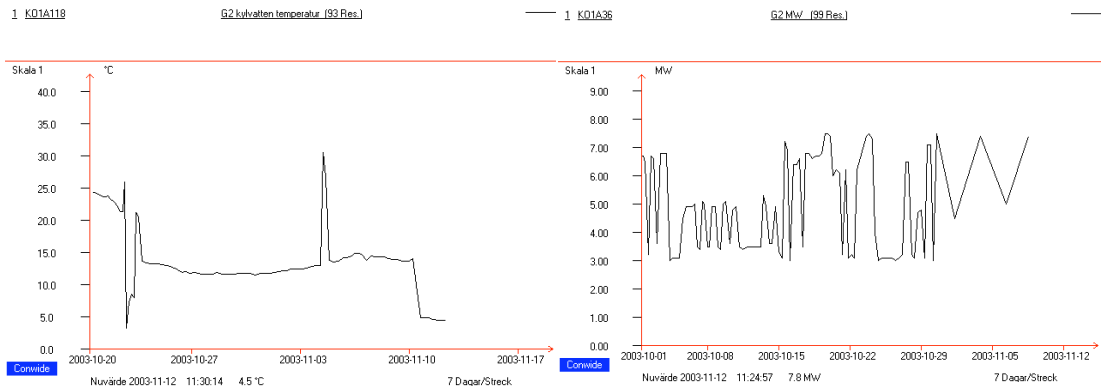


Fig. 14 Kylvattentemperaturen från mitten av oktober till 12 november samt producerad effekt från 1 oktober till 12 november 2003. De stora variationerna i kylvattentemperaturen beror på skalning och kalibrering.

5 SYSTEMDESIGN

5.1 INLEDNING

I Söderfors kraftverk är det två parametrar som framförallt påverkar bärlagertemperaturen. Dels är det kylvattentemperaturen och dels är det kylvattenflödet. Kylvattenflödet är reglerat med avseende på lageroljans temperatur. Detta innebär att det är ett återkopplat system, där det är svårt att identifiera processparametrarna. I tidigare examensarbete, Sjölund 2002 (se appendix 2), har en RLS-algoritm använts för skattning av bärlagertemperaturen. Den klarade inte av att identifiera långsamma temperaturförändringar och den var endast baserad på bärlagertemperaturen då vissa anläggningar inte har reglerat kylvattenflöde och då räcker kylvattentemperaturen för identifieringen. För att klara av att både identifiera abrupta och långsamma temperaturförändringar över ett brett område har fysikaliskt modellbygge undersökts. Det fysikaliska modellbygget har resulterat i en modell med flera okända parametrar. Identifieringen av dessa parametrar visade sig vara svår varvid en stationär modell togs fram då förändringar i bärlagertemperaturen generellt sker mycket långsamt.

5.2 FYSIKALISKT MODELLBYGGE

Som tidigare beskrivits i kapitel 3 är fysikaliskt modellbygge uppdelat i tre faser. Den första fasen går ut på att konstruera ett blockschema, för att få en tydlig begränsning på problemet. Den andra fasen går ut på att ta fram de ekvationer och samband som råder mellan de inblandade variablerna. I den tredje och sista fasen sammanställs ekvationerna till tillståndsmodeller för att kunna användas på ett vettigt sätt. Här har även en ekvation tagits fram för det stationära fallet. Tillståndsmodellen eller ekvationen för det stationära fallet syftar till att kunna beräkna en skattad bärlagertemperatur för hela året. För det krävs befintliga dataserier för ett helt år.

5.2.1 Fas 1 – Blockschema

Den första fasen går ut på att definiera problemet och förenkla det så långt som möjligt. Här ska det bestämmas vilka signaler och interna variabler som påverkar systemet.

Systemet som skall modelleras är kylsystemet för bärlagertemperaturen i en vattenkraftturbin. Det som sker är att kylvatten leds in från dammen, se figur 15. Temperaturen och flödet mäts med givare. Flödet är dessutom reglerat och beror på oljetemperaturen. Kylvattnet kyler sedan bärlageroljan för att få ner den till önskad temperatur. Därefter transporteras oljan till bärlagret där oljan smörjer och kyler bärlagret.

Figur 15 visar hela systemet. Ur detta kan urskiljas två delsystem, ett som kyler oljan och ett som kyler bärlagret. För att lättare se hur de påverkar varandra delas systemet upp i mindre delar (se figur 16).

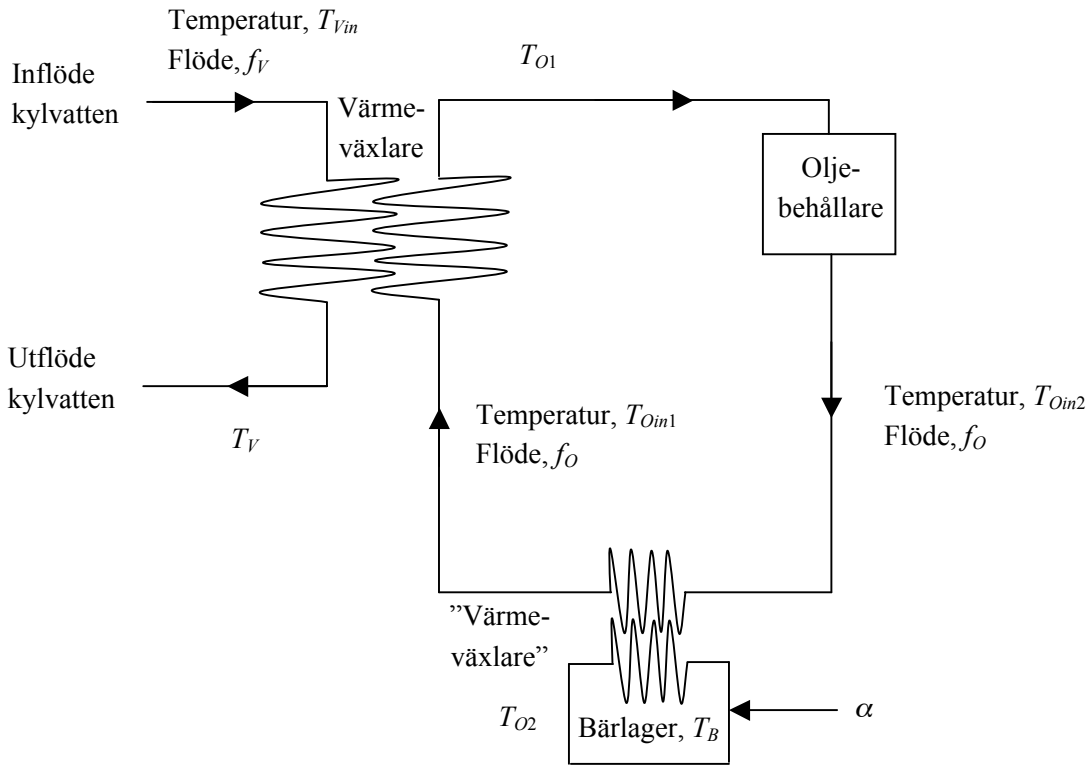


Fig. 15 En principskiss över bärlagerkylsystemet.

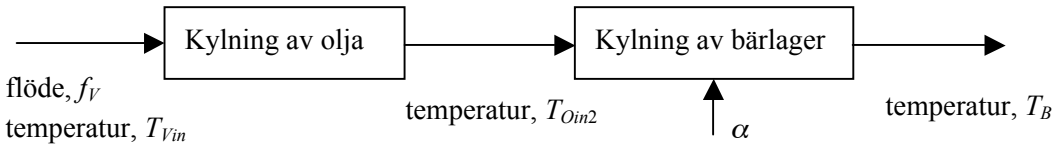


Fig. 16 Blockschema för kylsystemet. α är tillförd värme till bärlagret.

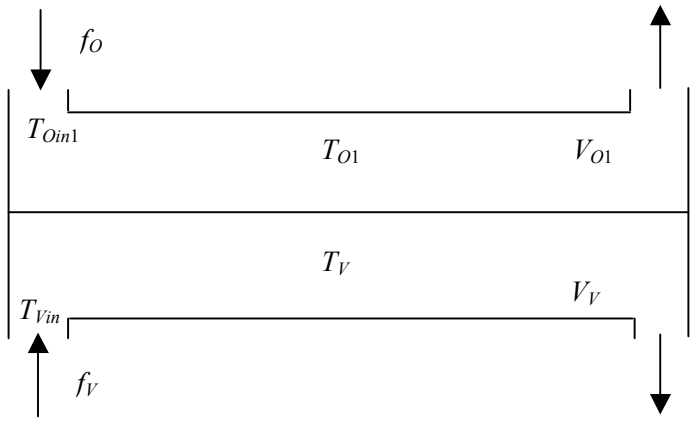


Fig. 17 En enkel värmeväxlare. f_V och f_O är flödena av kallt vatten och varm olja. T_{O1}, T_V, T_{Oin1}, T_{Vin} betecknar temperaturerna vid respektive punkter, och V_V och V_{O1} är volymerna av vatten och olja (Glad & Ljung, 1997).

5.2.2 Fas 2 – Ekvationer och samband

Varje del kan ses som en separat värmeväxlare (fig. 17), där f_V och f_O är flödena av kallt vatten och varm olja. T_{O1}, T_V, T_{Oin1}, T_{Vin} betecknar temperaturerna vid respektive punkter, och V_V och V_{O1} är volymerna av vatten och olja. Värmeväxlare är en apparat avsedd att överföra värme (energi) från ett medium till ett annat (Alvarez, 1990). Med en del förenklingar kan funktionen beskrivas på följande sätt (Glad & Ljung, 1997). In i den kalla delen strömmar vattenflödet f_V (m^3/s) med temperaturen T_{Vin} (K). På motsvarande sätt strömmar flödet f_O av varm olja med temperaturen T_{Oin1} in i den varma delen. Då vattnet och oljan ”möts” (i skilda rör), värmer oljan upp det kalla vattnet till temperaturen T_V. Samtidigt kyls det själv till T_{O1}. Värmemängden i den kalla delen av värmeväxlaren varierar enligt (Weyer et al., 2000)

$$V_V \frac{dT_V}{dt} = f_V (T_{Vin} - T_V) + \frac{\mu_1 A_1}{c_{pV} \rho_V} (T_{O1} - T_V). \tag{5.1}$$

Här representerar första termen på höger sida den tillförda värmemängden i det kalla inflödet (normalt är ju $T_{Vin} \leq T_V$ så det är egentligen fråga om bortfört värme). Den andra termen anger hur mycket värme som förts bort från den varma till den kalla delen av värmeväxlaren. Denna är proportionell mot temperaturskillnaden. Parametern A_1 (m^2) är värmeöverföringsarean, c_{pV} (J/Kkg) är specifik värmekapacitet, ρ_V (kg/m^3) densiteten och μ_1 (J/m^2Ks) är värmegenomgångskoefficient (Alvarez, 1990) som beror på materialet i värmeväxlaren. På motsvarande sätt fås för den varma delen

$$V_{O1} \frac{dT_{O1}}{dt} = f_O (T_{Oin1} - T_{O1}) - \frac{\mu_1 A_1}{c_{pO} \rho_O} (T_{O1} - T_V). \tag{5.2}$$

På samma sätt kan kylningen av bärlagret ses, men med skillnaden att flödet är noll för bärlagret. För oljan fås då

$$V_{O2} \frac{dT_{O2}}{dt} = f_O (T_{Oin2} - T_{O2}) + \frac{\mu_2 A_2}{c_{pO} \rho_O} (T_B - T_{O2}) \tag{5.3}$$

och för bärlagret fås

$$V_B \frac{dT_B}{dt} = -\frac{\mu_2 A_2}{c_{pB} \rho_B} (T_B - T_{O2}) + \alpha \tag{5.4}$$

där α (K m^3/s) är en konstant som beskriver uppvärmningen av bärlagret då turbinen körs. Den beror egentligen på lasten (det vill säga uttagen effekt), men kan ses som konstant då effekten varierar långsamt. De samband som kan urskiljas ur ekvation (5.1) - (5.4) och figur 15 är att $T_{Oin1} = T_{O2}$ och $T_{Oin2} = T_{O1}$.

5.2.3 Fas 3 – Tillståndsframställningen

Anta nu att oljeflödet f_O är konstant och lika med f och se vattenflödet och vattentemperaturen som insignaler: $u_1 = f_V$ och $u_2 = T_{Vin}$. För att underlätta tillståndsframställningen antas en till insignal $u_3 = 1$. Denna hålls konstant och multipliceras med tillförd värme, α, till bärlagret. Utsignalen $y = T_B$. Eftersom areorna A_1, A_2 och volymerna V_V, V_{O1}, V_{O2}, V_B (m^3) är okända ansätts $\beta_1 = 1/c_{pV}\rho_V$ (känd) och så vidare. Sedan väljs tillståndsvariablerna $x_1 = T_V$, $x_2 = T_{O1}$, $x_3 = T_{O2}$ och $x_4 = T_B$, och då fås tillståndsframställningen

$$
\begin{aligned}
\dot{x}_1 &= \frac{u_1}{V_V}(u_2 - x_1) + \frac{\beta_1\mu_1 A_1}{V_V}(x_2 - x_1) = \frac{u_1 u_2}{V_V} - \frac{u_1 + \beta_1\mu_1 A_1}{V_V}x_1 + \frac{\beta_1\mu_1 A_1}{V_V}x_2 \\
\dot{x}_2 &= \frac{f}{V_{O1}}(x_3 - x_2) - \frac{\beta_2\mu_1 A_1}{V_{O1}}(x_2 - x_1) = \frac{\beta_2\mu_1 A_1}{V_{O1}}x_1 - \frac{(f + \beta_2\mu_1 A_1)}{V_{O1}}x_2 + \frac{f}{V_{O1}}x_3 \\
\dot{x}_3 &= \frac{f}{V_{O2}}(x_2 - x_3) + \frac{\beta_3\mu_2 A_2}{V_{O2}}(x_4 - x_3) = \frac{f}{V_{O2}}x_2 - \frac{(f + \beta_3\mu_2 A_2)}{V_{O2}}x_3 + \frac{\beta_3\mu_2 A_2}{V_{O2}}x_4 \\
\dot{x}_4 &= -\frac{\beta_4\mu_2 A_2}{V_B}(x_4 - x_3) + \frac{\alpha}{V_B}u_3 = \frac{\beta_4\mu_2 A_2}{V_B}x_3 - \frac{\beta_4\mu_2 A_2}{V_B}x_4 + \frac{\alpha}{V_B}u_3 \\
y &= x_4
\end{aligned}
\qquad (5.5)
$$

där $A_1\mu_1 = d_1$, $A_2\mu_2 = d_2$, $1/V_V = c_1$, $1/V_{O1} = c_2$, $1/V_{O2} = c_3$ och $1/V_B = c_4$. Därefter ska systemet ses på tillståndsform:

$$
\begin{aligned}
\dot{x}(t) &= \begin{bmatrix} -(u_1 + \beta_1 d_1)c_1 & \beta_1 c_1 d_1 & 0 & 0 \\ \beta_2 c_2 d_1 & -(f + \beta_2 d_1)c_2 & fc_2 & 0 \\ 0 & fc_3 & -(f + \beta_3 d_2)c_3 & \beta_3 c_3 d_2 \\ 0 & 0 & \beta_4 c_4 d_2 & -\beta_4 c_4 d_2 \end{bmatrix} x(t) + \begin{bmatrix} c_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \alpha c_4 \end{bmatrix} \begin{bmatrix} u_1(t)u_2(t) \\ u_3(t) \end{bmatrix} \\
y(t) &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} x(t)
\end{aligned}
\qquad (5.6)
$$

Ur ekvation (5.6) ses att tillståndsekvationen har en tidsvarierande variabel i form av u_1. Detta innebär att det är en olinjär modell, även kallad bilinjär. Denna form av modell komplicerar systemidentifieringen. En metod att lösa detta på är att se på systemet då det är stationärt (se avsnitt 5.2.4). Tillståndsekvationen har också ett antal okända parametrar som måste bestämmas. För identifierbarhet och observerbarhet se appendix 4.

