# ANNEXE 1

## Kalman filtering of the DACFOS model

C. Ø. Jensen

# Kalman Filtrering af DACFOS modellen.

10. juni 1998

## 1 Den modellerede proces.

Den måde Kalman Filtret (KF) bruges til i vores setup er til såkaldt parameter-estimation. Dette betyder at vi bruger KF til at bestemme et antal ubekendte parametre i en funktional som vi ønsker at bruge til at modellere vores proces: Altså Ozon-koncentrationen i Jægersborg. Over en periode (4 dage før forcast-/analysetidspunktet i det nuværende setup) opbygges filtret i en iterartiv proces, beskrevet ved ligningerne (1-5)nedenfor, således at de resulterende parametre er optimerede under den antagelse at vi simulere en essentielt Markovsk[1] proces. Derefter benyttes til korrektion af DACFOS' forcast ved succesiv anvendelse af den modellen med de fundne parametre for hver forcast tidspunkt. Modellen vi ønsker at fitte til vores dynamiske process er givet ved ligning (6 )nedenfor. Her er matricen $H_k$ en funktion af de eksterne variable vi ønsker at inkludere dynamikken af. Hvis vi fx. ønsker at anvende vindstyke($v_k$), temperatur ($T_k$), og modelleret ozon i en fast niveau ($O_{3,k}$) vil modellen være

$$z_k = v_k x_k[1] + T_k x_k[2] + O_{3,k} x[3],$$

enhver anden model af denne type er naturligvis mulig, blot man formår at udtrykke en lignende funktionel sammenhæng mellem observation og tilstandsvektor og at sammenhængen er lineær. Dette sidste er naturligvis en begrænsning af filterets modelleringsegenskaber. Ønskes mere komplekse sammenhænge må man benytte mere avancerede filtre som fx. 'the Extended Kalman Filter'. Kalman Filter biblioteket kan bruges til dette også, men gøres det pt. ikke.

### 1.1 Ligningerne.

Det diskrete, lineære Kalman Filter består af følgende ligninger:

$$x_{k+1}^{-} = A_k x_k + B u_k \tag{1}$$

[1] Det er netp denne antagelse der udtrykkes i ligning 1

$$P_{k+1}^{-} = A_k P_k A_k^T + Q \tag{2}$$

$$K_k = P_k^{-} H_k^T \left( H_k P_k^{-} H_k^T + e^{-1/\tau} R \right)^{-1} \tag{3}$$

$$x_k = x_k^{-} + K \left( z_k - H_k x_k^{-} \right) \tag{4}$$

$$P_k = e^{-1/\tau} \left( I - K_k H_k \right) P_k^{-} \tag{5}$$

,hvor ligning (1) er *a priori* tilstandsopdaterings formlen, ligning (2) giver en *a priori* extrapolation af model usikkerheds covariance matricen, ligningen for $K$ er det såkaldte 'Kalman Gain' og (4) og (5) er *a posteriori* updateringer af tilstandensvetoren og covarians matricen. Filtret benytter endvidere den af brugeren supplerede model for sammenhængen mellem målinger og tilstandsvektoren, $x_k$, givet ved

$$z_k = H_k x_k \tag{6}$$

# 2 Estimerings metodik.

For at opbygge filtret og finde det ovennævnte sæt parametre, $x$, skal bruges to tidsserier fra indlæringsperioden. En fil med værdierne af de uafhængige variable, der indgår i vores model, kronologisk listet for hele perioden. Denne tidsserie skal også indeholde de forcastede værdier, som senere skal korrigeres ved brug af modellen. Den anden fil skal indeholde målinger af de observerede størrelser – de afhængige variable – der gennem ligning 6 ønskes modelleret. I vores model er denne identisk med ozon-målingerne i Jægersborg. Data i denne fil skal helst foreligge indtil forecastorigo. Hvis tidsserien er trunkeret før dette tidspunkt vil filtret stoppe indlæringen her og derefter gå i ren forecast 'mode'. Hvis der derimod er data helt ind i eller for hele forecast perioden, vil filtret alligevel skifte til forecast mode ved forescast-origo, men så ydermere benytte de resterende målinger til at beregne et antal statistiske mål.

