EXERCISE 8: Forecast Cycle Observation Impacts

Computation of the observation impacts for the forecast cycle involves multiple steps, so it is important that you follow the instructions below *carefully* and perform all the steps described in the order in which they are presented. Please do not deviate from these instructions.

Description of the problem

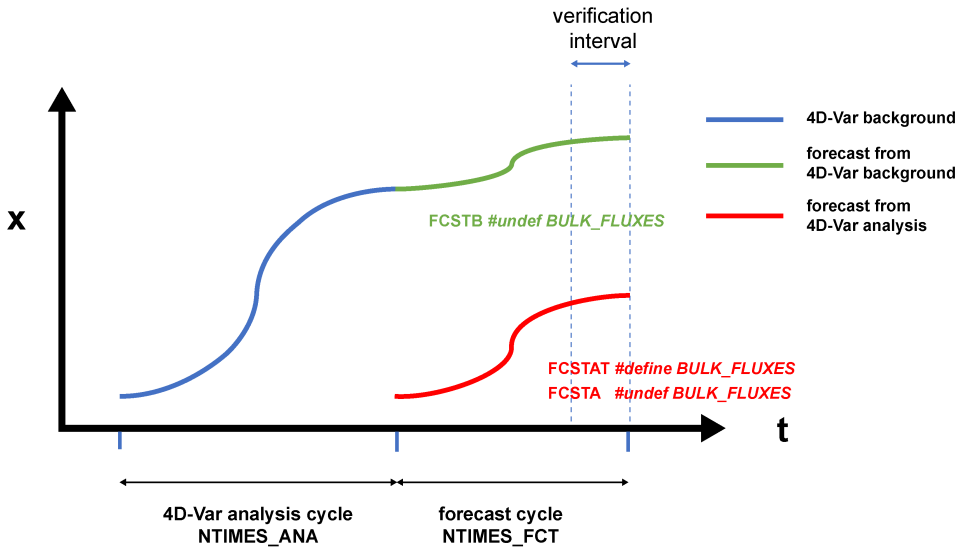

Figure 1: A schematic of a typical operational analysis-forecast cycle. During the analysis cycle, an ocean state estimate is computed using 4D-Var to assimilate all available observations. The blue curve represents the *background* circulation, $\mathbf{x}_b$, for this cycle and is derived from the state estimate from the previous 4D-Var cycle. The number of time steps during the analysis cycle is given by **NTIMES_ANA**. At the end of the analysis cycle there are two possible forecasts: **FCSTA** - the red forecast which is initialized using the state estimate at the end of the analysis cycle, and **FCSTB** - the green forecast which is an extension of the 4D-Var background into the forecast cycle. The number of time steps during the forecast cycle is **NTIMES_FCT**. These two forecasts can be verified against either a new analysis or against new observations during the "verification interval." The red forecast **FCSTA** has received the benefit of the observations assimilated during the analysis interval, while the green forecast **FCSTB** has not. Therefore, the difference in forecast error between **FCSTA** and **FCSTB** can be used to quantify the impact of the observations assimilated during the analysis cycle on the subsequent forecast skill of **FCSTA**.

Figure 1 shows a schematic representation of a typical analysis-forecast cycle using ROMS. During the analysis cycle a background solution (represented by the blue line) is corrected using observations that are available during the analysis window. The analysis state estimate at the end of the analysis window is then used as the initial condition for a forecast. The forecast generated during the forecast cycle from the 4D-Var analysis is represented by the red curve. The skill of the forecast on any day can then be assessed by comparing it to a new analysis that verifies on the same forecast day, or by comparing the forecast with new observations that have not yet been assimilated into the model.

To quantify the impact of the observations on the forecast skill it is necessary to run a second forecast which is initialized from the background solution at the end of the analysis cycle. This is represented by the green curve in Fig. 1. The difference in forecast skill between the red forecast and the green forecast is then due to the observations that were assimilated into the model during the analysis cycle.