5.2.4 Stationära förhållanden

Då processen är stationär, det vill säga då inga förändringar av temperaturer sker fås följande enligt (5.5):

$$V_V \frac{dT_V}{dt} = 0 \Rightarrow u_1(u_2 - T_V) + \frac{\mu_1 A_1}{c_{pV}\rho_V}(T_{O1} - T_V) = 0 \tag{5.7}$$

$$V_{O1} \frac{dT_{O1}}{dt} = 0 \Rightarrow f(T_{O2} - T_{O1}) - \frac{\mu_1 A_1}{c_{pO}\rho_O}(T_{O1} - T_V) = 0 \tag{5.8}$$

$$V_{O2} \frac{dT_{O2}}{dt} = 0 \Rightarrow f(T_{O1} - T_{O2}) + \frac{\mu_2 A_2}{c_{pO}\rho_O}(T_B - T_{O2}) = 0 \tag{5.9}$$

$$V_B \frac{dT_B}{dt} = 0 \Rightarrow -\frac{\mu_2 A_2}{c_{pB}\rho_B}(T_B - T_{O2}) + \alpha = 0 \tag{5.10}$$

Vidare fås ur (5.10) att

$$T_{O2} = T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} \tag{5.11}$$

Genom att sätta in (5.11) i (5.9) erhålls

$$f(T_{O1} - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2}) + \frac{\mu_2 A_2}{c_{pO}\rho_O}(T_B - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2}) = 0$$

$$T_{O1} - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} = -\frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f}$$

$$T_{O1} = T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} - \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} \tag{5.12}$$

Insättning av (5.11) och (5.12) i (5.8) ger

$$f(T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} + \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f}) - \frac{\mu_1 A_1}{c_{pO}\rho_O}(T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} - \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} - T_V) = 0$$

$$\frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O} = \frac{\mu_1 A_1}{c_{pO}\rho_O}(T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} - \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} - T_V)$$

$$T_V = T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} - \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} - \frac{\alpha c_{pB}\rho_B}{\mu_1 A_1} \tag{5.13}$$

Slutligen sätts uttrycken i (5.12) och (5.13) i (5.7) in

$$u_1(u_2 - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} + \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} + \frac{\alpha c_{pB}\rho_B}{\mu_1 A_1}) +$$

$$+ \frac{\mu_1 A_1}{c_{pV}\rho_V}(T_B - \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} - \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} + \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} + \frac{\alpha c_{pB}\rho_B}{\mu_1 A_1}) = 0$$

$$u_2 - T_B + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} + \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} + \frac{\alpha c_{pB}\rho_B}{\mu_1 A_1} = -\frac{\alpha c_{pB}\rho_B}{c_{pV}\rho_V u_1}$$

$$T_B = u_2 + \frac{\alpha c_{pB}\rho_B}{c_{pV}\rho_V u_1} + \frac{\alpha c_{pB}\rho_B}{\mu_2 A_2} + \frac{\alpha c_{pB}\rho_B}{c_{pO}\rho_O f} + \frac{\alpha c_{pB}\rho_B}{\mu_1 A_1} \qquad (5.14)$$

Ekvation (5.14) ger ett uttryck för bärlagertemperaturen uttryckt i insignalerna u_1 och u_2. Eftersom övriga parametrar i ekvation (5.14) är konstanter kan (5.14) skrivas som

$$T_B(t) = u_2(t) + k_1 u_1^{-1}(t) + k_2 \qquad (5.15)$$

Detta resulterar i att det för stationära förhållanden erhålls en enkel statisk modellstruktur. Observera att detta gäller för den kontinuerliga modellen. Det är dock mycket enkelt att göra om till diskret form. Sätt $T_B(t) = T_B(k)$ och $u_1(t)$, $u_2(t) = u_1(k)$, $u_2(k)$, där k är samplingstillfället.

5.3 SYSTEMIDENTIFIERING

Det intressanta är nu att estimera de okända parametrarna i ekvation (5.6) eller (5.15) beroende på vilken modell som är lättast att använda och ger bäst resultat. Vilka parametrar som är okända behandlas i appendix 5. Identifieringen görs utifrån uppmätta data. De data som används är insignalerna kylvattenflöde och temperatur, och utsignalen bärlagertemperatur (se kapitel 4). Från ekvation (5.6) fås en tillståndsekvation som skall identifieras. Här används grey-box identifiering med Kalmanfilter. Ekvation (5.15) ger statisk FIR-modell.

5.4 LARMHANTERING

Enligt avsnitt 3.4 finns det två olika fall av fel, långsamma och abrupta. Dessa fel behöver inte identifieras på samma sätt utan det kan användas två olika metoder. Däremot baseras larmhanteringen på samma sak, se figur 18. Först beräknas en skattad bärlagertemperatur som sedan jämförs med den riktiga. Skulle skillnaden (felet) avvika med mer än ett visst gränsvärde ges larm, annars inte.

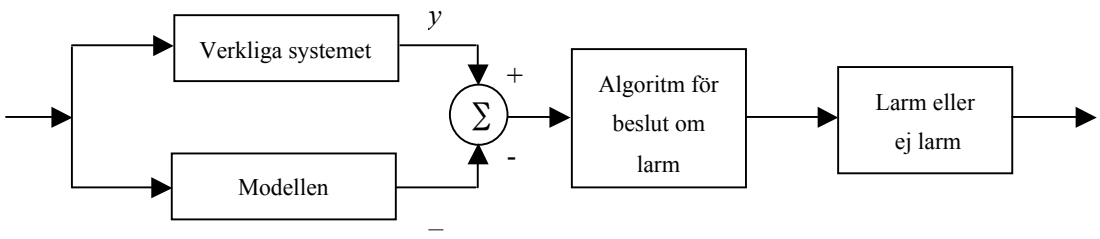


Fig. 18 Schematisk bild över larmstruktur.

5.4.1 Detektering av långsamma fel

Eftersom modellen endast beror på nuvärden för flöde och temperatur är det ingen påverkan på den skattade bärlagertemperaturen från tidigare värden. Detta gör att det skulle kunna tillämpas samma metod för detektering av långsamma fel som för abrupta fel. Det kan dock vara så att abrupta fel skall upptäckas tidigare än vad långsamma fel görs. Ett långsamt fel skulle kunna få pågå ett bra tag innan det åtgärdas. Hur en sådan metod skulle kunna se ut är att det sparas undan gamla värden och jämförs med nya. Kan då en trend urskiljas som visar på att temperaturen långsamt ökar ges ett larm.

Eftersom att det inte finns några dokumenterade långsamma fel, måste ett antagande göras över hur ett långsamt fel ser ut. Ett långsamt fel skulle till exempel kunna uppstå som en ramp. Den metod som kommer att användas i det här arbetet är en jämförelse mellan skattad och riktig bärlagertemperatur på samma sätt som för abrupta fel (se nedan).

5.4.2 Detektering av abrupta fel

Feldetekteringen av abrupta fel baseras på CUSUM-algoritmen. Även för abrupta fel saknas dokumentation över hur det uppstår. Här måste alltså ett antagande göras för hur felutvecklingen ser ut. Som nämndes i kapitel 3 finns det två typer av abrupta fel. Det ena är en förändring av medelvärdet och den andra är en förändring av standardavvikelsen. Dessa fel är sådana som uppstår då en förändring i lagret sker. Mätfel som kan uppstå i en givare måste även identifieras för att undvika larm. Ett mätfel kan uppstå som en impuls eller som en kortvarig förändring av mätvärdet. Skulle förändringen bestå utan att den riktiga temperaturen ändrats skall ett larm ändå avges för att åtgärda felet.

5.4.3 Hantering av maximal bärlagertemperatur

Skulle det vara så att den maximala bärlagertemperaturen uppnås måste det finnas en larmhantering av detta. Det första som skall göras vid feldetekteringen är att kontrollera att den maximala bärlagertemperaturen inte är överskriden. Även här bör det tillåtas att flera sampel kan överskrida larmgränsen innan larm ges för att undvika falsklarm. För Söderfors är den maximala temperaturen 46°C.

5.5 INVERKAN AV REGLERINGEN AV KYLVATTENFLÖDET PÅ RESULTATEN

Eftersom det i Söderfors kraftverk är reglerat kylvattenflöde är flödet bestämt genom återkoppling från bärlagertemperaturen. Detta innebär problem när de okända parametrarna ska identifieras. Betrakta en enkel ARX-modell

$$y(t) + ay(t-1) = bu(t-1) + e(t) \tag{5.16}$$

Antag att systemet regleras med en P-regulator under datainsamlingen

$$u(t) = -fy(t) \tag{5.17}$$

Prediktorn är alltså

$$\hat{y}(t \mid \theta) = -ay(t-1) + bu(t-1) = (-bf - a)y(t-1) \tag{5.18}$$

Alla värden på modellparametrarna, a och b, sådana att $bf + a$ är ett visst givet värde, ger alltså identiska prediktioner under återkopplingen. Det finns alltså inga möjligheter att bestämma a och b då det är återkoppling, och detta oavsett vad det sanna systemet är (Glad & Ljung, 1991).

Resultatet av detta är att det är svårt att få fram någon bra modell över systemet. De parametrar som modellen får kan gälla mycket bra för data som används vid estimeringen, men kan ge att helt annat resultat för nya data. För att undvika detta bör data samlas in utan återkoppling, eller att försöka ändra på börvärdena så mycket som möjligt.

6 GENOMFÖRANDE

6.1 SYSTEMIDENTIFIERING

6.1.1 Förbehandling av data

För att kunna identifiera en modell måste data förbehandlas. I kapitel 4 gjordes en grov förbehandling där data anpassades till användbart format och till korrekt samplingsintervall, det vill säga en gång per minut. De dataserier som där erhålls måste i sin tur behandlas. Först måste dataserien betraktas (fig. 19) för att se att den inte innehåller några ”outliers”, det vill säga värden som avviker stort från medelvärdet. De ”outliers” som finns ersätts med ett rimligt värde. Därefter måste serierna anpassas så att alla mätvariabler har lika långa serier och att alla onormala sekvenser är borta (fig. 20). Onormala sekvenser kan vara då generatorn varit avstängd eller avbrott gjorts i mätningarna. Om den fysiska nivån på mätvärdena inte spelar någon roll kan medelvärdet tas bort från varje serie. Detta har gjorts för de statiska modellerna. Sedan skall dataserien delas upp i två intervall, ett med kalibreringsdata och ett med valideringsdata (fig. 21). Detta för att se om den framtagna modellen även gäller för en helt ny datasekvens.

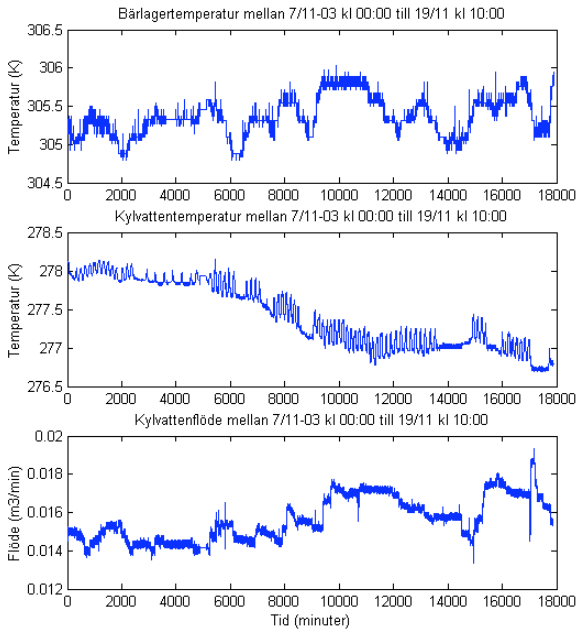


Fig. 19 Figurer över de dataserier som använts för systemidentifieringen. Insamlade data sträcker sig från tiden 7/11-03 kl 00.00 till 19/11-03 kl 10.00.

I *Matlab* finns en toolbox som heter *System Identification Toolbox*. Den har använts för all programmering eftersom den stödjer de flesta funktioner som är användbara inom systemidentifiering. Bland annat har den ett grafiskt användargränssnitt, *Ident*, som kan användas. Det grafiska användargränssnittet är lätt att använda och är mycket överskådligt.

Mängden data har slutligen blivit mycket liten (ungefär en vecka), vilket kommer att påverka resultatet negativt.

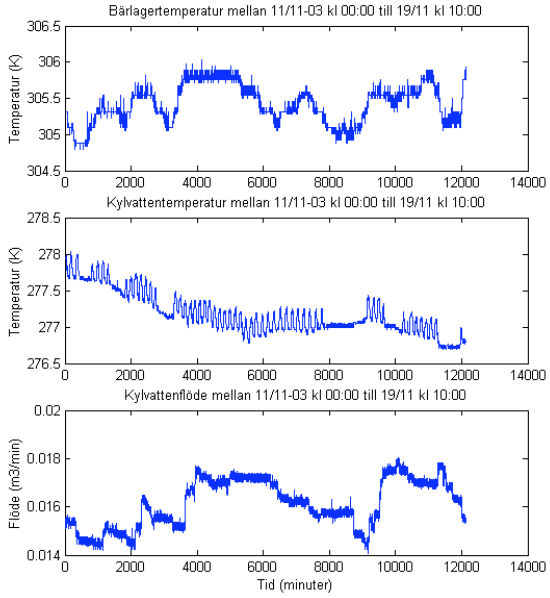


Fig. 20 Figurer över de förbehandlade dataserier som använts för systemidentifieringen. Förbehandlade data sträcker sig från tiden 11/11-03 kl 00.00 till 19/11-03 kl 10.00.

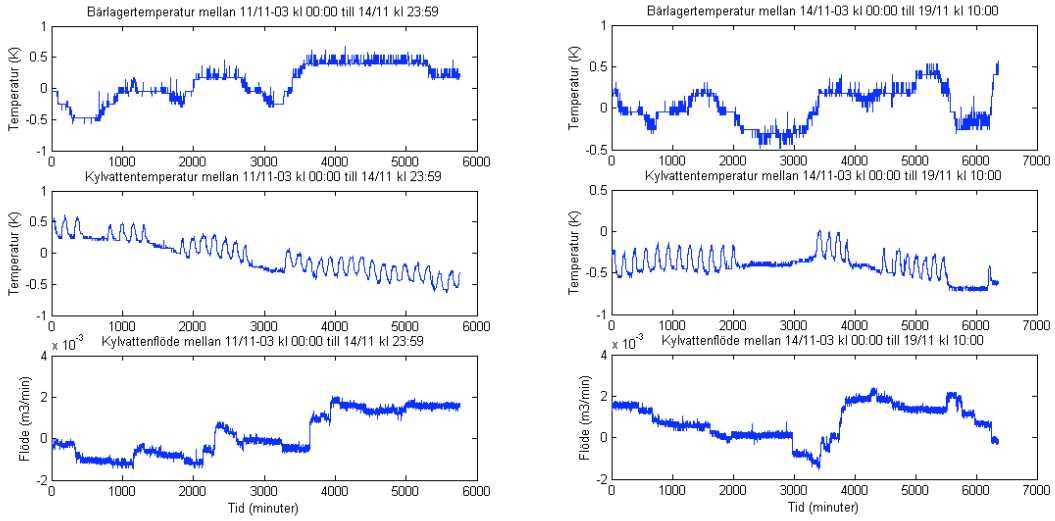


Fig. 21 Figurer över kalibreringsdata och valideringsdata. Kalibreringsdata sträcker sig över tiden 11/11 kl 00:00 till 14/11 kl 23:59 och valideringsdata från 15/11 kl 00:00 till 19/11 kl 10:00.

6.1.2 Identifiering

6.1.2.1 Tillståndsmodellen

Efter att data förbehandlats kan själva identifieringen göras. Detta gjordes med *Matlab* och *System Identification Toolbox*. Det första som gjordes var att testa tillståndsmodellen. Den innehåller åtta okända parametrar som skall identifieras (se ekvation 5.22). Identifieringen gjordes med programmet *tillstandest.m* (se appendix 9) som tar mycket lång tid att köra. Estimeringsprocessen beror mycket på hur bra initialvärdena på θ-parametrarna väljs. Väljs dessa för stora eller för små tar det lång tid att hitta bra värden eller så fås helt enkelt fel värden. Därför är det viktigt att kunna ”gissa” bra initialvärden. Även initialvärden på tillstånden $\dot{x}(t)$ behöver ges. Dessa är dock enklare att uppskatta utifrån mätdata. För att kunna uppskatta storleken på θ-parametrarna utgås det från tidigare beteckningar, area, volym och värmegenomgångskoefficient, se appendix 5.

6.1.2.2 Statisk modell

Eftersom tillståndsmodellen har så mycket som åtta okända parametrar, kan den vara svår att estimera. Som en jämförelse jämförs även det stationära fallet, där det är en statisk modell

$$y = u_2 + \theta_1/u_1 + \theta_2 \tag{6.1}$$

För estimering av parametrarna θ_1 och θ_2 har programmet *statisk_est.m* använts (se appendix 10). Denna skall även jämföras mot vanliga konfektionsmodeller av typ ARX, OE och ARMAX. För jämförelse av ARX, OE och ARMAX-modellerna användes *Ident*. De erhållna modellerna jämfördes med valideringsdata för några olika valideringsmetoder.