Jvf. beskrivelsen af selve KF biblioteket (se note om dette) er det ikke helt trivielt at holde styr på processeringen af dataflowet til filtreringen. Filteret selv har ikke noget 'tidsbegreb' og benytte derfor rimelig ukritisk de sidst supplerede data i det næst forestående trin. Det genbruger dog ikke data, medmindre de fødes til filtret igen. Problematikken opstår specielt fordi data i de to input filer ikke er begænset til en bestemt regularitet eller samhørighed i forekomst. Man kan således fint forestille sig at benytte (hvad vi allerede gør) modeldata hver tredie time sammen med 10 minutters observationer.

Da selve filter biblioteket er lavet helt generelt er det ikke muligt (med denne forfatters begrænsede viden) at lave en god, generel håndtering af det omtalte

tidsbegreb. Derfor skal der nødvendigvis laves et/flere program(-mer) til håndtering af dette og andre specifikke problemer omkring I/O og postprocessering af data, til et konkret problem. Det er beskrivelsen af dette setup der er hovedformålet med denne note.

# 3 Setup af KF for DACFOS ozon i Jægersborg.

I Appendix 1 er beskrevet alle programmer o.a. omkring afviklingen, præ-og post-processeringen af KF og deres funktion i kort oversigts form. Vi vil her koncentrere os om de funktionaliteter der ligger i selve filtreringsprogrammet. Programmet er pt.(7/3-97) stadig i udviklingsfasen og existerer derfor i flere parallelle versioner. Der er således en version der blot hedder *forecast* som ligger i 'bin' direktoriet, som indikerer at dette er det operationelle system. Det er dog en ikke særlig moden version, som primært tjener til at teste automatikken i systemet når det køres via. crontab'en. Den version der er beskrevet her ligger placeret i 'test_bin' direktoriet og det er executable'en *filter_tests* og det tilhørende script *run_filter_tests.scr*, der bruger til at afvikle førstnævnte.

For at holde styr på de omtalte logistiske problemer omkring rækkefølgen af processeseringen af data, må vi operere med 3 tidspunkter i programmet; Det nuværende filtertidspunkt og de 2 næste hhv. model- og observationstidspunkter. De to sidste bruges til at holde styr på om nye data skal indlæses eller om de nuværende værdier stadig er for nye til at dette er aktuelt og vi derfor skal foretage endnu en iteration med 'den anden del' (opdateringen eller predikteringen jvf. Kalman bibliokets dokumentationen), inden vi tjekker dette igen.

Da der desværre (endnu) ikke er fundet en entydigt bedste model er filteret bygget så man kan experimentere med flere modeller, ud fra en antal basiskomponenter, der er indbygget i modellen. Positivt set kan man vælge at betragte det således at filteret yder tilstrækkelig flexibilitet til at forskellige modeller kan benyttes på fx. forskellige ugedage og/eller måneder, alt efter hvilke processkomponenter man finder spiller den største rolle aktuelt. Modellen sammenstykkes ved at benytte argumentet '-s' for hver af de basis komponenter man vil have med. Mulige numre er:

| Argument | Model funktion |
|---|---|
| -s 1 | Anvend 'normalen' af DACFOS' 10 niveauer |
| -s 2 | Anvend DACFOS's 10 niveauer |
| -s 3 | Benyt HIRLAM temperaturer |
| -s 4 | Benyt HIRLAM vindhastighed |
| -s 5 | Brug sinusiodal dagsvariation |
| -s 6 | Tilføj constant offset |

Option '-s 1' implicerer automatisk '-s 2'. Disse 6 muligheder vil dog relativt kunne udvides/modificeres. Det kræver dog god forståelse for selve C-kodens

program-flow og de enkelte variables anvendelse (En indbygget guide ligger i at følge fx. den specifikke kode for en af de andre options og så lave tilsvarende kode for sin nye 'byggesten').

Foreløbige kørsler tyder på at kombinationen '-s 1 -s 3 -s 5' er rimelig god. Man bør dog overveje hvor megen succes man kan forvente af filteret, idet det som sagt forsøger at fange en sandsynligvis ikke-lineær process med en lineær model. Endvidere er målingerne jo i sagens natur lokale mens DACOS modelresultaterne for ozon og HIRLAMs vinde gælder for mesoscala niveau.

