EnKF Sequential Data Assimilation

July 2, 2019
McCone Rm 265 @ 5:30p

Tutorial Summary

Sequential data assimilation techniques provide a vital framework for forecasting geophysical systems. The Ensemble Kalman Filter (EnKF) is a Monte Carlo based sequential data assimilation approach that provides model updates through time as observations are assimilated. The EnKF has been adopted by the climate modeling community as a state-of-the-art scheme for data assimilation due to its ease of implementation, its ability to handle highly nonlinear systems and data with disparate temporal and spatial scales. Given the widespread adoption of the EnKF approach, UCAR/NCAR Research Application Laboratory provides computing and resources for groups interested in utilizing this technique: https://ral.ucar.edu/solutions/products/ensemble-kalman-filter-system-enkf.

This tutorial walks through the application of EnKF to ground deformation modeling at active volcanoes. Synthetic, time varying InSAR data are assimilated into the 1958 Mogi Model, which is updated at each time step to investigate volcanic system evolution. Python is used for this exercise – Please cite Gregg and Pettijohn (2016) and Zhan and Gregg (2017).

Learning Goals

- Creation of synthetic InSAR ground deformation data
- Calculating ground deformation using Mogi's 1958 analytical Model
- Running the EnKF in Python
- The impact of temporal sparsity on model convergence
- Understand the relationship between ensemble size, number of iterations, and convergence
- Evaluating the relationship between initial parameter guess and convergence

References

The EnKF Technique was adopted for volcano deformation modeling in:

Gregg, P. M., and J. C. Pettijohn (2016), A multi-data stream assimilation framework for the assessment of volcanic unrest, *Journal of Volcanology and Geothermal Research*, *309*, 63-77, 10.1016/j.jvolgeores.2015.11.008.

The EnKF was further expanded for volcano deformation modeling in:

Zhan, Y., and P. M. Gregg (2017), Data assimilation strategies for volcano geodesy, *Journal of Volcanology and Geothermal Research*(special issue on Volcano Geodesy), 10.1016/j.jvolgeores.2017.02.015.

The EnKF Data Assimilation Technique was originally developed in:

Evensen, G. (1994), Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, *J. Geophys. Res.-Oceans*, *99*(C5), 10143-10162, 10.1029/94jc00572.

The EnKF Data Assimilation Textbook:

Evensen, G. (2009), *Data Assimilation: The Ensemble Kalman Filter, Second Edition*, Springer, Heidelberg, 10.1007/978-3-642-03711-5.

Exercise

Step 1: Download EnKF Tutorial Files from the box.
Step 2: Open Jupyter Notebook
In Terminal:
```
$ jupyter notebook
```
Step 3: Go to the EnKF Tutorial Folder and open EnKF_Zhan_Gregg.ipynb

The EnKF Approach:

$$A_a = A + P_e H^T (H P_e H^T + R_e)^{-1} (D - H A)$$

A: Forecast ensemble matrix
A_a: updated ensemble matrix
P_e: ensemble covariance matrix
R_e: measurement error covariance matrix
H: measurement operator relating the true model state

In this tutorial you will use an ensemble of Mogi models to assimilate synthetic InSAR data and track the evolution of an evolving volcanic system.

The Analytical Model:

Kiyoo Mogi derived a relationship between magma chamber pressure and surface displacement in his famous 1958 paper.

$$U_z = \left[\frac{(1+\nu)(1\text{-}2\nu)}{E\nu} \right] \frac{3r^3 P d}{4(d^2 + x^2)^{3/2}}$$

U_z is surface deformation in the vertical direction, v is Poisson's ratio, E is Young's Modulus, r is magma chamber radius, P is magma chamber pressure, d is the depth to the center of the magma chamber, x is the horizontal distance along the surface above the source.

The Data:

In this tutorial we will create a synthetic InSAR data set (*we are not geodesists, so this is just a very simple test – be kind!*) to mimic an inflating volcanic system using a Mogi model with a time varying pressure source. You will create a synthetic data set and then assimilate it back into the model to see if the EnKF can resolve the parameters.