6.2 LARMHANTERING

När ett fel detekterats måste det analyseras för att avgöra ifall ett larm skall avges eller ej. Eftersom det inte finns någon tillgänglig information över hur ett felutvecklingsförlopp ser ut måste antaganden göras över hur fel utvecklas. De fel som används i simuleringsstudien är:

- Långsamma fel: Bärlagertemperaturen ökar mycket långsamt under en lång tid, till exempel 0,001-0,01ºC/min.
- Abrupta fel: Bärlagertemperaturen ökar snabbt med till exempel ett steg eller att standardavvikelsen på bruset ökar markant.
- Tillfälliga fel: Fel som beror på mätfel eller givarfel. Mätfel som pågår under kort tid skall hanteras som falsklarm och inte larma. Givarfel hanteras på samma sätt, fast om det är ett permanent fel på givaren kommer larm att avges efter ett tag.

Inverkan av återkopplingen komplicerar feldetekteringen något. Utan återkoppling skulle en ökning kunna ses av bärlagertemperaturen som är oberoende av kylvattenflöde och temperatur. Denna kan lätt detekteras då temperaturen avviker från den estimerade. När det är återkoppling på flödet fås en ökning av flödet då temperaturen i bärlagret ökar. Ökning av flödet motverkar då att temperaturen i bärlagret avviker från det normala ända tills maximalt flöde uppnåtts. För att så tidigt som möjligt upptäcka felet måste hänsyn tas till flödet. Det som händer är att den skattade temperaturen kommer att bli mycket lägre/högre (beroende på modellens utformning) än den riktiga (då den kyls mycket). Här kan alltså en negativ skillnad (riktig bärlagertemperatur är lägre än skattad bärlagertemperatur) ses på temperaturen. Denna kan upptäckas om en dubbelsidig CUSUM-algoritm används som detekterar både förändringar då riktig bärlagertemperatur överstiger estimerad och tvärt om.

En kontroll på om bärlagertemperaturen överstiger den maximalt tillåtna måste även införas i feldetekteringen för att säkerställa att temperaturen inte blir för hög.

För att testa larmhanteringen och bestämma larmparametrarna v, h och antalet fel som tillåts innan larm (*run test*) skrevs först ett program i Matlab (*cusumtest.m*, se appendix 11). Här testades olika värden på parametrarna och jämfördes med hänsyn till tiden för larm och antalet falsklarm.

6.3 SYSTEMSTRUKTUR OCH IMPLEMENTERING I DELPHI

För att kunna tillämpa det erhållna feldetekteringssystemet skrevs ett program i *Delphi* (se appendix 12). Programmet kommunicerar med en kraschdumpdatabas (fig. 22). Kraschdumpsdatabasen är en applikation med svartlåda funktion och som utvecklades under ett tidigare examensarbete. Den sparar undan samplande data från Conwides koncentrator i Söderfors till en lokal databas. Applikationen övervakar sedan ständigt parametrar som är avgörande för att detektera driftstörningar. Om en driftstörning

inträffar kopieras denna databas och sparas undan till en katalog med kraschdumpar för framtida analys.

Efter beräkningar i *Delphi* sparas värden undan i en separat databas som innehåller samtliga beräknade och uppmätta värden. Dessutom sparar den undan eventuella värden och tidpunkter för larm. I framtiden är det tänkt att databasen ska finnas tillgängligt över Internet och presentera trendkurvor samt varningsindikator då larm inträffar för kraftverkspersonal. Andra funktioner som kan finnas är inställning av till exempel larmparametrar (endast för dem som har behörighet).

Programmet i *Delphi* hämtar data ifrån kraschdumpdatabasen där nuvärden för signalerna samlas (fig. 23). Efter inläsningen kontrollerar programmet om maximal bärlagertemperatur har uppnåtts. Om inte beräknas en skattad bärlagertemperatur som jämförs med de uppmätta. Ifall CUSUM-algoritmen ger ett fel sparas tidpunkt och värden i en larmdatabas. Inget nytt larm kan ges förrän temperaturen är normal eller maximal bärlagertemperatur uppnåtts. Om effekten är noll görs ingenting. Därefter sparas alla mätvärden och beräknade värden samt eventuella larm undan i en ny databas. Denna kan användas för att studera trendkurvor i till exempel Excel.

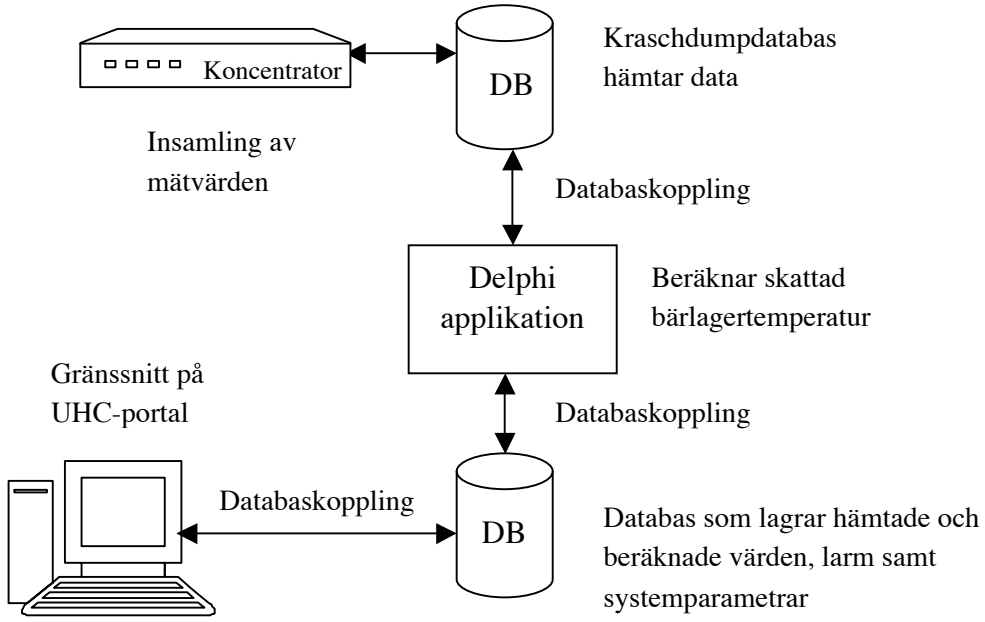


Fig. 22 Figur över feldetekteringssystemets struktur.

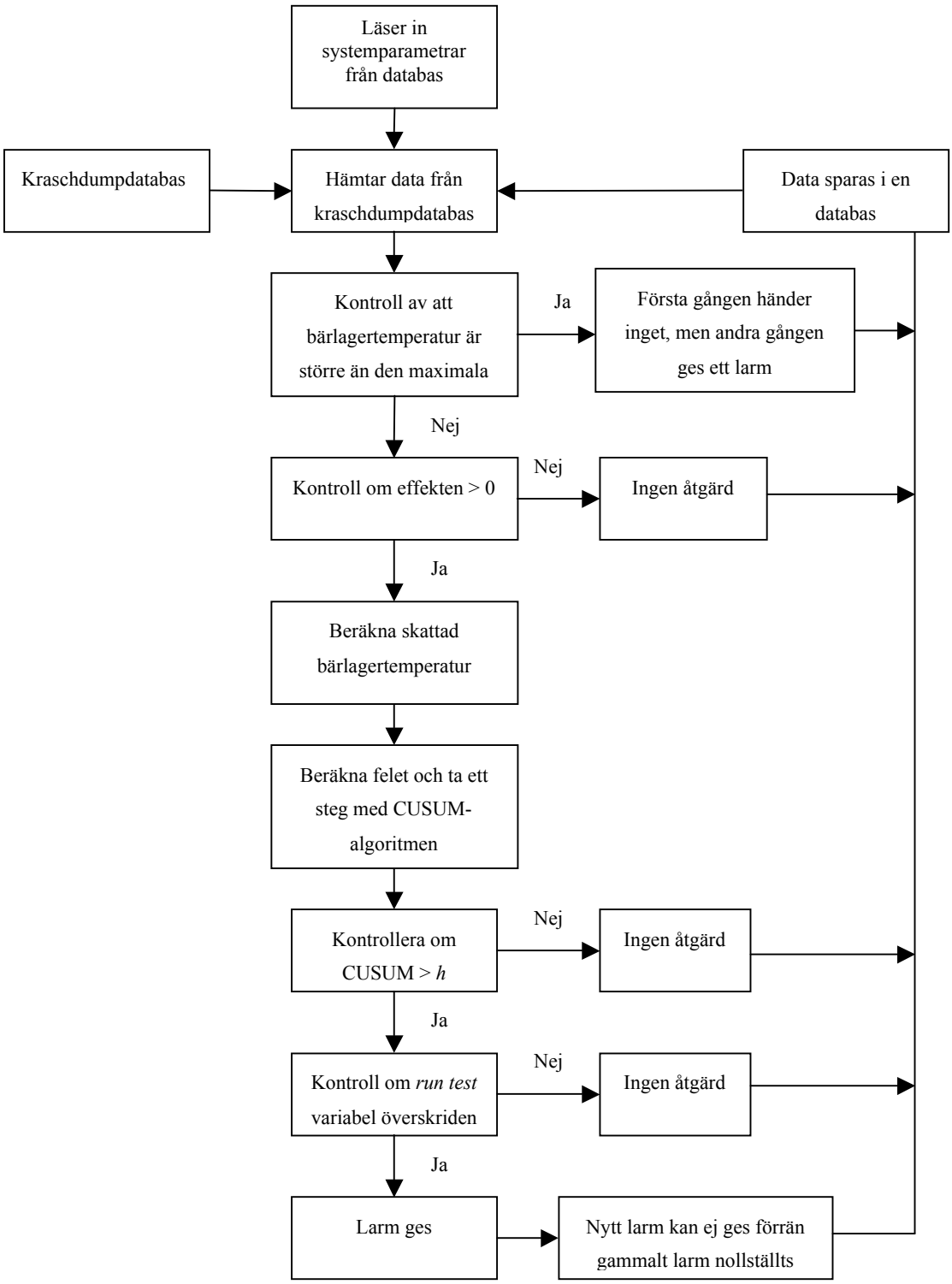


Fig. 23 Beskrivning av programstruktur.

7 RESULTAT

7.1 SYSTEMIDENTIFIERING – MODELLVALIDERING

7.1.1 Tillståndsmodellen

Den erhållna tillståndmodellen gav ett otillräckligt resultat då modellen inte följde valideringsdata och därför lades ett Kalmanfilter på för uppdatering av tillstånden $\dot{x}(t)$. Nu fick modellen en mycket bra prediktion mot valideringsdata (fig. 24). Däremot kan inte denna modell användas för feldetekteringen då den även kommer att följa felutvecklingen (fig. 24) och därför inte kan larma då det är avvikelser i bärlagertemperaturen. Gäller speciellt för långsamma avvikelser.

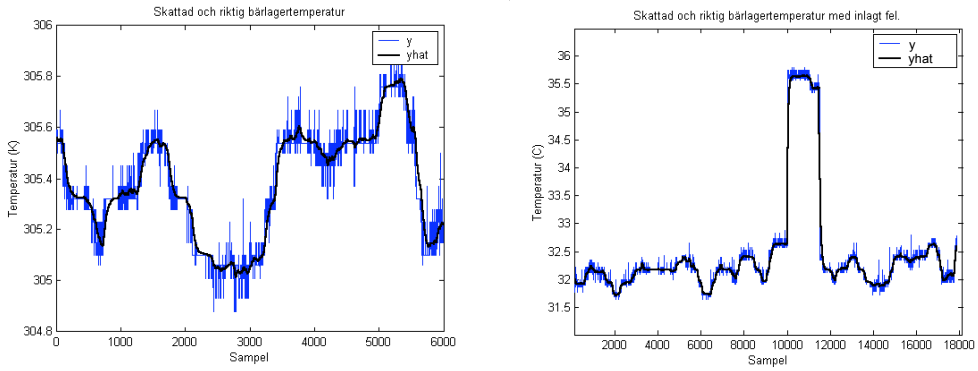


Fig. 24 Tillståndsmodellen med kalmanfilter. Den vänstra figuren visar valideringsdata och den högra orginaldata med ett inlagt fel vid 10 000 sampel. Den tjocka linjen är skattad bärlagertemperatur och den tunna (brusiga) är uppmätt bärlagertemperatur.

7.1.2 Statisk modell

För det stationära fallet är det en statisk modell enligt ekvation 6.1. Där är det två okända parametrar, nämligen θ_1 och θ_2. θ_2 kan ses som medelvärdet av den riktiga utsignalen y, vilket ger $\theta_2 = 32{,}21$. Då återstår bara en okänd parameter att estimera, det vill säga θ_1. Ur tabell 1 ses att den statiska modellen inte riktigt ger ett tillfredsställande resultat eftersom maximala prediktionsfelet uppgår till 1,1ºC.

Tab. 1 Tabell över valideringsinformation för tillståndsmodellen och den statiska modellen baserad på stationära förhållanden.

Modell	Residualanalys	Maximala prediktionsfelet
Tillståndsmodell – med Kalman	18	0,30
Statisk modell	1344	1,12

Istället tittas det på en statisk modell med tre okända parametrar, θ_1, θ_2 och θ_3, (STAT_1) med information från den stationära modellen, det vill säga med θ_1/u_1, och θ_2 är fortfarande medelvärdet av y

$$y = \theta_1/u_1 + \theta_3 * u_2 + \theta_2 \tag{7.1}$$

Då fås ett bättre resultat med avseende på maximala prediktionsfelet, residualanalys och AIC. Modellen har heller ingen dynamik eller tidsfördröjning. Med dynamik menas att det tas hänsyn till tidigare mätdata för att få en jämnare prediktion och mindre känslighet för variationer i mätsignalerna. Ett annat alternativ är att titta på en statisk modell med $\theta_1 * u_1$ istället för θ_1/u_1 (STAT_2). Residualanalysen och AIC blev nu bättre, då de fick lägre värden.

Tab. 2 Tabell över statiska modeller.

Modell	Residualanalys	Max. pred. fel	AIC
Statisk modell	1344	1,12	0,280
STAT_1	758	0,79	0,053
STAT_2	549	1,01	0,036

Ur tabell 2 kan det konstateras att STAT_1 och STAT_2 är jämbördigt lika med fördel för STAT_2. För att ytterligare se om modellen kan förbättras läggs dynamik och tidsfördröjningar till modellen. Först tittas det på STAT_1 (se tabell 3). Ur tabellen ses att dynamik på insignalen u_1 ger det bästa resultatet eftersom den ger lägre värde på residualanalysen, men inte tillräckligt mycket för att motivera att använda den modellen. Samma sak gäller även för STAT_2 (se tabell 4).

Tab. 3 Tabell över STAT_1 med dynamik på insignalerna och tidsfördröjningar, där första kolumnen motsvarar *na* [nb_1 nb_2] [nk_1 nk_2].

STAT_1	Residualanalys	Max. pred. fel	AIC
0 [4 1][0 0]	747	0,79	0,053
0 [1 4][0 0]	761	0,79	0,053
0 [4 4][0 0]	748	0,79	0,053
0 [1 1] [8 0]	760	0,79	0,053
0 [1 1] [0 8]	762	0,78	0,053
0 [1 1] [8 8]	763	0,77	0,053

Tab. 4 Tabell över STAT_2 med dynamik på insignalerna och tidsfördröjningar, där första kolumnen motsvarar *na* [nb_1 nb_2] [nk_1 nk_2].

STAT_2	Residualanalys	Max. pred. fel	AIC
0 [4 1][0 0]	548	0,98	0,036
0 [1 4][0 0]	549	1,01	0,036
0 [4 4][0 0]	548	0,98	0,036
0 [1 1] [8 0]	559	1,00	0,037
0 [1 1] [0 8]	551	1,01	0,036
0 [1 1] [8 8]	559	1,00	0,037

Även några ARX, ARMAX- och OE-modeller testades i *Ident*, men ingen av dem gav något resultat som var bättre är de statiska modellerna.

7.1.3 Resultat

Från ovanstående fås att både STAT_1 och STAT_2 ger liknande resultat i valideringen. Eftersom endast en modell ska användas jämförs kurvorna mot varandra. Ur figur 25 och 26 ses att STAT_2 följer verklig data på ett mer följsamt sätt. Därför väljs den modellen till den slutgiltiga.