## 3.1 Output data.

Outputtet fra filteret er en tidsserie med modelresultatet for både indlæringsperiode og forecast, beregnet med de fundne koefficienter. Formatet af denne fil er det samme som for input filerne (kf.obs og kf.model) og ligger placeret i ./data direktoriet. For at kunne gemme flere kørsler sammen i data direktoriet, kan man vælge et suffix, der vil blive appended til filnavnet vha. option '-x' Fx. kunne man for en kørsel med ovennævnte 4 gode modelkomponenter passende bruge '-x s135'. I det nuværende test setup (se Appendix 2) laves mange forskellige kombinationer af diverse parametre og midling af observationer så denne option er god til at holde styr på disse detaljer. Desuden er det med programmet *forecast_view* muligt at sætte en give en stribe af disse extensions som option, så man kan bladre i flere forecasts interaktivt og dermed let sammeligne resultater, med forskellige modelsetups.

Derudover udskrives en mængde statistiske mål. Hvis der i observationsfilen var måledata for selve forecast perioden, beregnes fejl, correlation og signifikans af disse og resultatet udskrives til 'stderr'. Derudover udskrives statistik for sekvensen af innovationer (udtrykket i parentes i ligning 4) under indlæringen. Disse kan bruges til at vurdere om filteret potentielt er 'skævt'. Se iøvrigt appendix 3 for en udførlig testkørsel.

## 3.2 Andre parametre.

Performance af filteret afhænger, ud over regulariteten i observationerne, også af de 2 usikkerhedsmatricer $Q$ og $R$, som angiver hhv. usikkerheds covariance matricerne for model og observationer. Disse antages at være diagonalmatricer med samme værdi i alle diagonalelementer. Derfor kan de sætte med blot 2 options '-q' og '-r'. Antagelserne om ukorrelerede rækker og søjler og specielt om at variancen er ens for alle komponenter er naturligvis ikke strengt korrekt, men følsomheden er så relativt lille at det ikke er afgørende for resultatet (det er ihvertfald den foreløbige konklusion).

### 3.3 Run scriptet.

Som nævnt styres kørslen af scriptet run_filter_tests.scr (se Appendix 2 for en udskrift af dette). Dette tager som eneste argument '-b YY:MM:DD:HH' origo for forecasten. Hvis argumentet undlades vil det køre det nyest mulige tidspunkt. På den måde vil det umiddelbart kunne kaldes fra et crontab script. Scriptet trækker derefter de relevante data frem fra databaserne vha. programmet *get_kalman_data* og efter at have forberedt data til at kunne bruge Kalman Filtrerede Hirlam data og midlede observationer køres de induviduelle kørsler. Slutteligt komprimeres outputte og overflødigt output slettes.

## 4 Postproccesering af output.

Med alle de indstillingsmuligheder og da hvert valg kan have en vis grad af succes under specielle forhold er det nødvendigt at have nogle stærke værktøjer til at assistere med at vurdere og udvælge dem med den bedste performance. Dette gælder både mhb. på grafisk fremstilling og samlende statistik for flere kørsler ogver længere perioder. Nogle få utilities er lavet til dette, men flere bør fremstilles.

Til at se resultatet af den færdigt 'optimerede' model er udviklet programmet *forecast_view*. Med dette kan man som tidligere nævnt se flere KF forecast (baseret paa '-x' argumenterne) samtidigt med de observerede målinger og resultatat for DACFOS' individuelle niveauer og middel.

# Appendix 1. Relaterede programmer o.a.

**Datageneratorer** Programmer til generering af rå data til brug for KF.

| Program | Udvikler | Funktion |
|---|---|---|
| DACFOS | (jhs) | Laver bl.a. $O_3$ forecasts i pt. 10 niveauer i PBL<br>Laver også trajektorie filer med HIRLAM temp og vind |
| get_jgb_data | (ali) | script er ftp'er målinger fra Jægersborg |
| 'gts-systemet' | (V,kec) | Henter sonde data fra det globale GTS netværk |