The methodology will be described first for a standard generic quadratic forecast error metric given by:

$$e = (\boldsymbol{x}_f - \boldsymbol{x}_t)^T \boldsymbol{C} (\boldsymbol{x}_f - \boldsymbol{x}_t) \tag{1}$$

where $\boldsymbol{x}_f$ denotes the forecast state-vector, $\boldsymbol{x}_t$ denotes the true state-vector, and $\boldsymbol{C}$ is a weight matrix. For example, if $\boldsymbol{C}$ is a diagonal matrix with elements equal to 1 corresponding to all surface temperature grid points, and zero elsewhere, then e would represent the sum of the squared errors in SST. Forecast error metrics of the form (1) are very common in numerical weather prediction and oceanography, so (1) is a good starting point. In practice, however, the true state $\boldsymbol{x}_t$ will never be known, so the forecast error (1) is usually computed relative to a verifying analysis $\boldsymbol{x}_a$, in which case:

$$e = (\boldsymbol{x}_f - \boldsymbol{x}_a)^T \boldsymbol{C} (\boldsymbol{x}_f - \boldsymbol{x}_a) \tag{2}$$

where $\boldsymbol{x}_a$ denotes the verifying analysis at the appropriate forecast time.

These days operational weather prediction models generally yield very high-quality analyses, so the assumption that $\boldsymbol{x}_a$ is a reasonable approximation for $\boldsymbol{x}_t$ is probably reasonable. In oceanography, however, this is a more questionable assumption, so when possible, it may be more prudent to verify a forecast against independent observations, or observations that have not yet been assimilated into the model. In this case, (2) would be reformulated as:

$$e = (\boldsymbol{y}_f - \boldsymbol{y})^T \boldsymbol{C} (\boldsymbol{y}_f - \boldsymbol{y}) \tag{3}$$

where $\boldsymbol{y}_f$ is the model forecast of the vector of verifying observations $\boldsymbol{y}$.

Case 1: Measuring observation impacts using a verifying analysis

In the first example, the 4-day analysis cycle of Exercise 3 is used which spans the 4-day period midnight on 3 Jan 2004 to midnight on 7 Jan 2004. The forecast interval spans the 7-day period midnight on 7 Jan 2004 to midnight on 14 Jan 2004. We will use (2) as a measure of the forecast error and choose $\boldsymbol{C}$ so that e is the forecast error in the 37ºN transport averaged over forecast day 7 as measured against a verifying analysis. This is indicated as the "verifying interval" in Fig. 1. The verifying analysis $\boldsymbol{x}_a$ was computed by running a second 4D-Var analysis cycle for the 7-day window 7-14 Jan 2004. This has been precomputed for you in these examples.

First, compute the verifying analysis. It is computed by running a second RBL4DVAR-RPCG, but for seven days (Jan 7-14, 2004) by assimilating new observations (**../Data/forecast_obs.nc**).

Go to the directory **WC13/RBL4DVAR/VERIFYING_ANALYSIS** and run RBL4DVAR-RPCG initialized from Exercise 3 **wc13_dai.nc** stored in **EX3_RPCG**.

Then, to run this example, go to the directory **WC13/RBL4DVAR_forecast_impact**

Step 1:
Go next to the subdirectory **FCSTAT.** In this first step you will run the forecast initialized from the 4D-Var analysis computed in Exercise 3 using RBL4DVAR-RPCG (**EX3_RPCG**). This is the red curve in Fig. 1. As in previous exercises, edit the **build_roms.csh** script as necessary and compile the model. Before running the forecast, be sure to run the script **job_fcstat.csh**. It computes the surface fluxes used in **FCSTA** and **FCSTB** to guarantee identical surface forcing.

Step 2:
Go next to the subdirectory **FCSTA**. In this second step it is necessary to run the forecast in step 1 again, but this time without **BULK_FLUXES** and using instead the fluxes computed in **FCSTAT** during the forecast. This step is necessary because the red forecast and green forecast in Fig. 1 ***must*** be subject to the same surface and lateral boundary conditions. In practice, the difference between the forecast skill of **FCSTAT** and **FCSTA** will be small.

As in previous exercises, edit the **build_roms.csh** script as necessary and compile the model. Before running the forecast, be sure to run the script **job_fcsta.csh**.

Step 3:
Go next to the subdirectory **FCSTB**. In this first step you will run the forecast initialized from the 4D-Var background solution at the end of the 4D-Var window. Before doing this, you must first run the Matlab script **create_ini_fcstb.m** to create the forecast initial condition NetCDF file using the background solution of RBL4DVAR-RPCG in Exercise 3 (**EX3_RPCG**). As in step 2, this forecast will use the surface fluxes computed during step 1. As in previous exercises, edit the **build_roms.csh** script as necessary and compile the model. Before running the forecast, be sure to run the script **job_fcstb.csh**.