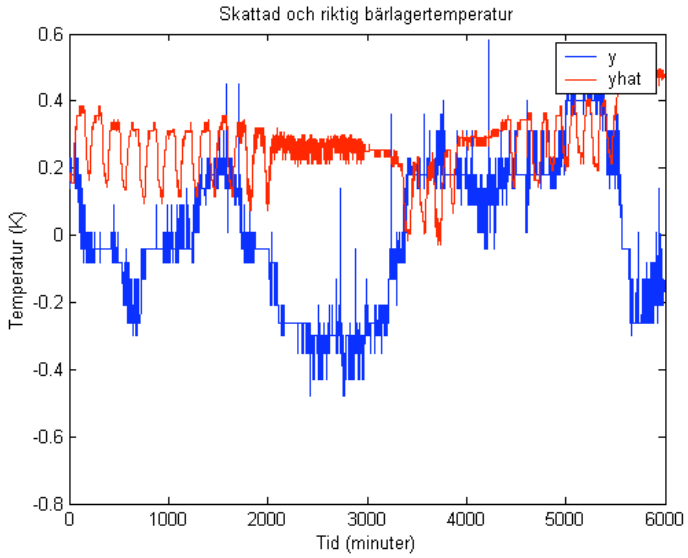


Fig. 25 Skattad (blå) och uppmätt (röd) bärlagertemperatur med STAT_1.

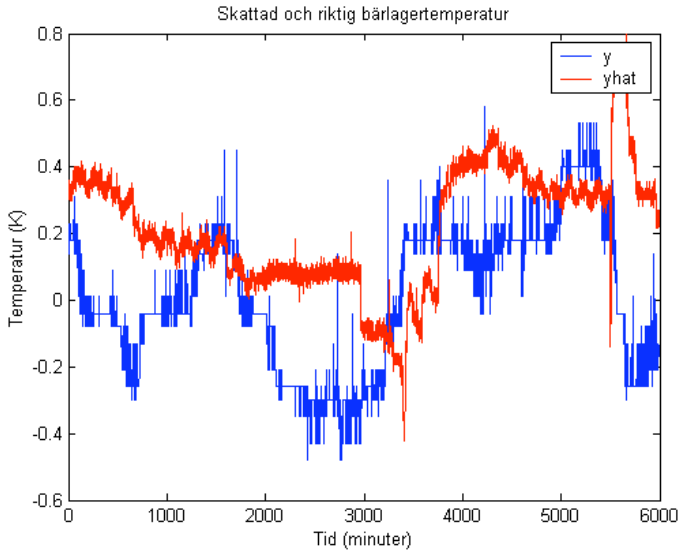


Fig. 26 Skattad (blå) och uppmätt (röd) bärlagertemperatur med STAT_2.

Den resulterande modellen är:

$$y = 187{,}4*u_1 - 0{,}1535*u_2 + 32{,}21 \quad (7.2)$$

där u_1 är kylvattenflödet dividerat med 1000 (för att få i enheten m^3/min) minus medelvärdet av kylvattenflödet ur dataserien i figur 21, vilket är 0,01564 m^3/min. u_2 är kylvattentemperaturen minus medelvärdet av kylvattentemperaturen ur dataserien i figur 21, vilken är 4,27°C och y är den skattade bärlagertemperaturen i °C.

För implementering i ett feldetekteringssystem skall modellen skrivas som:

$$y = 187{,}4*(u_1/1000 - 0{,}01564) - 0{,}1535*(u_2 - 4{,}27) + 32{,}21 \quad (7.3)$$

I ekvationen är det ett omvänt beroende på kylvattenflöde och kylvattentemperatur. Egentligen borde det vara ett minustecken framför flödet och plustecken framför temperaturen. Anledningen till att det inte är det är förmodligen att den datamängd som användes inte var tillräckligt exciterat.

7.2 LARMHANTERING

Eftersom det är parametrarna h och v i CUSUM-algoritmen som påverkar feldetekteringen har de varierats för att se hur de påverkar antalet larm på dataserierna. Det ska inte vara några larm eftersom det inte finns några fel i dataserierna från början. Som ses ur tabell 5 minskar antalet larm då parametrarna h och v ökas. Eftersom det inte ska vara några larm i onödan sätts larmgränsen v till 2ºC (den temperatur där avvikelsen räknas som tillräckligt stor för att ett fel skall ha inträffat). Som nämndes i kapitel 6 studeras tre olika fel. Resultaten från dessa och en sammanställning presenteras nedan i separata avsnitt. Den modell som används för simuleringen är modellen i ekvation (7.3).

Tab. 5 Tabell över hur parametrarna h och v påverkar antalet larm.

h	***v***	**Antalet larm**
0,1	0,5	45
1	0,5	4
10	0,5	0
0,1	1	0
0,1	2	0

7.2.1 Detektering av långsamma fel

Långsamma fel antas uppstå som en temperaturökning på 0,01ºC/min under ett dygn. Två metoder har testats, den ena där inverkan av återkopplingen är liten till exempel om det är lång tidsfördröjning mellan bärlagertemperatur och flöde, den andra då inverkan av återkopplingen är stor. Parametern *v* är konstant eftersom det är den temperaturavvikelsen som är intressant. Parametern *h* har varierats för att se skillnaden på tiden till larm mellan olika värden. Ur tabell 6 ses att det skiljer 40 min mellan $h = 0{,}1$ och $h = 10$. Vilket värde på *h* som väljs kanske inte har så stor betydelse för långsam feldetektering då felutvecklingstiden ändå är lång. Samma resultat ses även i tabell 7 där det är inverkan av återkoppling på flödet, det vill säga att det är flödet som ökar då bärlagertemperaturen ökar. Det kan även konstateras att det blir ett larm då temperaturskillnaden strax överstigit 2ºC.

Tab. 6 Tabell över hur parametern *h* påverkar tiden för upptäckt av larm utan återkoppling.

h	*v*	Tid till upptäckt (min)	Felet (ºC)
0,1	2	230	2,04
1	2	239	2,14
10	2	270	2,44

Tab. 7 Tabell över hur parametern *h* påverkar tiden för upptäckt av larm med återkoppling.

h	*v*	Tid till upptäckt (min)	Felet (ºC)
0,1	2	214	2,00
1	2	224	2,14
10	2	254	2,47

7.2.2 Detektering av abrupta fel

Abrupta fel utvecklas på två olika sätt, som ett steg eller som en ökning av standardavvikelsen. Dessa har testats både med och utan inverkan av återkoppling. För steg antas bärlagertemperaturen plötsligt öka med 3ºC. Som ses ur tabell 8 och 9 upptäcker algoritmen felet tidigt för $h = 0{,}1$ och 1, men efter ungefär 10 min för $h = 10$. Detta fel skall upptäckas så tidigt som möjligt för att snabbt kunna åtgärdas.

Tab. 8 Tabell över hur parametern *h* påverkar tiden för upptäckt av larm utan återkoppling.

h	*v*	Tid till upptäckt (min)
0,1	2	1
1	2	2
10	2	14

Tab. 9 Tabell över hur parametern *h* påverkar tiden för upptäckt av larm med återkoppling.

h	*v*	Tid till upptäckt (min)
0,1	2	1
1	2	1
10	2	10

För ökning av standardavvikelsen på bruset antas att den ökar 7 ggr. I tabell 10 och 11 ses att tiden ökar markant då $h = 10$.

Tab. 10 Tabell över hur parametern *h* påverkar tiden för upptäckt av larm utan återkoppling.

h	*v*	Tid till upptäckt (min)
0,1	2	1
1	2	7
10	2	42

Tab. 11 Tabell över hur parametern *h* påverkar tiden för upptäckt av larm med återkoppling.

h	*v*	Tid till upptäckt (min)
0,1	2	21
1	2	25
10	2	47

7.2.3 Falsklarm

Falsklarm uppstår till exempel då det är ett mätfel på bärlagertemperaturen, flödet eller kylvattentemperaturen. De mätfel som testats är då bärlagertemperaturen har fått ett alldeles för högt mätvärde, på 3 och 10 grader för högt. Dels under ett sampel och dels under 11 sampel. Som ses i tabell 12 klarar inte $h = 0{,}1$ och $h = 1$ av ett mätfel på 3 eller 10 graders skillnad. Samma sak gäller för 11 sampel , men då ger även $h = 10$ larm (se tabell 13). För att det inte ska bli något larm då det är mätfel införs ett så kallat *run test*, där det krävs att flera fel ska upptäckas innan det blir ett larm. I tabell 14 ses att då en *run test* variabel införs förlängs tiden som ett mätfel får pågå.. Med ett värde på 3 kan 4 sampel på 10 graders avvikelse accepteras innan ett larm ges.

Tab. 12 Tabell över hur parametern *h* påverkar tiden för beslut av larm för mätfel under ett sampel.

h	***v***	**Tid till upptäckt för 3 respektive 10 graders skillnad (min)**
0,1	2	0 respektive 0
1	2	0 respektive 0
10	2	Ej upptäckt

Tab. 13 Tabell över hur parametern *h* påverkar tiden för beslut av larm för mätfel under 11 sampel.

h	***v***	**Tid till upptäckt för 3 respektive 10 graders skillnad (min)**
1	2	0 respektive 0
10	2	7 respektive 1

Tab. 14 Tabell över hur parametern *h* påverkar tiden för beslut av larm för mätfel under 11 sampel med *run test*.

h	***v***	**Tid till upptäckt för 3 respektive 10 graders skillnad (min)**	**Run test variabel**
1	2	0 respektive 3	2
1	2	0 respektive 5	3

7.3 RESULTAT

Ur resultaten ovan kan det konstateras att den erhållna modellen klarar av att beskriva bärlagertemperaturen för de befintliga dataserierna. Hur modellen kommer att klara av att beskriva temperaturen då det är högre/lägre vattentemperatur och högre/lägre vattenflöde kan det inte sägas något om eftersom data för det inte finns tillgängligt. Innan implementering behöver detta testas, se beskrivning appendix 8.

Vad gäller feldetekteringen klarar modellen och algoritmen av att upptäcka inlagda fel för de olika fall av fel som antas kunna uppstå. De parametrar som valts är $h = 1$, $v = 2$ och *run test* variabel = 3. Den sistnämnda parametern påverkar tiden till larm som visats i tabell 5 till 11, men inte mer än acceptabelt. I figur 26 och 27 ses hur felen ser ut utan återkoppling respektive med återkoppling. För hantering av maximal bärlagertemperatur har *run test* variabeln satts till 2 för att undvika falsklarm på grund av mätfel.

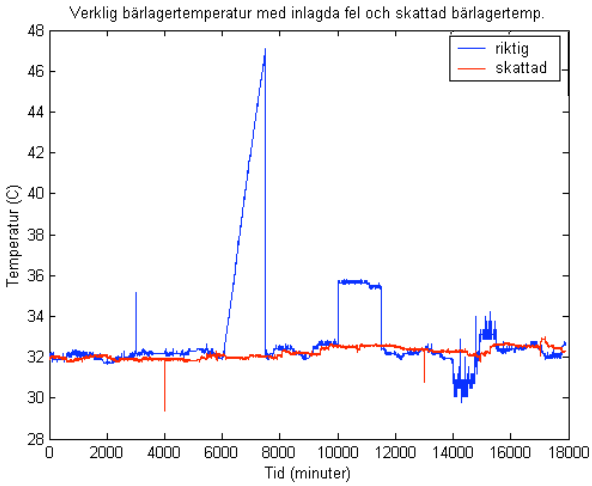


Fig. 26 Figur över skattad och uppmätt bärlagertemperatur då det är inlagda fel på bärlagertemperaturen samt mätfel på flöde och vattentemperatur, utan återkoppling.

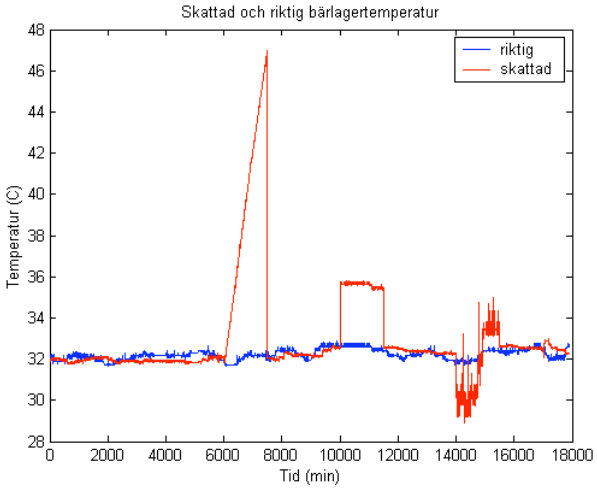


Fig. 27 Figur över skattad och uppmätt bärlagertemperatur då det är inlagda fel på bärlagertemperaturen med återkoppling.

8 DISKUSSION

I denna rapport har studerats hur bärlagertemperaturen beror av kylvattentemperaturen och kylvattenflödet. Den modell som togs fram klarade inte så väl av att beskriva bärlagertemperaturen för det intervall för vilket modellen togs fram ifrån, men ansågs ändå vara den bästa möjliga med tanke på mängden data. På grund av att mängden data var liten har det inte gått att ta fram någon välkalibrerad modell utan den framtagna modellen får ses som ett exempel på hur den framtagna modelleringsansatsen kan användas. För att få en väl fungerande modell bör data samlas in under minst ett år. Insamling kan i framtiden förmodligen göras från en databas i Råcksta då en databas med minutvärden snart kommer att finnas där.

Det studerade feldetekteringssystemet fungerar bra då den kan upptäcka både abrupta och långsamma fel. Huruvida den klarar av att undvika falsklarm får utvärderas i framtida undersökningar då det inte gått att simulera ett stopp eller typiska förlopp för olika fel.

Det framtagna feldetekteringssystemet kan enkelt implementeras i andra stationer där flödet är reglerat. En unik modell för varje kraftstations aggregat måste dock tas fram utifrån mätdata, då aggregaten och stationerna skiljer sig från varandra. För stationer utan reglerat flöde, det vill säga då flödet är konstant, borde det gå att använda samma metod fast med en insignal.

Inverkan av återkopplingen av flödet på modellen gör att den i dagsläget inte är tillförlitlig. En undersökning hur återkopplingen påverkar modellkvalitén bör göras genom att till exempel samla in data utan återkoppling för att se om någon skillnad kan urskiljas.

Inverkan av mätfel i feldetekteringssystemet gör att det lättare kan inträffa falsklarm. Enstaka mätfel kan tas om hand med ett *”run test”* som kräver att larmgränsen måste överstigas fler än en gång.

Vad gäller den fysikaliska modellen skulle en extra givare för mätning av vattentemperaturen ut ur värmeväxlaren kunna ge en enklare och eventuellt mer noggrann modell. Tas även mätningar av oljetemperaturen på ett par ställen med kan sannolikt också modellen förbättras. Dessa tillstånd blir inte längre okända och behöver då inte skattas.

8.1 FÖRSLAG PÅ FRAMTIDA ARBETEN

En ny modell över bärlagertemperaturen bör tas fram innan systemet kan börja användas av kraftverkets driftpersonal. Modellen som finns i dagsläget har begränsat giltighetsområde och när vattentemperaturen stiger till sommaren kommer modellen inte kunna beskriva bärlagertemperaturen. En ny modell skall baseras på minst ett års data. En beskrivning över hur en ny modell skall tas fram finns i appendix 8.

En alternativ metod är att mäta in- och uttemperatur på oljan från bärlagret och bärlagertemperaturen på bärlagrets ytskikt. Då erhålls mindre temperaturförluster eftersom mätningarna ligger närmare varandra. Detta borde kunna ge en säkrare modell eftersom antalet felkällor minskar. Även oljeflödet kan tas med eftersom den varierar något beroende på effekten.

9 REFERENSER

Alvarez, H., 1990. Energiteknik – del 1, Studentlitteratur, Lund.

Basseville, M., Nikiforov, V., 1993. Detection of Abrupt Changes – Theory and Application, Prentice-Hall International, Cambridge.

Dochain, D., Vanrolleghem, P., 2001. "Dynamical Modelling and Estimation in Wastewater Treatment Processes", IWA Publishing.

Glad, T., Ljung, L., 1991. Modellbygge och simulering, Studentlitteratur, Lund.

Glad, T., Ljung, L., 1997. Reglerteori – Flervariabla och olinjära metoder, Studentlitteratur, Lund.

Glemme, M., 2001. "Utveckling av larm för tekniska system i vattenkraftstationer", Examensarbete på Vattenfall Utveckling AB, Uppsala Tekniska Högskola, Uppsala, UPTEC W 01 031.