TEST 1: Temporal resolution test

Data sparsity in space or time can greatly impact model convergence. In this test you will investigate the sensitivity of EnKF convergence to the temporal sparsity of InSAR data. InSAR data can have long return periods between observations (however, this has been vastly improved in recent years). To test the impact of return time, you will create a 5-year InSAR data set with a 6 month return interval.

Under ***Create Deformation Data***:
```
# time series
t = np.linspace(0,5,10)
```

1. Run the code as is to view the convergence and how well the EnKF captures the parameter changes.
   Is EnKF able to track the changes in overpressure? What do the results tell you about the Mogi model vs. the EnKF's abilities? Is the model inadequate or EnKF?

2. Now change the number of time steps to see how this impacts model convergence. For example: `t = np.linspace(0,5,20)` to mimic 4 InSAR observations per year and/or `t = np.linspace(0,5,5)` for 1 InSAR reading per year.
   How does the temporal resolution impact the ability of the EnKF to converge?

> Prior to using the EnKF technique for your specific models and data, it is important to run tests to determine the sensitivity of the EnKF to different set ups. The next two tests will provide a sense for the sensitivity of the EnKF to how the scheme is set up.

TEST 2: Initial Ensemble Size and Iteration Number Test

The EnKF is an ensemble technique which requires defining an ensemble size. Additionally, our testing has indicated that when dealing with temporally sparse data sets, iterating on data prior to moving forward to the next time step can improve data convergence.

Under ***Define the number of ensemble and number of iterations***:

```
# number of ensemble
nens = 200
# number of iteration
nitr = 5
```

1. Change the number of ensembles (`nens`) to 50 and/or 500 to test the sensitivity of the EnKF to the number of models in your ensemeble.
   How sensitive is model convergence to the number of ensembles?

2. Change the number of iterations (`nitr`) to 1 to test the sensitivity to the EnKF to iterating before continuing on to the next time step.
   How sensitive is model convergence to the number of iterations?

Which of these two values is most critical for EnKF convergence using this approach?

TEST 3: Initial Parameter Values

The range of values in the initial parameters for the EnKF ensemble can impact the outcome of the model. Next you will modify the starting values to observe how the EnKF responds to different starting parameter spaces.

```
# number of the parameters
npar = 5
# Initial guess of parameter values
# point source location
```

```
# x (km)
xc0 = np.random.standard_normal(nens) * 1 + -2
# y (km)
yc0 = np.random.standard_normal(nens) * 1 + 2
# z (km)
zc0 = np.random.standard_normal(nens) * 1 - 3
# radius (km)
rc0 = np.random.standard_normal(nens) * 0.2 + 0.3
# overpressure (MPa)
dp0 = np.random.standard_normal(nens) * 1 + 0
```

1. First play around with the starting guess for the spatial location of the pressure source. In the initial parameters above (x, y, z) are normal distributions centered around (-2, 2, -3). Change the code to center the initial spatial parameters at (0, 0, -4) and (3, 3, -1) to see how these assumptions impact the ability of EnKF to find the source location.
   How sensitive is the EnKF to the initial locations estimated? How might this impact the use of EnKF when you are unsure about the location of the pressure source.

2. As we saw in the first test, the non-uniqueness of the Mogi model impacts the ability of the EnKF to resolve Pressure and Radius. In this final test, let's pretend that we have tomographic or gravity data indicating that our pressure source has an approximate radius of 500 m. Let's also assume that we know that the source is inflating, so we can constrain our delP to positive numbers.

   First change the distribution of the first radius guess:
   ```
   rc0 = np.random.standard_normal(nens) * 0.2 + 0.3
   ```

   Next change the pressure distribution to be centered around 10 MPa:
   ```
   dp0 = np.random.standard_normal(nens) * 1 + 10
   ```

Did constraining the radius and pressure help the EnKF to find these parameters or better fit the volume change?

Discussion Questions

1. What are some challenges do you notice from running these tests? Are there particular caveats that should be considered when using the EnKF?

2. How might you use data assimilation in your own research? What model outputs might you be able to match to your observations?