Step 4:
If we denote the forecast error of the red forecast in Fig. 1 by e_r and the forecast error of green forecast by e_g then as discussed in Lecture 5, the forecast error difference $\delta e = e_r - e_g$ is given to 3rd-order by:

$$\delta e = \boldsymbol{d}^T \boldsymbol{K}^T \boldsymbol{M}_b^T \left[\boldsymbol{M}_g^T \boldsymbol{C}\left(\boldsymbol{x}_g - \boldsymbol{x}_a\right) + \boldsymbol{M}_r^T \boldsymbol{C}(\boldsymbol{x}_r - \boldsymbol{x}_a) \right] \qquad (4)$$

where $\boldsymbol{M}_g^T$ represents the adjoint model run backwards over the forecast interval and linearized about the green forecast solution $\boldsymbol{x}_g$; $\boldsymbol{M}_b^T$ is the adjoint model run backwards over the 4D-Var analysis interval and linearized about 4D-Var background $\boldsymbol{x}_b$; $\boldsymbol{M}_r^T$ denotes the adjoint model linearized about the red forecast solution $\boldsymbol{x}_r$; $\boldsymbol{K}^T$ is the transpose of the gain matrix; and $\boldsymbol{d}$ is the innovation vector. Thus, the 3rd-order impact given by (4) requires two integrations of the adjoint model: one forced by $\boldsymbol{C}\left(\boldsymbol{x}_g - \boldsymbol{x}_a\right)$ and another forced by $\boldsymbol{C}(\boldsymbol{x}_r - \boldsymbol{x}_a)$ and linearized about different forecast solutions. The next step is to compute these forcing functions for the adjoint model

To create adjoint forcing NetCDF files, go now to the subdirectory **WC13/Data** and run the Matlab script **adsen_37N_transport_fcst.m**. This will create two files **wc13_foi_A_2hours.nc** and **wc13_foi_B_2hours.nc** (if using **NHIS**=4, as discussed in the Readme) which correspond to $\boldsymbol{C}(\boldsymbol{x}_r - \boldsymbol{x}_a)$ and by $\boldsymbol{C}(\boldsymbol{x}_g - \boldsymbol{x}_a)$ respectively.

Step 5:
Go back to the subdirectory **WC13/RBL4DVAR_forecast_impact** and compile the forecast observation impact driver using **build_roms.csh** using the following cpp options:

```
  setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DRBL4DVAR_FCT_SENSITIVITY"
  setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DAD_IMPULSE"
  setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DOBS_IMPACT"
 #setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DOBS_SPACE"
  setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DRPCG"
```

Before running this application, you must first run the script **job_rbl4dvar_fct_impact.csh** which will copy all of the input files that you created during the previous steps, and those needed from Exercise 3, to the current working directory.

Before running this application, take a look at **roms_wc13.in** and notice that there are two additional timestep parameters, **NTIMES_ANA** and **NTIMES_FCT** which correspond to the number of time steps in the analysis and forecast cycles respectively as shown in Fig. 1. It is these parameters that control the time-stepping of ROMS in this case, and the usual parameter **NTIMES** is ignored. The values of **NTIMES_ANA** and **NTIMES_FCT** that are currently set in **roms_wc13.in** are the correct ones for this application, so do not change them. Notice also the input files **FOInameA** and **FOInameB** which correspond to the adjoint forcing functions that were created in step 4.

Now run the job using **romsM**.

Step 6:
To plot the output from step 5, go to the subdirectory **WC13/plotting** and run the Matlab script **plot_rbl4dvar_forecast_impact.m**.

Create a new subdirectory, **Case1**, and save the solution in it for analysis and plotting to avoid overwriting solutions when playing with difference CPP options and rerunning and recompiling:

```
mkdir Case1
mv Build_roms rbl4dvar.in *.nc log Case1
cp -p romsM roms_wc13_2hours.in Case1
```

where log is the ROMS standard output specified.