Gustafsson, F., 2000. Adaptive Filtering and Change Detection, John Wiley & Sons, England.

Kallin, K., 2003. "Reliability Centred Maintenance (RCM) inom vattenkraften med stöd av tillståndskontrollsystem", Examensarbete på Vattenfall Utveckling AB, Kungliga Tekniska Högskolan, Stockholm.

Lekby, M-E, 2001. "Beskrivning av r_ai_fil.exe", Conwide AB.

Ljung, L., 2002. "System Identification Toolbox User´s Guide", The MathWorks.

Ljung, P, 2003. Muntligt, 19/8-03. Driftpersonal Älvkarleby kraftverk.

Nordling, C., Österman, J., 1996. Physics Handbook for Science and Engineering, Studentlitteratur, Lund.

Rugh, W.J., 1996. Linear System Theory, Prentice Hall, Upper Saddle River.

Sjödin, T., 2003. "Riktlinje för larm i vattenkraftverk", PV-550/03, Vattenfall Utveckling AB, Älvkarleby.

Sjölund, M., 2002. "Ett feldetekteringssystem för bärlager i en vattenkraftstation", Examensarbete på Vattenfall Utveckling AB, Uppsala Tekniska Högskola, Uppsala, UPTEC W 02 024.

Sohlberg, B., 1998. Supervision and control for industrial processes, Springer-Verlag, London.

Söderström, T., Stoica, P., 1989. System Identification, Prentice-Hall International, Cambridge.

Vattenfall, 2003. Vattenfall AB:s hemsida, www.vattenfall.se/om_vattenfall/var_verksamhet/, datum 2003-07-01.

Weyer, E., Szederkényi, G., Hangos, K., 2000. "Grey box fault detection of heat exchangers", Control Engineering Practice 8 (2000), Elsevier science.

10 APPENDIX

Appendix 1

Sammanfattning av Minna Glemmes examensarbete ”Utveckling av larm för tekniska system i vattenkraftstationer”, 2001

Inledning

Examensarbetet hade två syften. För det första att ta fram en metodik för det som kallas dynamiska larm. För det andra att utveckla kunskapsnivån på området dynamiska larm för vattenkraftanläggningar. Arbetet tillämpades på Bodens kraftstation och alla mätvärden var tagna från ett och samma aggregat. Framtagna modeller och larmparametrar har främst baserats på värden som samplats en gång per dygn trots att händelseförloppen vid fel kan vara mycket snabba. Detta eftersom tätare samplingar inte fanns registrerade mer än en kort tid. Den insamlade informationen användes för att ta fram dynamiska larm för kontroll av oljemängd i kaplanturbiner och övervakning av temperaturer och oljenivåer i lager. Här kommer endast övervakning av temperaturer i lager att tas upp.

Teori och metod

Systemidentifiering
Det första som gjordes var att undersöka korrelationen för att få en indikation på hur tydliga sambanden var mellan olika signaler. Är korrelationen nära ett finns ett samband och är den nära noll är sambandet litet. För att modellera de olika sambanden mellan signalerna har ARX-modeller använts eftersom de antogs kunna beskriva systemet på ett bra sätt. Flera olika modeller prövades i Matlab med det grafiska gränssnittet, *Ident (System Identification Toolbox)*. För validering har främst korsvalidering använts.

Larmdesign
Två olika typer av larm testades, dynamiskt larm och trendlarm. Det dynamiska larmet bygger på att en skattad bärlagertemperatur jämförs med den riktiga. Felet mellan värdena får sedan inte avvika mer än med ett visst gränsvärde. Trendlarmet bygger på att en variabel som ändrats alltför hastigt indikerar att något var fel. För att studera ändringen gjordes en funktion som tog hänsyn till hastigheten i ett förlopp genom att beräkna derivatan. En maximal derivata beräknades som fick fungera som larmgräns.

Resultat

Systemidentifiering
Det första antagandet var att bärlagertemperaturen, Tb, påverkades av vattentemperaturen, Tv, eftersom lagret var vattenkylt, samt troligtvis också av produktionen, P, och producerad reaktiv effekt, Pr. Av korrelationsresultaten framgår att det fanns ett tydligt samband mellan Tb och Tv för dygnsvärden, men för

minutvärdena var dataserien troligen från en för kort tidsperiod för att sambandet skulle ses eftersom vattentemperaturen varierar så långsamt. Inget samband mellan *Tb* och *P* samt mellan *Tb* och *Pr* kunde fastställas med säkerhet.

Ett försök gjordes för att ta fram modeller utifrån minutvärdena men eftersom att serierna var så korta gick det inte att ta fram några intressanta modeller. Det slutgiltiga resultatet var att använda sig av en statisk linjär trendmodell baserad på dygnsvärden och med vattentemperaturen som enda insignal för skattning av bärlagertemperaturen. Eventuell tidsfördröjning och dynamik hos systemet kunde inte fångas upp av dygnsvärden.

Larmhantering
Larmgränser och parametrar togs fram för både dynamiskt larm och trendlarm (tabell B1).

Tab. B1 Larmparametrar för bärlagertemperaturen.

Larmparameter	**Värde**	**Enhet**	**Kommentar**
Lgmax	70	°C	
Lgmin	-	°C	Lägsta värde saknas i Boden
Avvikelse	2	°C	
Maxderivata	0,4	°C/minut	
B1	0,2943	$°C_{BT}/°C_{vatten}$	
Y_{medel}	65	°C	$Y_{minutvärden}$ 63°C
U_{medel}	10	°C	$U_{minutvärden}$ 5,1°C

Slutsats

Larmparametrarna för det dynamiska larmet var tämligen tillförlitliga då långa serier med data från flera år fanns tillgängliga. De största svagheterna fanns troligen i trendlarmet och då framförallt eftersom det inte fanns minutvärden tillgängliga för att se hur temperaturförloppet uppförde sig i situationer då bärlagret överhettats.

För att det överhuvudtaget ska vara värt att försöka bestämma larmparametrarna enligt den metodik som utarbetats här krävs att de relevanta variablerna registrerats under minst ett år men helst längre. Denna metodik lämpar sig alltså bäst för aggregat som använts under en tid och inte vid nyinstallationer eftersom att det tar lång tid innan all data är insamlad.

Det måste även poängteras att de specifika lösningarna för varje larm som tagits fram är unika för just detta aggregat med den kalibrering av givare som gällt då de använda dataserierna registrerats.

Appendix 2

Sammanfattning av Mattias Sjölunds examensarbete ”Ett feldetekteringssystem för bärlager i en vattenkraftstation”, 2002

Inledning

Målet för det här examensarbetet var att utveckla larm för bärlagertemperaturen i en Kaplanturbin och att implementera en demonstrationslösning i en vattenkraftstation. Arbetet har delats upp i två delar. Den första delen har varit att utveckla en modell som uppdateras rekursivt baserad på data samplat en gång per minut. Den andra delen har varit att utveckla en detektor för att upptäcka avvikelser i bärlagertemperaturen. Modellen för bärlagertemperaturen uppdateras med rekursiva minsta kvadratmetoden. Detektorn har baserats på CUSUM-algoritmen.

Metod och utförande

Den valda demonstrationsanläggningen har liksom för Minna Glemme (2001) varit Bodens kraftstation. De parametrar som studerats är bärlagertemperaturen och kylvattentemperaturen. Minutdata från februari 2001 till februari 2002 fanns tillgängliga. Den valda identifieringsmetoden är den rekursiva minstakvadratmetoden med glömskefaktor (RLS-algoritm). Det innebär att den skattade bärlagertemperaturen beror på tidigare uppmätta och skattade värden. För feldetekteringen har CUSUM-algoritmen valts. Idén med CUSUM-algoritmen är att ett larm skall avges när summan av skillnaden mellan skattat och verkligt värde överstiger en viss larmgräns, h. För att motverka att tiden för upptäckt blir för snabb, så nollställs testvariabeln (g) när den är mindre än noll. Endast abrupta fel kan upptäckas av RLS-algoritmen.

För att visuellt kunna presentera resultat och grafer under körning har ett enkelt GUI (Graphical User Interface) utvecklats i Matlab. Programmet installerades inte i Boden då en fristående applikation inte varit möjlig att skapa. För att kunna installera programmet i Boden för en demonstrationskörning så krävdes det att koncntrator.dll installerades samt det separat exekverbara programmet, vilket inte var möjligt.

Resultat

Flera testkörningar gjordes för att jämföra vilket värde på λ som gav bäst resultat för upptäckt av fel. Detta gjordes genom att grafiskt studera skillnaden mellan predikterat och faktiskt värde på bärlagertemperaturen. För att möjliggöra upptäckt sattes $\lambda = 0{,}999$.

Parametrar som använts för inställning av CUSUM-algoritmen är h och v där h är gränsvärdet som det beräknade g jämförs med och v kan tolkas som den avvikelse som tolereras innan ett larm beslutas. Exempelvis kan $v = 2$ innebära att en 2°C skillnad tolereras. Utifrån resultaten från testkörningarna valdes följande parameterinställningar:

$\lambda = 0{,}999$
$v = 2$
$h = 1000$

För att kunna testa CUSUM-algoritmens möjlighet att upptäcka abrupta fel lades ett konstgjort fel in i datasetet. Det konstgjorda felet som lades in var en 0,1°C temperaturökning av bärlagertemperaturen per minut under 30 minuter. Det inlagda felet upptäcktes av CUSUM-algoritmen efter 20 min och då har bärlagertemperaturen avvikit med 2°C. Tiden till upptäckt av avvikelsen är beroende på utformningen av det inlagda felet.

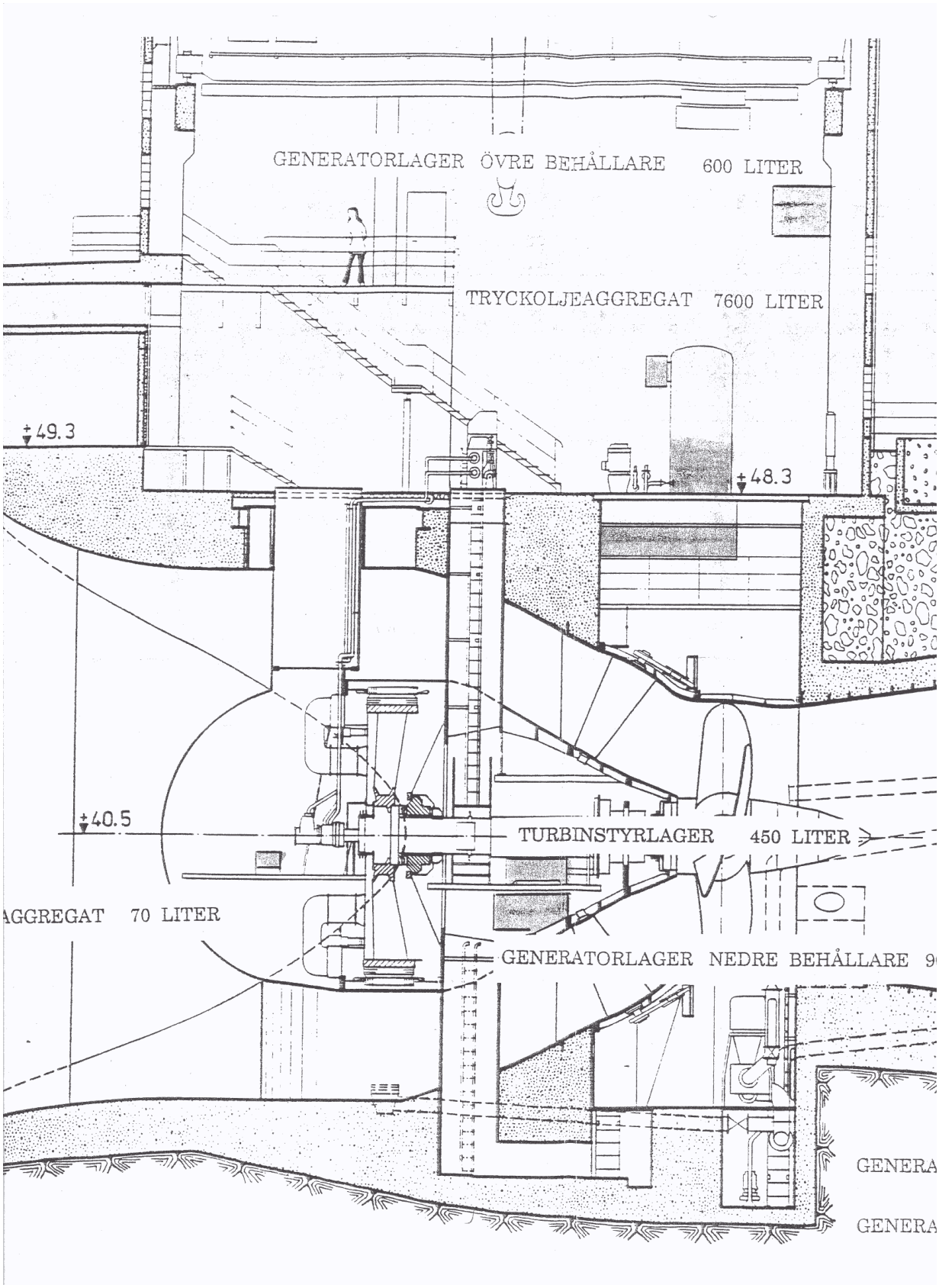
Diskussion

Den valda utformningen av larmfunktion har visat sig att fungera väl på insamlad mätdata från Bodens kraftstation. Om bärlagrets medeltemperatur ökar sakta under en tid kommer det inte att upptäckas av felalgoritmen. Detta eftersom glömskefaktorn i RLS-algoritmen har valts så att fel av en plötslig art skall upptäckas. För att kunna upptäcka ett sakta ökande medelvärde krävs ytterligare utveckling av algoritmen.

Det har inte funnits data lagrad från eventuella larm för bärlagertemperaturen varvid en studie av feldata inte har skett. Även om det funnits så hade det troligen varit på formen dygnsdata vilket inte är tillräckligt då förloppet kan vara abrupt.

Appendix 3

Figur över turbin i Söderfors.



Appendix 4

Identifierbarhet

När det gjorts en skräddarsydd modell är det intressant att veta om de valda parametrarna överhuvudtaget kan bestämmas ur data. Begreppet identifierbarhet används för att ange att en viss parameter kan identifieras ur insignal-utsignal data. Om två olika parametervektorer θ_1 och θ_2 ger upphov till identiska prediktioner kan de alltså inte skiljas åt med identifieringsmetoder. Betrakta den olinjära tillståndsmodellen

$$\begin{aligned} \frac{dx}{dt} &= f(x,u,\theta), \quad x(0) = x_0(\theta) \\ y(t,\theta) &= h(x,\theta) \end{aligned} \tag{1}$$

där x, u, y och θ representerar tillståndsvektorn, insignalsvektorn, utsignalsvektorn och den okända parametervektorn. Metoden att undersöka identifierbarhet är baserad på Taylorutveckling av $y(t)$ vid tiden $t = 0$.

$$y(t) = y(0) + t\frac{dy}{dt}(0) + \frac{t^2}{2!}\frac{d^2y}{dt^2}(0) + \ldots \tag{2}$$

Sedan undersöks de på varandra följande derivatorna för att se om de innehåller någon information om parametrarna som skall identifieras. Mer precist kan det sägas att $y(0)$ och derivatorna av $y(t)$ vid tiden $t = 0$ kan skrivas utifrån (1) som funktioner av de okända parametrarna $\theta^T = [\theta_1, \theta_1, \ldots, \theta_1]$:

$$\begin{aligned} y(0) &= \gamma_0(\theta) \\ \frac{dy}{dt}(0) &= \gamma_1(\theta) \\ &\vdots \\ \frac{d^q y}{dt^q}(0) &= \gamma_q(\theta) \end{aligned} \tag{3}$$

Nästa steg innebär att försöka invertera ovannämnda uttryck till att uttrycka parametrarna θ_i (i = 1 till p) som funktioner av enbart $y(0)$, dess derivator och insignalen u.

$$\begin{aligned} \theta_1 &= \beta_1(y(0), \frac{dy}{dt}(0), \ldots, \frac{d^q y}{dt^q}(0), u) \\ \theta_2 &= \beta_2(y(0), \frac{dy}{dt}(0), \ldots, \frac{d^q y}{dt^q}(0), u) \\ &\vdots \\ \theta_p &= \beta_p(y(0), \frac{dy}{dt}(0), \ldots, \frac{d^q y}{dt^q}(0), u) \end{aligned} \tag{4}$$

Om en sådan uppsättning ekvationer existerar så är parametrarna θ_i (i = 1 till p) identifierbara. Det kan också hända att de ovannämnda ekvationerna endast kan skrivas

som kombinationer av parametrarna θ_i, och att endast dessa kombinationer kan identifieras (Dochain et al., 2001). För tillståndsmodellen i ekvation (5.6) blir ovanstående ekvationssystem ganska komplicerat och har ej utförts.