**Environment** variable brugt til at lokalisere data

| Navn | Beskrivelse |
|---|---|
| DACFOSHOME | Direktorie hvor DACFOS output og trajektorie data ligger |
| METHOME | Synop database placering |
| JGBHOME | Direktorie hvor Jægersborg data er lagret. |
| KFHOME | Bruges af postprocessor for at finde KF output filer |
| KFEXTENSION | default suffix/extension brugtaf forecast_view programmet |

**Pre-processering** Disse programmer benyttes til at generere inddata filerne i de rette formater.

| Program | Placering | Beskrivelse |
|---|---|---|
| get_kalman_data | ./bin | Henter model og observationsdata ud fra databaser. |
| forecast_met | ./bin | Bruges til at kalmanfiltere HIRLAM temp og vinde. |
| means | $HOME/bin | Bruges til at lave running mean af observationerne |
| date_merge | $HOME/bin | Bruges til at stykke to filer sammen kronologisk |

**Input** Data til brug for KF skal foreligge i et specifikt format og i filerne kf.model og kf.obs. Første kolonner er år måned dag time minut. Derefter følger et antal kolonner med hhv. modeldata eller observationer. Se appendix 3 for eksempler på formatet.

**Filteret**

| Program | Placering | Funktion |
|---|---|---|
| filter_tests | ./test_bin | Selve filteret. Kaldes af nedenstående |
| run_filter_tests.scr | ./test_bin | Script til styring af logistikken omkring KF |
| forecast | ./bin | Testprogram i semi-operationelt brug. |

**Output** Det output der ikke skrives direkte til stderr, og derfor typisk pipes til en 'log'-fil, bliver skrevet i et ./data underdirektorie til det hvor programmet exekveres fra. Dette er dog typisk et link til et andet direktorie.

**Post-processering** Følgende programmer og utilities kan bruges til at analysere output og performance af filteret. Flere bør dog udvikles specielt let overskuelig grafik for fx. månedsresultater.

| Program | Placering | Funktion |
|---|---|---|
| libkf.so | $HOME/lib | Bibliotek til at indlæse data fra filtrering |
| forecast_view | ./bin | GUI til at se resultat af DACFOS og KF samt obs'er |
| dacfos_view | $HOME/bin | GUI til kun at se DACFOS + observationer |
| test_table.awk | ./test_bin | awk script der putter output til stderr i tabelform |
| get_correl.scr | ./test_bin | Finder bedste correlation og mindste rms pr. dag |
| get_error.scr | ./test_bin | Finder succesraten af KF vs. DACFOS pr. md. |