Case 2: Measuring observation impacts using independent observations

In this second example, we will use equation (3) as a measure forecast error. For this exercise, e is chosen to be the mean squared error in SST during forecast day 7 over the target area identified in Fig. 2. In this case, the 3rd-order approximation for δe becomes:

$$\delta e = \boldsymbol{d}^T \boldsymbol{K}^T \boldsymbol{M}_b^T \left[\boldsymbol{G}_g^T \boldsymbol{C}(\boldsymbol{y}_g - \boldsymbol{y}) + \boldsymbol{G}_r^T \boldsymbol{C}(\boldsymbol{y}_r - \boldsymbol{y}) \right] \qquad (5)$$

where $\boldsymbol{G}_g^T$ and $\boldsymbol{G}_r^T$ denote the adjoint model *forced* at the observation points and linearized about green and red forecasts (cf Fig. 1) respectively. Steps 1-3 are identical to Case 1, so there is no need to repeat these. However, we now need to create the forcing functions for the adjoint since they will be different for this case.

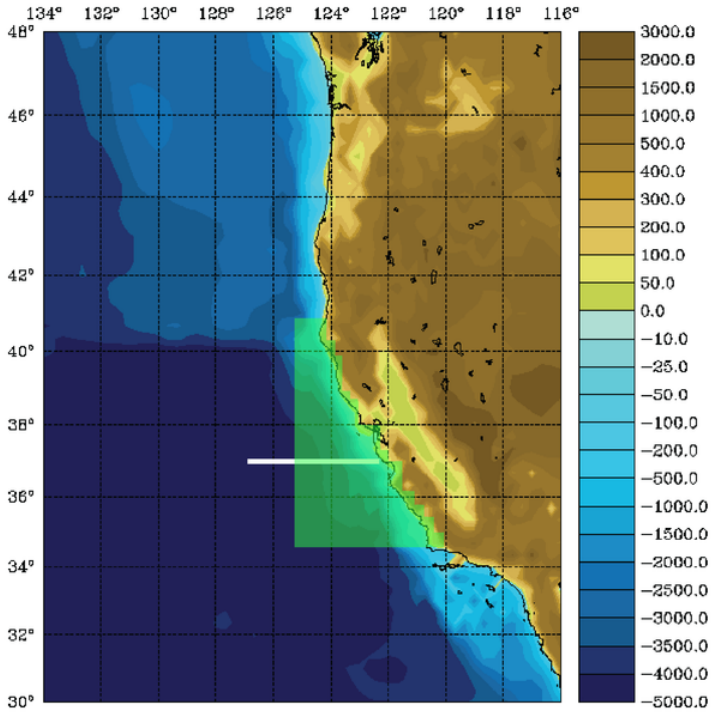

Figure 2: The central California target area used to define the mean squared SST forecast error metric (green box) given by equation (3).

Step 7:
Go now to the subdirectory **WC13/Data** and run the Matlab script **adsen_SST_fcst.m**. This will overwrite the two netcdf files **wc13_oifA.nc** and **wc13_oifB.nc** which correspond to $\boldsymbol{C}(\boldsymbol{y}_r - \boldsymbol{y})$ and by $\boldsymbol{C}(\boldsymbol{y}_g - \boldsymbol{y})$ respectively.

Step 8:
Go back to the subdirectory **WC13/RBL4DVAR_forecast_impact** and compile the forecast observation impact driver using **build_roms.csh** using the following cpp options:

```
setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DRBL4DVAR_FCT_SENSITIVITY"
setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DAD_IMPULSE"
setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DOBS_IMPACT"
setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DOBS_SPACE"
setenv MY_CPP_FLAGS "${MY_CPP_FLAGS} -DRPCG"
```

The cpp flag **OBS_SPACE** informs the model that you are using a forecast error metric that resides in observation space instead of state-space.

Before running this observations impact application, you must first run the script **job_rbl4dvar_fct_impact_obs_space.csh** which will copy all of the input files that you created during the previous steps, and those needed from Exercise 3, to the current working directory.

Before running this application, take a look at the input file *s4dvar.in* and take note of the input files **OIFnameA** and **OIFnameB** which correspond to the adjoint forcing functions created in step 7.

Now run the job using **romsM**.

Create a new subdirectory, **Case2**, and save the solution in it for analysis and plotting to avoid overwriting solutions when playing with difference CPP options and rerunning and recompiling:

```
mkdir Case2
mv Build_roms rbl4dvar.in *.nc log Case2
cp -p romsM roms_wc13_2hours.in Case2
```

where log is the ROMS standard output specified

Step 9:
To plot the output from step 8, go to the subdirectory **WC13/plotting** and run the Matlab script **plot_rbl4dvar_forecast_impact_obs_space.m**.