Observerbarhet

Systemet är observerbart då det saknar icke observerbara tillstånd. De icke observerbara tillstånden utgör ett linjärt rum, nämligen nollrummet till matrisen (observerbarhetsmatrisen för olinjära modeller) (Rugh, 1996)

$$\mathrm{O}(A,C) = \begin{bmatrix} L_0(t) \\ L_1(t) \\ \vdots \\ L_q(t) \end{bmatrix} \tag{5}$$

där, utifrån (5.6), har

$$\begin{aligned} L_0(t) &= C(t) \\ L_j(t) &= L_{j-1}(t)A(t) + L_{j-1}(t), \quad j = 1,2,\ldots \end{aligned} \tag{6}$$

Systemet är alltså observerbart då O har full rang. För systemet (5.6) gäller att den har full rang då $a_{21}, a_{32}, a_{43} \neq$

Appendix 5

Okända parametrar

Ur ekvation (5.6) ses att det är 12 okända parametrar:

α (K m^3/s) = tillförd värme till bärlagret
β_1, β_2, β_3, β_4 (m/s) = $1/c_{pV,O,B}\rho_{V,O,B}$
c_1, c_2, c_3, c_4 (1/m^3) = $1/V_{V,O1,O2,B}$
d_1, d_2 (J/Ks) = $A_{1,2}\mu_{1,2}$
f (m^3/s) = oljeflöde

Här ses att α inte kan beräknas utan måste skattas. Däremot kan β_1, β_2, β_3 och β_4 bestämmas utifrån värden på specifik värmekapacitet och densitet. Eftersom volymerna är okända kan inte heller c_1, c_2, c_3 eller c_4 bestämmas utan måste skattas. Samma sak även för d_1, d_2 och f då arean, värmegenomgångstalet och oljeflödet är okända. Värmegenomgångstalet beror på material i ”värmeväxlaren” och varierar beroende på beläggningar på väggarna och kan därför inte beräknas.

Beräkning av β_1, β_2, β_3 och β_4:

Specifik värmekapacitet (300 K):
c_{pV} = 1040 J/kg K (Water)
c_{pO} = 1870 J/kg K (Lubricating oil)
c_{pB} = 449 J/kg K (Iron)

Densitet (300 K):
ρ_V = 997 kg/m^3 (Water)
ρ_O = 880 kg/m^3 (Lubricating oil)
ρ_B = 7300 kg/m^3 (Iron) (Nordling & Österman, 1996)

$\beta_1 = 1/(c_{pV}\rho_V) = 1/(1040*997) = 1/1036880$ K m^3/J
$\beta_2 = 1/(c_{pO}\rho_O) = 1/(1870*880) = 1/1645600$ K m^3/J
$\beta_3 = 1/(c_{pO}\rho_O) = 1/(1870*880) = 1/1645600$ K m^3/J
$\beta_4 = 1/(c_{pB}\rho_B) = 1/(449*7300) = 1/3277700$ K m^3/J

Utifrån detta kan således konstateras att det är åtta okända parametrar. Inför en vektor Θ som innehåller parametrarna θ_1, …, θ_8, där $\theta_1 = c_1$, $\theta_2 = c_2$, $\theta_3 = c_3$, $\theta_4 = c_4$, $\theta_5 = d_1$, $\theta_6 = d_2$, $\theta_7 = f$ och $\theta_8 = \alpha$. Eftersom att $\beta_2 = \beta_3$ ersätts β_3 med β_2 och sen sätts $[u_1(t)u_2(t)\ u_3(t)]^T = u(t)$. Med vektorn Θ fås ekvation (5.6) som

$$\dot{x}(t) = \begin{bmatrix} -(u_1 + \beta_1\theta_5)\theta_1 & \beta_1\theta_1\theta_5 & 0 & 0 \\ \beta_2\theta_2\theta_5 & -(\theta_7 + \beta_2\theta_5)\theta_2 & \theta_7\theta_2 & 0 \\ 0 & \theta_7\theta_3 & -(\theta_7 + \beta_2\theta_6)\theta_3 & \beta_2\theta_3\theta_6 \\ 0 & 0 & \beta_4\theta_4\theta_6 & -\beta_4\theta_4\theta_6 \end{bmatrix} x(t) + \begin{bmatrix} \theta_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \theta_8\theta_4 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} x(t)$$

För det stationära fallet är det enligt ekvation (5.15) två okända parametrar, k_1 och k_2 som innehåller några av parametrarna i vektorn Θ. Hädanefter betecknas $k_1 = \theta_1$ och $k_2 = \theta_2$.

Uppskattning av parametrar

Nedan har en grov uppskattning gjorts över initialvärden på parametrarna.

A_1 (m^2): Överföringsarean i värmeväxlaren mellan kylvatten och olja. Kan uppskattas till cirka 0,5 m^2.
A_2 (m^2): Överföringsarean mellan olja och bärlager. Beror på bärlagrets storlek. Uppskattas till ett par kvadratmeter.
V_V (m^3): Volymen vatten i värmeväxlaren. Uppskattningsvis 0,010 m^3.
V_{O1} (m^3): Volymen olja i värmeväxlaren. Uppskattningsvis 0,100 m^3.
V_{O2} (m^3): Volymen olja vid bärlagret. Uppskattningsvis 0,100 m^3.
V_B (m^3): Volymen på bärlagret. Uppskattningsvis 0,200 m^3.
μ_1 (J/m^2Ks): Värmegenomgångskoefficienten för väggmaterialet i värmeväxlaren. Enligt jämförbar värmeväxlare, ca 150-200 J/m^2Ks (Weyer et al., 2000).
μ_2 (J/m^2Ks): Värmegenomgångskoefficienten för bärlagret. Enligt ovan, ca 150-200 J/m^2Ks (Weyer et al., 2000).
f (m^3/s): Oljeflöde. Flera liter i sekunden, 50 l/s = 0,05 m^3/s.
α (K m^3/s) = Tillförd värme till bärlagret. Svår att uppskatta, ca 0,0001 K m^3/s.

Appendix 6

Beskrivning av hur behandling av mätdata innan systemidentifiering ska ske. Detta måste göras som ett första steg då alla dataserier har värden med olika samplingsintervall och inte samplade vid samma tidpunkter. Resultatet är att få ut data med en minuts samplingsintervall och att alla mätvärden är vid samma tidpunkter, mellan serierna.

1. Samla in mätdata. Detta görs med ett DOS-program utvecklat av Conwide. Programmet heter R_AI_FIL.EXE. Användarbeskrivning finns nedan. För stora mängder data (mer än 45000 värden) delas intervallet upp i lagom stora delar och sparas på olika disketter. Om data samplas var 15:e sekund ryms data från 7 dagar på en diskett.
2. Mätdata som sparats som *txt*-filer öppnas i t.ex. WordPad. Därefter markeras all data och kopieras. Dessa klistras in i ett tomt Excelblad.
3. Eftersom dataposterna innehåller tecken som -, : och , så måste de ersättas med andra tecken. Detta görs med kommandot *Ersätt* som hittas under *Redigera* menyn. Ersätt först – med *blank*, d.v.s. ingenting. Det som då händer är att alla – försvinner. Får då #####. För att göra om dessa till siffror måste hela kolumnen markeras och därefter högerklicka, välja *Formatera celler*, *Tal* och *Allmänt*. Det resulterar i att t.ex. 2003-10-23 blir till 20031023.
4. Ersätt sedan : med ett *mellanslag*. Tiden 08:01:02 blir 08 01 02.
5. Markera sedan hela kolumnen med tid. Högerklicka på kolumnen och välj *Formatera celler*. Välj *Tal* och ändra till *Allmänt*.
6. Ersätt mellanslag med *blank*. Får då 080102.
7. Ersätt slutligen , med . för att lättare hantera i Matlab.
8. Kopiera alla värden och klistra in i ny m-fil i matlab. Därefter spara filen som *.mat*-fil. Denna kan sedan användas i programmet utfyllnad.m.
9. Kör programmet utfyllnad.m. Innan start måste det ses till att rätt filnamn anges och att filen som data sparas i och dess variabelnamn får nya namn. Annars sparas den existerande filen över. OBS! Kan endast hantera en månad i taget, till exempel 1/11-30/11. Får sedan sätta ihop alla månader i en fil.

Hur programmet r_ai_fil.exe (Lekby, 2001) ska användas för att samla in data från en koncentrator.

Steg 1 Bestäm dig för vilken signal samt analogport som är intressant tex. KO1A10 Som är MW G1 i detta exempel.

Steg 2 Anslut en monitor och tangentbord till koncentratorn. Du ska se en bild med massor av siffror på. Längst uppe i vänstra hörnet finns datum och tid. Tiden ska ändra sig varje sekund.

Steg 3 Tryck Esc på tangentbordet så stannas programmet på ett bra sätt. Det ska nu stå C:\KONC> på skärmen

Steg 4 Läs in programmet R_AI_FIL.EXE som finns på en diskett.
Tex. Med Copy A:R_AI_FIL.EXE C:*.* <ret>

Steg 5 Starta R_AI_FIL.EXE

Mata in parametrar i enlighet med exemplen i programmet.
Inga konstigheter (streck punkter eller kolon eller annat tjafs i parametrarna) utan

Analog inkanal 10
Startdatum 20010501
Stoppdatum 20010516
Starttid 081500
Stopptid 083500
Spara till A:\MWG1.TXT

Är tänkt att ta värden från KO1A10 från 2001-05-01 kl 08:15:00 till 2001-05-16 kl 08:35:00 och lägga dessa i utfilen.

Utfilen består av en TAB separerad textfil som kan importeras i Excel. De resultat som fås är osorterade med avseende på datum och tid. Denna sortering görs i Excel.

Steg 6 När du är klar starta loggern igen genom att till C:\KONC> skriva K <ret>
(Programmet i koncentratorn heter K.EXE)

Steg 7 Ta bort tangentbord och monitor.

Appendix 7

```
%utfyllnad.m. Program som fyller ut en dataserie med värden varje sekund.
%Data lagras i en fil varje minut för att få samma samplingsintervall.
%Går endast att ta en månad i taget. Klarar inte av övergångar mellan månader.

clear;
format long

%Filen där ursprungsdata är lagrad öppnas (här flödet)
load -ascii flode.mat

%Anger vilka värden som är intressanta
test = flode(5838:101203,:);

%Räknare
m = 1;
f = 0;
h = 0;
g = 0;

%Sparar endast tid och värde i matris
testfil(1,1:2) = test(1,2:3);

%Begynnelsevärdet. Det klockslag som man vill börja körningen på, t.ex. 80000 för 08:00:00
testfil(1,1) = 0;
prov = testfil(1,1:2);

%En loop som läser igenom alla data
for i=1:length(test(:,2))
   if (i>2)
      h = h + test(i,1)-test(i-1,1);
   end

   while (prov(1,1)<(test(i,2)+240000*h))
      prov(1,1) = prov(1,1) + 1;

      if (mod(prov(1,1)-testfil(1,1)-100*f-10000*g,60)==0)
         f = f + 1;
         prov(1,1) = testfil(1,1) + 10000*g + 100*(f-60*g);

         if (mod(prov(1,1)-testfil(1,1)-10000*g,6000)==0)

            g = g + 1;
            prov(1,1) = testfil(1,1) + 10000*g;
         end
```

```
        end

        %Sparar varje minut
        if (mod(prov(1,1)-100*f-10000*g-testfil(1,1),60)==0)
            testfil(m+1,:) = prov(1,:);
            m = m + 1;
        end

        p = [prov(1,1) test(i,2)];
    end

    prov = test(i,2:3);
    prov(1,1) = test(i,2)+240000*h;
end

%Sparar värden i två kolumner f1 och f2 för flöde, v1 v2 för vtemp osv.
f1 = testfil(:,1);
f2 = testfil(:,2);

%sparar i en .mat-fil
save vattenflode.mat f1 f2
```

Appendix 8

Beskrivning hur framtida systemidentifiering bör göras

Detta är en nödvändig åtgärd eftersom att den nuvarande modellen inte är anpassad efter årstidsvariationer. När temperaturen och flödet ökar kommer modellen troligtvis inte att ge något bra resultat som den är nu. Därför måste en ny systemidentifiering göras innan feldetekteringssystemet börjar användas ”skarpt”.

1. Samla in data över effekt, vattentemperatur, vattenflöde och bärlagertemperatur. Data ska helst spänna över ett år och inte mindre än över ett halvår. Insamlingen sker på plats i Söderfors eller om möjligt från databas i Råcksta. Sker insamlingen på plats i Söderfors måste det ses till att data samlas in med jämna mellanrum så att inte koncentratorn hinner skriva över data. Ca 45 000 mätvärden ryms på en diskett. Data är samplade med minst 15 sek. och högst 60 sek. mellanrum. Om det är möjligt bör systemet exciteras genom att ändra på börvärdet för bärlageroljan och/eller använda data som är insamlat utan reglering av kylvattenflödet.

2. Förbehandla data, se beskrivning av *utfyllnad.m* i rapport, appendix 6. Programmet ser till att data får samma samplingsintervall. Var noggrann med att kontrollera att dataserierna börjar vid samma tidpunkt och att alla mätvärden är från samma tidpunkt.

3. Därefter kan kurvorna börja tittas på, förslagsvis med *Matlab*. Rita upp och studera utseendet. För samtliga kurvor ta bort sekvenser som har onormalt beteende, det vill säga då aggregatet varit avstängt eller koncentratorn varit avstängd. Samma sekvens måste tas bort från samtliga dataserier. Ändra avvikande värden till normala, till exempel mätfel. Ta även bort uppstartsförfarandet efter att aggregatet varit avstängt.

4. Om olika ARX-modeller skall prövas kan dataserierna direkt användas i *Ident.* I matlabfönstret skriv: load dataserie.mat. Då läggs dataserierna i Workspace och kan läsas av *Ident*. Öppna *Ident* genom att skriva *Ident* i Matlabfönstret. Därefter öppna *Import* och importera dataserierna ([*u*1 *u*2] och *y*). Dra bort medelvärden, dela upp i kalibreringsdata och valideringsdata och testa olika ARX-modeller. Utvärdera modellerna med avseende på *model output* m.m. När en bra modell erhållits dra modellen med musen till *workspace*. Parametrarna kan då hämtas från kommandofönstret med *data = namn på modell(till exempel arx010); z = get(data,’outputdata’); z2 = get(data,’inputdata’);*.

5. Skall *statisk_est.m* användas måste medelvärden dras bort. Dessa skall noteras för att senare läggas till på modellen. Dela upp serien i kalibreringsdata och valideringsdata. Ställ in parametrarna i *statisk_est.m* och välj rätt dataserier. Kör sedan programmet och validera modeller med avseende på passning och maximal avvikelse.

6. När en modell är vald notera dess parametrar. En ekvation på formen: $y = b_1*(u_1 - m_1) + b_2*(u_2 - m_2) + m_3$ skall nu vara erhållen om ingen dynamik är använd. Här är m_1 = medelvärdet på u_1, m_2 på u_2 och m_3 på y. Ekvationen används sedan för att estimera en ny bärlagertemperatur.

7. Implementera den nya modellen i antingen *cusumtest.m* eller i Delphi-programmet. Se till så att parametrarna är rätt inställda.

8. Använd orginaldata (data med 60 sekunders samplingsintervall men i övrigt obearbetade) för bärlagertemp, vattenflöde- och temperatur samt effekten (hela serierna). Välj först utan inlagda fel för att testa så att det fungerar. Använd sedan ett inlagt fel för att testa att feldetekteringen fungerar. Om inte ändra parametrar och testa vilken som passar bäst. Förslagsvis öka eller minska *h*. Blir det många falsklarm på grund av mätfel öka *run test* variabeln.

9. Vid uppstarter av aggregatet kan det ges falsklarm. Det bör läggas till en programsnutt som väntar en viss tid eller ett visst antal sampel innan feldetekteringen startar för att undvika falsklarm.

10. Implementera slutligen i Delphi och utvärdera så att modellen fungerar som den ska även här.

OBS! Kunskap om Matlab och systemidentifiering är viktigt då många beslut annars kan bli felaktiga. En rekommenderad version av är Matlab 6.5 och se till att System Identification Toolbox är inkluderad.