# Appendix 2. Run-script udskrift

Her følger en udskrift af run_filter_tests.scr .

```
#!/bin/tcsh
#
# $Id$
#
set prg     = $0
set filter  = ./filter_tests
set daytime = `date '+%y%m%d00'`
set period  = 4
set Q_value = 1
set R_value = 100

while ( $#argv )
  switch ($argv[1])
  case -b:
    set daytime = $2
    shift
    breaksw
  default:
    echo "Usage: $prg:t [ -b YYHHDD(00|12) ]"
    echo ""
    echo "    -b  :  forecast origo n format year:month:day:hour, where hour is one of 00 and 12"
    echo "           e.g. to get forecast starting at midday oct. 1st 1996: "
    echo "               > $prg:t -b 96100112 "
    exit 1
    breaksw
  endsw
  shift
end

set year    = `echo $daytime | cut -c1-2`
set month   = `echo $daytime | cut -c3-4`
set mday    = `echo $daytime | cut -c5-6`
set hour    = `echo $daytime | cut -c7-8`
set length  = `echo $daytime | wc -c`
if (${length} < 9) exit 1
set logfile = tests/${daytime}_log
set dayarg  = "${year}:${month}:${mday}:${hour}"
set daybeg  = `/pack/local/bin/date -d "${year}-${month}-${mday}  ${period} days ago"\
              "+%y:%m:%d:${hour}"`
#
#  retrieve relevant data for calculations
#
../bin/get_kalman_data -b $daybeg -o $dayarg
#
# kalman filter hirlam wind speeds and temp
#
../bin/forecast_met -o $dayarg -H .00005 -w 5.0 -T .0005 -t 20.0 >& $logfile
#
# calculate 3 hour centered running mean observation
#
cat kf.jgb_${daytime} | means -i 10800 > kf.jgb_ave_${daytime}

#--------------------------------------
#  Do the 20 (!) different filterings
#--------------------------------------
echo '======>  10 minutes obs.';
set echo
/bin/cp kf.jgb_${daytime} kf.obs
date_merge  kf.hirlam_${daytime} kf.dacfos_${daytime} > kf.model
$filter -s 2 -s 3 -o $dayarg -q e ${Q_value} -r ${R_value} -x h_rawobs_s23 >>& $logfile
$filter -s 2 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s25 >>& $logfile
$filter -s 3 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s3 >>& $logfile
$filter -s 3 -s 6 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s36 >>& $logfile
$filter -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s5 >>& $logfile
$filter -s 1 -s 2 -s 3 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s123 >>& $logfile
$filter -s 1 -s 2 -s 3 -s 6 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s1236 >>& $logfile
$filter -s 1 -s 2 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s125 >>& $logfile
$filter -s 2 -s 3 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s235 >>& $logfile
$filter -s 1 -s 2 -s 3 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_s1235 >>& $logfile

/bin/cp kf.jgb_ave_${daytime} kf.obs
$filter -s 2 -s 3 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s23 >>& $logfile
$filter -s 2 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s25 >>& $logfile
$filter -s 3 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s3 >>& $logfile
$filter -s 3 -s 6 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s36 >>& $logfile
$filter -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s5 >>& $logfile
$filter -s 1 -s 2 -s 3 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s123 >>& $logfile
$filter -s 1 -s 2 -s 3 -s 6 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s1236 >>& $logfile
$filter -s 1 -s 2 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s125 >>& $logfile
$filter -s 2 -s 3 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s235 >>& $logfile
$filter -s 1 -s 2 -s 3 -s 5 -o $dayarg -q ${Q_value} -r ${R_value} -x h_rawobs_ave_s1235 >>& $logfile
```

```
unset echo

#
# compress log file
#
gzip -f -9 $logfile
#
# compress datafiles
#
gzip -f -9 tests/jgb_${daytime}*
#
#  clean up
#
rm  *_${daytime}*
#
rm *predict* *update*
exit
```

# Appendix 3. Eksempel på input og output.

Vi skal her vise alle de komandoer og eksempler på datafiler og output for en kørsel med origo den 9 August 1996 kl. 00:00. Sammenlign med kommandoerne i udskriften i Appendix 2.

Forløbet vil svare til et kald til run_filter_tests.scr med følgende argument:

```
1 >run_filter_tests.scr -b 96080900
```

Først skal de relevante data hentes fra de respektive databaser/direktorier. Da vi vil indlære filteret over en 4 dags periode sker det med følgende kald.

```
2>get_kalman_data -b 96:08:05:00 -o 96:08:09:00
```

Vi har nu få}et lavet 4 filer med data genereret: kf.dacfos_96080900, kf.hirlam_96080900, kf.jgb_96080900 og kf.synop_96080900.

En 'head -3 *96080900* giver følgende udskrift, der giver et indtryk af dataformatet og indhold:

```
==> kf.dacfos_96080900 <==
1996  08   5   0   0 21.352 22.543 19.134 18.318 17.022 16.176 15.759 15.792 15.983 15.896
1996  08   5   3   0 8.559 6.918 5.952 6.054 6.134 5.102 4.145 3.552 2.521 0.041
1996  08   5   6   0 9.438 9.881 9.421 9.798 9.199 9.39 9.562 9.921 10.838 11.339

==> kf.hirlam_96080900 <==
96 8 5 0 0 11.75 0.842912
96 8 5 3 0 10.85 1.38946
96 8 5 6 0 15.25 1.55425

==> kf.jgb_96080900 <==
1996  08   5   0   0 -4.5
1996  08   5   0  10 -4.4
1996  08   5   0  20 -1.6

==> kf.synop_96080900 <==
1996  08   5   0   0 10.2 0.54
1996  08   5   1   0 9.8 1.08
1996  08   5   2   0 9.6 0.54
```