Appendix 9

```
%tillstandest.m. Estimering av tillståndsmodellen.

clear;

global y u u1 N

%Öppnar filen där data är sparat. Data skall ej vara medelvärdesbildat.
load behandlatdata.mat

%Väljer ut intressant datamängd
y = y(5761:5760+6000);
u2 = u2(5761:5760+6000);
u1 = u1(5761:5760+6000);

N = length(y);
u3 = ones(1,N);

%Anger insignalerna
u = [u1'.*u2'; u3];

%Anger begynnelsevärden på theta-parametrarna
tho = [10 10 10 50 75 300 5 0.05];

%startar minimeringsalgoritmen
th = fminsearch('tillstandtheta',tho);
```

```
%tillstandtheta.m. Estimeringsalgoritmen.

function J = tillstandtheta(theta)

global y u u1 N

%Anger R1, R2 och P-matris för Kalmanfiltret
R2 = 0.001;
R1 = diag([1 1 1 0.005]);
P = 1*ones(4);

t=1:1:N;

%Initialvärden på tillstånden samt betavärden
x=[278;296;296;305.32];
beta = [1/1036880 1/1645600 1/3277700];
yhat=zeros(1,N);

%Loop för uppdatering av tillståndsmodell
for j=1:N
   %A-matrisen
   A = [-(u1(j)+beta(1)*theta(5))*theta(1) beta(1)*theta(1)*theta(5) 0 0;
      beta(2)*theta(2)*theta(5) -(theta(7)+beta(2)*theta(5))*theta(2) theta(7)*theta(2) 0;
      0 theta(7)*theta(3) -(theta(7)+beta(2)*theta(6))*theta(3) beta(2)*theta(3)*theta(6);
      0 0 beta(3)*theta(4)*theta(6) -beta(3)*theta(4)*theta(6)];

   B = [theta(1) 0; 0 0; 0 0; 0 theta(8)*theta(4)];
   C = [0 0 0 1];

   %Gör om till state-space
   sys = ss(A,B,C,0);

   %Gör om till diskret tid
   sysD = c2d(sys,0.1);

   %Sparar yhat
   yhat(1,j) = sysD.C*x;

   %Beräknar nya tillstånd
   x = sysD.A*x + sysD.B*u(:,j);

   %Uppdatering av Kalmanfilter då den används
%     K = sysD.A*P*sysD.C'*inv(sysD.C*P*sysD.C' + R2);
%     x = sysD.A*x + sysD.B*u(:,j) + K*(y(j)-yhat(1,j));
%           P = sysD.A*P*sysD.A' + R1 - sysD.A*P*sysD.C'*inv(sysD.C*P*sysD.C' +
R2)*sysD.C*P*sysD.A';
```

```
end

%beräknar prediktionsfelet
err=y'-yhat;

%Beräknar minimeringskriterie
J=err*err'

%Beräknar maxfelet
maxfel = max(abs(err))

%Ritar ut kurvor då man jämför med valideringsdata

% figure(1);
% plot(t,y');
% hold on;
% plot(t,yhat','r');
% title('Skattad och riktig bärlagertemperatur');
% xlabel('Sampel');
% ylabel('Temperatur (K)');
% legend('y','yhat');
% hold off;
%
%
% figure(2)
% plot(err);
% title('Residualer');
% xlabel('Sampel');
% ylabel('Temperaturskillnad (K)');
```

Appendix 10

```
%statisk_est.m. Beräkning av olika statiska modeller.

clear;

%Laddar intressanta dataserier
load estdata.mat
load valdata.mat

%Anger tidsfördröjningar
% u1e = u1e(9:length(u1e));
% u2e = u2e(1:length(u2e)-8);
% ye = ye(1:length(ye)-8);
%
% u1v = u1v(9:length(u1v));
% u2v = u2v(1:length(u2v)-8);
% yv = yv(1:length(yv)-8);

%Lägger in kalibreringsdata
u1 = u1e;
u2 = u2e;
y = ye;

%Anger ordningen på modellen [b1 b2] = nb och a1 = na
b1 = 1;
b2 = 1;
a1 = 0;

na = a1;
nb = [b1 b2];

%Ska alltid vara [0 0]. Tidsfördröjningar anges längst upp.
nk = [0 0];

yhat(:,1) = zeros(6000,1);

%Anger ifall man testar med 1/u1 eller 1*u1
u1_invers = 1.*u1;     %Ändra här för den stationära modellen
y_ny = y;

%Skapar ett iddata-objekt
data = iddata(y_ny,[u1_invers u2]);

%Beräknar statisk modell
m = arx(data, 'na', na, 'nb', nb, 'nk', nk)
```

```
%Loop för beräkning av yhat.
for i = b1:6000

   for c = 0:b1-1
      %Här ändras beroende på u1_invers! Ändra om stationär modell.
      yhat(i,1) = yhat(i,1)+m.b(1+2*c)*u1v(i-c);
   end

   if (b2 == 0)
      yhat(i,1) = yhat(i,1)+u2v(i);
   else
      for a = 1:b2
         yhat(i,1) = yhat(i,1)+m.b(2*a)*u2v(i-a+1);
      end
   end

   if (a1 == 0)
      1;
   else
      for x = 1:a1
         yhat(i,1) = yhat(i,1)-m.a(x+1)*yhat(i-x,1);
      end
   end
end

%beräknar prediktionsfelet
err = yv(10:6000) - yhat(10:6000);

%Minimerar kriterie
J = err'*err

%beräknar maxfelet
maxfel = max(abs(err))

%Plottar figurer
figure(1);
plot(yv(10:6000));
hold on;
plot(yhat(10:6000),'r');
title('Skattad och riktig bärlagertemperatur');
xlabel('Tid (minuter)');
ylabel('Temperatur (K)');
legend('y','yhat');
hold off;
```

```
figure(2)
plot(err);
title('Residualer');
xlabel('Tid (minuter)');
ylabel('Temperaturskillnad (K)');
```

Appendix 11

%cusumtest.m. Program för att testa om feldetekteringen fungerar som den ska och för inställning av CUSUM-parametrar

```
clear;

%Öppnar orginaldata.mat som innehåller originaldata, får ej vara medelvärdesbildat m.m.
load orginaldata.mat


%Lägger in lite hypotetiska fel
for i = 1:1500
   %Ramp, långsamt fel. Ökning av 0.01 grader per minut
    y(6000+i) = y(6000+i)+0.01*i;
    %Statiskt fel, förändring av medelvärde
    y(10000+i) = y(10000+i)+3;
    %Förändring av standardavvikelse
    y(14000+i) = (y(14000+i)-305.3597934376886+273.15)*5+305.3597934376886-273.15;
end
%Ett mätfel som pågår 11 sampel
y(3000:3010)=y(3000:3010)+3;
%Givar- eller mätfel på flödesmätaren
u1(4000:4010) = 1;
%Givar- eller mätfel på tempmätaren
u2(13000:13010) = u2(13000:13010) + 10;

%Bärlagrets maximala temperatur
y_max = 46;
%Räknar antalet larm som har inträffat
larm_raknare = 0;

s = 0;
p = 0;
k = 0;


%CUSUM-parametrar
h = 10;
v = 2.0;

%RUN-test
larm_grans = 3;

g_old(1:2,1) = 0;
larm = zeros(1,6);
```

```
%Beräkning av skattad bärlagertemperatur. Ändra här för ny modell.
yhat = -0.1535*(u2-277.4177772242127+273.15)+187.4*(u1/1000-
0.01563805244247)+305.3597934376886-273.15;

%Loop som kollar efter larm
for i = 1:length(y)
   tid = i;

   %Beräknar felet
   err = y(i) - yhat(i);

   %Kollar så att temperaturen inte överstiger maximal temperaturen
   if (y(i) > y_max)
      p = p + 1;

      %Ger larm då det hänt två gånger på rad
      if (p == larm_grans - 1)
      larm_raknare = larm_raknare +1;   %Räknar larm
      larm(larm_raknare,1) = larm_raknare;
      larm(larm_raknare,2) = y(i);
      larm(larm_raknare,3) = 1;   %Typ av larm
      larm(larm_raknare,4) = err;
      larm(larm_raknare,5) = 0;
      larm(larm_raknare,6) = tid;
   end
   else
      k = k+1;

      %Beräkning av CUSUM-algoritmen
      err_pos = err;
      err_neg = -err;
      pos_g(k,1:2) = [(g_old(1,1) + err_pos - v) 0];
      neg_g(k,1:2) = [(g_old(2,1) + err_neg - v) 0];
      pos_g_new = max(pos_g(k,1:2));
      neg_g_new = max(neg_g(k,1:2));

      %Kollar om CUSUM överstiger gränsvärde h
      if (pos_g_new > h | neg_g_new > h)
         s = s + 1;
         if (s == larm_grans)
            larm_raknare = larm_raknare + 1;
            larm(larm_raknare,1) = larm_raknare;
            larm(larm_raknare,2) = y(i);
            larm(larm_raknare,3) = 2;    %Larmtyp
            larm(larm_raknare,4) = err;  %Felet
```

```
            larm(larm_raknare,5) = pos_g_new;
            larm(larm_raknare,6) = tid;
        end

        g_old(1:2,1) = 0;
    else

        %Nollställning av larm då det är slut (kan göras manuellt, s.k. larmkvittering
        if (err < 0.5)
            s = 0;
            p = 0;
        end

        %Om ej larm sätts g_old till nytt värde
        g_old(1,1) = pos_g_new;
        g_old(2,1) = neg_g_new;

    end
  end
end

larm_raknare


%Uppritning av figurer
figure(1)
plot(y);
hold on;
plot(yhat,'r');
hold off;

figure(2)
plot(pos_g(:,1));
hold on;
% plot(neg_g(:,1),'r');
hold off;

figure(3)
plot(larm(:,5))
hold on;
% plot(larm(:,6),'r')
hold on;
plot(larm(:,4),'g')
hold off;
```

Appendix 12

Nedan är Delphi-koden presenterad. Det är endast huvudkoden (Unit 2) som är med.

```
unit Unit2;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DBCtrls, Menus, DB, ADODB, Grids, DBGrids, ExtCtrls,
  ActnList, ActnMan, comctrls, DateUtils;

type
  TMyThread = class(TThread)
  private
  protected
    procedure Execute; override; // Main thread execution
  published
    constructor Create();
    destructor Destroy; override;
  end;

//Egen typdef av Vektor och Matris
Type TMinVector = array[1..2] of Real;
Type TMinMatrix = array[1..2] of array[1..2] of Real;
(*Ändra här för hur stor vektor och matris maximalt kommer att vara*)

//Egen typdef av y vektor för uppritning av kurva
//Type PlotVektor = array[1..100] of Real;
//Type PlotVektor = array[1..100] of Real;

Function MaxVector(VecA : TMinVector): Real;
(*Pre: En (1*2)vector skickas in för kontroll av det största elementet.
  Post: Det största elementet i vectorn returneras.*)

procedure LogEvent(Msg: String);
Function GetVarde(varde:String):String;

implementation

uses
 Unit1;

constructor TMyThread.Create();
begin
```

```
  inherited Create(true);       //Create thread suspended
  Priority := tpIdle;           //Sätt prioritetsnivå på processen
  FreeOnTerminate := true;      //Tråden frigörs vid terminated
  resume;                       //Tråden återupptas
end;

destructor TMyThread.Destroy;
begin
   inherited destroy;
end;

//Returnerar det största elementet i vectorn som skickas in i funktionen
Function MaxVector(VecA : TMinVector): Real;
var
  i : Integer;
  Max : Real;
begin
  Max := 0;
  for i := 1 to Length(VecA) do
  begin
    if (VecA[i] > Max) then
      Max := VecA[i]
    else
      Max := Max;
  end;
  MaxVector := Max;
end;

procedure TMyThread.Execute; //Huvudfunktion för tråden

//ShowMessage(IntToStr(Length(Avektor)));
//Showmessage('Skalär = '+FloatToStr(Resultatvektor));
//ShowMessage('Resultatvektor = ['+FloatToStr(Resultatvektor[1])+'
'+FloatToStr(Resultatvektor[2])+' '+FloatToStr(Resultatvektor[3])+']');
//ShowMessage('                 '+FloatToStr(Resultatvektor[1,1])+'
'+FloatToStr(Resultatvektor[1,2])+' '+FloatToStr(Resultatvektor[1,3])+' '+#13+'Resultatvektor =
'+FloatToStr(Resultatvektor[2,1])+' '+FloatToStr(Resultatvektor[2,2])+'
'+FloatToStr(Resultatvektor[2,3])+' '+#13+'                 '+FloatToStr(Resultatvektor[3,1])+'
'+FloatToStr(Resultatvektor[3,2])+' '+FloatToStr(Resultatvektor[3,3]));



//Variabeldeklarationer
var
  i : Integer;
  flode, VTemp, BTemp, BTemp_est, effekt, err : Real;
```

```
  h, RUN_test, RUN_test_max : Integer;
  v: Real;
  pos_cusum, pos_cusum_new, err_pos, neg_cusum, neg_cusum_new, err_neg : Real;
  larm : boolean;
  larmcounter, stopcounter, s_counter, p_counter, k_counter : Integer;
  pos_Cusum_Vector, neg_Cusum_Vector : TMinVector;
  last_time_end_channel : Real;
  test : Integer;
  larmTyp, testVarde : String;
  ok : boolean;
  //Konstantdeklarationer
const
  flode_medel = 0.01563805;
  VTemp_medel = 4.267778;
  BTemp_medel = 32.20979;
  theta1 = 187.4;
  theta2 = -0.1535;
  bTemp_max = 46;     //Ändras till riktiga värdet!!!   Ska vara 46 grader
  maxrows_DB = 150000;

begin

  //test = 1 för körning mot kraschdump och test = 2 för körning mot databas
  test := 1;

  //Begynnelsevärden på olika variabler och räknare
  BTemp_est := 0;
  err := 0;
  s_counter := 0;
  p_counter := 0;
  larmCounter := 0;
  stopCounter := 1;
  pos_Cusum := 0;
  neg_Cusum := 0;
  err_neg := 0;
  err_pos := 0;

  //Tar bort gammal data ur tabell Data i databasen BLDB.mdb
  //FormMain.deleteData.ExecSQL;

  //Tar bort gammal data ur tabell Larm i databasen BLDB.mdb
  //FormMain.deleteLarm.ExecSQL;

  //Läser in larmparametrarna från databasen henke.mdb
  with FormMain.Larmparametrar do begin
    Open;
```

```
  v := FieldByName('Ny').AsFloat;
  h := FieldByName('h').AsInteger;
  RUN_test := FieldByName('runtest').AsInteger;
  RUN_test_max := FieldByName('runtest_max').AsInteger;
  close;
 end;


 //**************************Anslut till KDDB för hämtning av data************
 if (test = 1) then
 begin
  FormMain.getBTemp.Open;
  FormMain.getVTemp.Open;
  FormMain.getFlode.Open;
  FormMain.getPMW.Open;
 end
 else if (test = 2) then
 begin
  FormMain.getBTemp1.Open;
  FormMain.getVTemp1.Open;
  FormMain.getFlode1.Open;
  //Hämtar första värdet ur databasen
  FormMain.getBTemp1.First;
  FormMain.getVTemp1.First;
  FormMain.getFlode1.First;
  //Sätter effekten till 7 eftersom den aldrig är avstängd
  effekt:=7;
 end;

//*****************************************************************************
 //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 //Vid körning mot annan station än Söderfors måste det kontrolleras vilka
 //kanaler som loggar de intressanta värdena (bärlagertemp,
 //vattentemp, flöde och effekt). Värdet på dessa kanaler läggs sedan in i
 //SQL-frågorna getBLTemp, getPMW och getVTemp.
 //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

//*****************************************************************************
 //StrToFloat(Form1.GetVarde(String(tmpRec.Data)));
 ok:=false;
 while not ok do begin
 Try
  if (test = 1) then
  begin
   BTemp := StrToFloat(GetVarde(FormMain.getBTemp.FieldByName('värde').AsString));
   effekt := StrToFloat(GetVarde(FormMain.getPMW.FieldByName('värde').AsString));
```

```
    VTemp := StrToFloat(GetVarde(FormMain.getVTemp.FieldByName('värde').AsString));
    flode := StrToFloat(GetVarde(FormMain.getFlode.FieldByName('värde').AsString));
  end
  else if (test = 2) then
  begin
    BTemp := FormMain.getBTemp1.FieldByName('värde').AsFloat;
    VTemp := FormMain.getVTemp1.FieldByName('värde').AsFloat;
    flode := FormMain.getFlode1.FieldByName('värde').AsFloat;
  end;
 Except on E: EConvertError do
  LogEvent(E.Message+': fel vid start');
 end;
 if ((BTemp=-100) or (effekt=-100) or (flode=-100) or (VTemp=-100)) then begin
  ok:=false;
  LogEvent('val() = -100');
 end else
  ok:=true;
 end;

//*************************************************************************
 //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 //Vid körning mot annan station än Söderfors måste det kontrolleras vilken
 //kanal som loggar det sista av de intressanta värdena (bärlagertemp,
 //vattentemp, flöde eller effekt). För Söderfors är det vattentemperaturen som loggas
 //sist, därför används det värdet här i "last_time_end_channel"
 //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

//*************************************************************************

 if (test = 1) then
 begin
 try
  last_time_end_channel := FormMain.getVTemp.FieldByName('TID').AsFloat;
 Except on E: EConvertError do
  LogEvent(E.Message+': last_ start');
 end;
  FormMain.getBTemp.Close;
  FormMain.getVTemp.Close;
  FormMain.getFlode.Close;
  FormMain.getPMW.Close;
 end
 else if (test = 2) then
 begin
  FormMain.getBTemp1.Close;
  FormMain.getVTemp1.Close;
  FormMain.getFlode1.Close;
```

```
  end;

//*****************************************************************************
  //******************Här loopar programmet tills tråden TMyThread1 avbryts*****

//*****************************************************************************

  //Öppna tabellen Data i BLDB.mdb
  //FormMain.Data.Open;
  while (Terminated = false) do
  begin
    larm := false;
    FormMain.Data.Open;
    if (test = 1) then
    begin
    FormMain.getBTemp.Open;
    FormMain.getVTemp.Open;
    FormMain.getFlode.Open;
    FormMain.getPMW.Open;
    end
    else if (test = 2) then
    begin
    FormMain.getBTemp1.Open;
    FormMain.getVTemp1.Open;
    FormMain.getFlode1.Open;
    end;
    //showmessage('last_time = '+FloatTostr(last_time_end_channel));
    //showmessage('getVTid =
'+FloatToStr(FormMain.getVTemp.FieldByName('TID').AsFloat));
    //Applikationen väntar tills ett nytt last_time_end_channel har lästs in
    try
    //ÄNDRA HÄR DÅ TEST = 1 ELLER 2
    if (FormMain.getVTemp.FieldByName('TID').AsFloat > last_time_end_channel) then
    //for i:=2 to 17880 do //17984 do
    begin
       if (test = 1) then
       begin
         ok:=false;
        while not ok do begin
        try
         BTemp := StrToFloat(GetVarde(FormMain.getBTemp.FieldByName('värde').AsString));
        {except on E: EConvertError do
         LogEvent(E.Message+': fel vid 1 och Btemp '+FloatToStr(BTemp));
        end; }

       // try
```

```
        effekt := StrToFloat(GetVarde(FormMain.getPMW.FieldByName('värde').AsString));
       { except on E: EConvertError do
        LogEvent(E.Message+': fel vid 1 och PMW '+FloatToStr(effekt));
       end;

       try }
        flode := StrToFloat(GetVarde(FormMain.getFlode.FieldByName('värde').AsString));
       {except on E: EConvertError do
        LogEvent(E.Message+': fel vid 1 och flöde '+FloatToStr(flode));
       end;

       try}
        VTemp := StrToFloat(GetVarde(FormMain.getVTemp.FieldByName('värde').AsString));
       except on E: EConvertError do
        LogEvent(E.Message+': fel vid 1: Vtemp '+FloatToStr(VTemp)+', BTemp
'+FloatToStr(BTemp)+', effekt '+FloatToStr(effekt)+', flode '+FloatToStr(flode));
       end;
       if ((BTemp=-100) or (effekt=-100) or (flode=-100) or (VTemp=-100)) then begin
        ok:=false;
        //LogEvent('val() = -100');
        sleep(5);
       end else
        ok:=true;
       end;
      end
      else if (test = 2) then
      begin
      //Stegar till nästa värde i databasen
      FormMain.getBTemp1.Next;
      FormMain.getFlode1.Next;
      FormMain.getVTemp1.Next;
      BTemp := FormMain.getBTemp1.FieldByName('värde').AsFloat;
      flode := FormMain.getFlode1.FieldByName('värde').AsFloat;
      VTemp := FormMain.getVTemp1.FieldByName('värde').AsFloat;
      end;
     if (test = 1) then
     begin
      try
       last_time_end_channel := FormMain.getVTemp.FieldByName('TID').AsFloat;
      except on E: EConvertError do
       LogEvent(E.Message+': vid last_ 1');
      end;
     end;

     //******Kontroll så ej BT stiger över maxvärdet för anläggning***************
     if (BTemp > bTemp_Max) then
```

```
    begin
      p_counter := p_counter + 1;

      //Ger larm då det inträffat två gånger på rad
      if (p_counter = RUN_test_max) then
      begin
        //Anger att larm inträffat för lagring av larm i databas
        larm := true;
        //Anger vilken typ av larm det är
        larmTyp := 'B_Max överskriden!';
        larmCounter := larmCounter + 1;

        //Beräknar estimerad bärlagertemperatur och prediktionsfelet
        BTemp_est := theta1*(flode/1000 - flode_medel) + theta2*(VTemp - VTemp_medel) +
BTemp_medel;
        err := (BTemp - BTemp_est);

        //Lagra larm i tabellen Larm i databasen BLDB.mdb om överskriden temperatur
        with FormMain.Larm do begin
          Open;
          insert;
          FieldByName('Larm').AsFloat  := larmCounter;
          FieldByName('B_Temp').AsFloat  := BTemp;
          FieldByName('B_Temp_est').AsFloat := BTemp_est;
          FieldByName('Flöde').AsFloat := flode;
          FieldByName('V_Temp').AsFloat  := VTemp;
          FieldByName('Felet').AsFloat  := err;
          FieldByName('Typ').AsString := larmTyp;
          if (test = 2) then
            FieldByName('Tid').AsDateTime := i
          else if (test = 1) then begin
            FieldByName('Tid').AsDateTime := last_time_end_channel;
            {try
              FieldByName('Tid').AsDateTime :=
FormMain.getVTemp.FieldByName('TID').AsDateTime;
            except on E: EConvertError do
              LogEvent(E.Message+': vid last_ 2');
            end;}
          end;
          post;
          close;
        end;
      end;
    end

    //*****Kontroll om aggregat är igång - effekten större än 0.**
```

```
    else if (effekt > 0) then
    begin
      stopCounter := 1;              //Räkaren stoppas och sätts till startvärde

      //****Beräkna skattade bärlagertemperaturen*******************************
      BTemp_est := theta1*(flode/1000 - flode_medel) + theta2*(VTemp - VTemp_medel) +
BTemp_medel;

      //Uppdatera prediktionsfelet err
      err := (BTemp - BTemp_est);

      //CUSUM-algoritmen
      err_pos := err;
      err_neg := -err;
      pos_Cusum_Vector[1] := (pos_Cusum + err_pos - v);
      pos_Cusum_Vector[2] := 0;
      neg_Cusum_Vector[1] := (neg_Cusum + err_neg - v);
      neg_Cusum_Vector[2] := 0;
      pos_Cusum_new := MaxVector(pos_Cusum_Vector);
      neg_Cusum_new := MaxVector(neg_Cusum_Vector);

      //******Kontrollerar ifall CUSUM överstiger gränsvärde
      if ((pos_Cusum_new > h) or (neg_Cusum_new > h)) then
      begin
        s_counter := s_counter + 1;

        //Kollar om det inträffat flera gånger
        if (s_counter = RUN_test) then
        begin
          larm := true;
          //Räknar antal larm som sker
          larmCounter := larmCounter + 1;
          larmTyp := 'Temp. avvikelse!';

          //Lagra larm i databasen BL_DB.mdb
          with FormMain.Larm do begin
            Open;
            insert;
            FieldByName('Larm').AsFloat  := larmCounter;
            FieldByName('B_Temp').AsFloat  := BTemp;
            FieldByName('B_Temp_est').AsFloat := BTemp_est;
            FieldByName('Flöde').AsFloat := flode;
            FieldByName('V_Temp').AsFloat  := VTemp;
            FieldByName('Felet').AsFloat  := err;
            FieldByName('Typ').AsString := larmTyp;
            if (test = 2) then
```

```
            FieldByName('Tid').AsDateTime := i
          else if (test = 1) then begin
            FieldByName('Tid').AsDateTime := last_time_end_channel;
            {try
              FieldByName('Tid').AsDateTime :=
FormMain.getVTemp.FieldByName('TID').AsDateTime;
            except on E: EConvertError do
              LogEvent(E.Message+': vid last_ 3');
            end;}
          end;
          post;
          close;
        end;
        pos_Cusum := 0;
        neg_Cusum := 0;
        pos_Cusum_new := 0;
        neg_Cusum_new := 0;
      end;
    end
    else begin
      if (Abs(err) < 1) then
      begin
        s_counter := 0;
        p_counter := 0;
      end;
      neg_Cusum := neg_Cusum_new;
      pos_Cusum := pos_Cusum_new;
    end;
  end  //if (effekt > 0)

 //****Kontroll om aggregat stoppat - effekten <= 0*************************

  else if (effekt <= 0) then
  begin
    //****Beräkna skattade bärlagertemperaturen*********************************
    BTemp_est := theta1*(flode/1000 - flode_medel)+theta2*(VTemp-
VTemp_medel)+BTemp_medel;
    err := (BTemp - BTemp_est);
    //stopCounter räknar upp tills aggregat startas igen, då sätts stopCounter=1
    stopCounter := stopCounter + 1;
  end;//if (effekt <= 0)

  //Lägger in data i databasen BL_DB.mdb
  with FormMain.Data do begin

    //Om inte max antal rader i BL_DB har nåtts, lägg till
```

```
        if (maxrows_DB > RecordCount) then
        begin
          insert;
          FieldByName('bTemp').AsFloat  := BTemp;
          FieldByName('bTemp_skattad').AsFloat  := BTemp_est;
          FieldByName('Flöde').AsFloat  := flode;
          FieldByName('vTemp').AsFloat  := VTemp;
          FieldByName('felet').AsFloat  := err;
          FieldByName('effekt').AsFloat  := effekt;
          if (test = 2) then
            FieldByName('tid').AsDateTime := i
          else if (test = 1) then begin
            FieldByName('Tid').AsDateTime := last_time_end_channel;
            {try
              FieldByName('tid').AsDateTime :=
FormMain.getVTemp.FieldByName('TID').AsDateTime;
            except on E: EConvertError do
              LogEvent(E.Message+': vid last_ 4');
            end;}
          end;
          if larm then
          begin
            FieldByName('larm').AsFloat  := larmCounter;

            //För att motverka att det alltid står larm i databasen
            larm := false;
          end
          else begin
            FieldByName('larm').AsFloat  := 0;
          end;
          post;
          //Om max antal rader i BL_DB.mdb nåtts, hoppa till första raden
          if (maxrows_DB <= RecordCount) then
          begin
            first;
          end;
        end

        //Om max antal rader i BLDB, skriv över rad och flytta fram till nästa rad
        else begin

          //Om sista raden i BLDB skrivits över, flytta till första raden
          if (FormMain.Data.Eof) then
          begin
            first;
            edit;
```

```
          FieldByName('bTemp').AsFloat  := BTemp;
          FieldByName('bTemp_skattad').AsFloat  := BTemp_est;
          FieldByName('Flöde').AsFloat  := flode;
          FieldByName('vTemp').AsFloat  := VTemp;
          FieldByName('felet').AsFloat  := err;
          FieldByName('effekt').AsFloat  := effekt;
          if (test = 2) then
          begin
            FieldByName('tid').AsDateTime := i;
          end
          else if (test = 1) then
          begin
            FieldByName('Tid').AsDateTime := last_time_end_channel;
            {try
              FieldByName('tid').AsDateTime :=
FormMain.getVTemp.FieldByName('TID').AsDateTime;
            except on E: EConvertError do
              LogEvent(E.Message+': vid last_ 5');
            end;}
          end;
          if larm then
          begin
            FieldByName('larm').AsFloat  := larmCounter;
            larm := false;
            end
          else begin
            FieldByName('larm').AsFloat  := 0;
          end;
          post;
          next;
        end
        else begin
          edit;
          FieldByName('bTemp').AsFloat  := BTemp;
          FieldByName('bTemp_skattad').AsFloat  := BTemp_est;
          FieldByName('Flöde').AsFloat  := flode;
          FieldByName('vTemp').AsFloat  := VTemp;
          FieldByName('felet').AsFloat  := err;
          FieldByName('effekt').AsFloat  := effekt;
          if (test = 2) then
          begin
            FieldByName('tid').AsDateTime := i;
          end
          else if (test = 1) then
          begin
            FieldByName('Tid').AsDateTime := last_time_end_channel;
```

```
            {try
              FieldByName('tid').AsDateTime :=
FormMain.getVTemp.FieldByName('TID').AsDateTime;
            except on E: EConvertError do
              LogEvent(E.Message+': vid last_ 6');
            end;}
          end;
          if larm then
          begin
            FieldByName('larm').AsFloat  := larmCounter;
            larm := false;
          end
          else begin
            FieldByName('larm').AsFloat  := 0;
          end;
          post;
          next;
        end;
      end; //if (maxrows_DB > RecordCount)
    end; //With Form.Data Do
  end;//if (FormMain.getBLTemp.FieldByName('INDEX').AsInteger <> index_BL)
  Except on E: EConvertError do
    LogEvent(E.Message+': If time > last_time');
  end;
  if (test = 1) then
  begin
  FormMain.getBTemp.Close;
  FormMain.getVTemp.Close;
  FormMain.getFlode.Close;
  FormMain.getPMW.Close;
  end
  else if (test = 2) then
  begin
  FormMain.getBTemp1.Close;
  FormMain.getVTemp1.Close;
  FormMain.getFlode1.Close;
  end;
  FormMain.Data.Close;
  if (test = 1) then
    sleep(5000)
  else
    showmessage('slut i rutan');
 end;//while (Terminated = false)
 //FormMain.Data.Close;
end;//TFormMain.BStartClick
```

```
procedure LogEvent(Msg: String);
(******
  Procedur för att skriva ner när sker och ting gått fel
******)
Const
  FileName = '.\Handelselogg.Log';
Var
  aFileStream: TFileStream;
  TmpStr : String;
  I : Integer;
begin
  If Not FileExists( FileName ) Then Begin
    Try
      aFileStream := TFileStream.Create(FileName, fmCreate);
      aFileStream.Free;
      Except
    End;
  End;
  Try
    aFileStream := TFileStream.Create(FileName, fmOpenReadWrite OR fmShareDenyWrite);
    Try
      I := CompareStr('Invalid filename', Msg);
      If I <> 0 Then Begin

      //aFileStream.Seek(soFromBeginning, aFileStream.Size);
      If aFileStream.Size > 20000 then begin
        Try
          aFileStream.Destroy;
          aFileStream := TFileStream.Create(FileName, fmCreate);
          aFileStream.Free;
          Except
        End;
      end;
      aFileStream.Position := aFileStream.Size;
      //Raderna formatteras enligt: "2003-04-02 10:05:23 : Msg"
      TmpStr := Format('%s %s : %s%s', [DateToStr(Now), TimeToStr(Now), Msg, #13#10]);
      aFileStream.Write(TmpStr[1], Length(TmpStr));
      End;
      Except
    End;
    Finally
      aFileStream.Free;
  End;

end;
```

```
//Funktion för att kontrollera värdet som läses in från kraschdumpen.
Function GetVarde(varde:String):String;
var
  x:Extended;
  p:integer;
begin
  //Kontrollerar om strängen som skickas in innehåller tomma tecken (mellanslag)
  while (Pos(' ',varde)<>0) do
    Delete(varde,(Pos(' ',varde)),1);

  //Gör om komman till punkter för rätt format i Val()-funktionen
  If Pos(',',varde)<>0 Then
  begin
    Insert('.',varde,(Pos(',',varde)));
    Delete(varde,(Pos(',',varde)),1);
  end;

  //Kontroll om varde är ett giltigt integer el. flyttal
  //om inte returneras en nolla.
  Val(varde,x,p);
  If p<>0 Then begin
    LogEvent('varde i Val = '+varde);
    varde:='-100';
  end;

  //Gör om punkter till komman
  If Pos('.',varde)<>0 Then
  begin
    Insert(',',varde,(Pos('.',varde)));
    Delete(varde,(Pos('.',varde)),1);

  end;
  //Returnerar värdet som en sträng
  GetVarde := varde;
end;

end.
```