Dette er det generelle format at tidsseriedata som både filteret og date_merge og kalman indlæ}sebiblioteket som bruges at *forecast_view* anvender. Vi undlader at kalmanfiltrere HILRLAM vindene og temperaturer da dette har vist sig af sekundær betydning. Istedet går vi direkte til 3 timers( = 10800 secs.) centreret running mean af observationerne som sker vha. *means* filteret:

```
3 > cat kf.jgb_96080900 | means -i 10800 >
kj.jgb_ave_96080900
```

Da filteret vil have observationerne i en fil med et bestemt navn kopierer vi lige dem til denne:

```
4 > cp kf.jgb_96080900 kf.obs
```

Filteret skal som sagt også have en fil med model data. Denne sammensættes af to filer ved hjælp af date_merge, der 'merger' filer kronologisk med et format som overfor vist.

```
5> date_merge kf.hirlam_96080900 kf.dacfos_96080900 >
kf.model
```

Herefter ser de første 3 linier i kf.model således ud:

```
1996 8 5 0 0  11.75 0.842912 21.352 22.543 19.134 18.318 17.022 16.176 15.759 15.792 15.983 15.896
1996 8 5 3 0  10.85 1.38946 8.559 6.918 5.952 6.054 6.134 5.102 4.145 3.552 2.521 0.041
1996 8 5 6 0  15.25 1.55425 9.438 9.881 9.421 9.798 9.199 9.39 9.562 9.921 10.838 11.339
```

Sammenlign med kf.hirlam_96080900 og kf.dacfos_96080900 ovenfor. Dermed er vi klar til at køre selve filteret. Vi vælger en model med DACFOS normalen og HIRLAM temperaturen. modelvariansen, q, sætter vi til 1.0 og observationsvariansen sættes til 100. Disse værdier er temmeligt arbitrært valg og burde principielt underkastes en mere rigorisk vurdering, men som flere tekster omkring Kalman Filtrering siger, er det praksis at bruge disse som justeringsparametre og blot finde nogle gode værdier der giver et stabilt performance. Vi vælger også at bruge de rå, umidlede observationsdata. Udfra disse vælges file-extension og kommandolinien kommer til at se således ud:

```
6 > filter_tests -s 1-s 3 - o 06:08:09:00 -q 1 -r 100 -x
h_rawobs_s13
```

Kommandoen giver følgende udskrift til stderr:

```
origo of forecast : 96:08:09:00
q = 1 r = 100
Using :
        use_mean 1
        use_df  1
        use_temp 1
        use_wind 0
        use_sinus 0
        use_const 0
        +h      KF        DACFOS
ozone:  3    21.4277     38.539     22.8
ozone:  6    21.9428     35.103     17.7
ozone:  9    27.5627     41.267       25
ozone:  12   34.2662     47.759     44.2
ozone:  15   36.3192     84.842       49
ozone:  18   29.1786     62.542       34
ozone:  21   22.6305     23.545     34.6
ozone:  24   22.4128     60.651     29.1
ozone:  27   21.2104     39.662     22.7
ozone:  30   22.6018     31.898     23.1
ozone:  33   29.3773     28.008     29.1
ozone:  36    32.449     27.966     41.8
forecast statistics   : --- 12  points ---
mean error    : 5.49054     16.7203
mean bias     : -4.31008     12.3902
correlation   : 0.8656      0.4695
significance : 0.0002746      0.1236
Fishers z     :  1.315      0.5094
innovation statistics   : --- 32 points ---
ave  = -0.549548
adev = 3.39705
sdev = 4.54759
var  = 20.6806
skew = -0.151059
curt = 1.18081
```

Bemærk at kolonnen med DACFOS data er normaler og ikke selve det semi-operationelle resultat, der er en ren midling over alle modelniveauer. Resultatet af DACFOS normalen vist ovenfor var ikke synderlig forskellig fra den rigtige

DACFOS værdi, dog en smule bedre i rms og lidt dårligere correlation. Læg også mærke til at KF faktisk gør det ret fortrinligt i dette testeksempel.

Vi kan nu kigge på vores resultat grafisk sammen med DACFOS' resultat og observationsdata med forecast_view. Efter at have sikret os at environment variablene er sat korrekt kan vi kalde vieweren med

```
7 > forecast_view -o 96:08:09:00 -x 'h_rawobs_s13'
```