APPM 5510 HW 5

Zane Jakobs

October 4, 2019

1

Let $\mathbf{\Pi}_{i;j}$ be the Jacobian of the flow map Ψ at time i with respect to the initial condition at time j. Varying one parameter at a time, starting with $\boldsymbol{x}_0$, we have

$$\begin{aligned}\nabla J_1 \cdot \delta \boldsymbol{x}_0 &= \nabla J_0 \cdot \boldsymbol{x}_0 + \lambda^T \mathbf{\Pi}_{1;0} \cdot \boldsymbol{x}_0 \\ &= 2(\boldsymbol{x}_0 - \boldsymbol{x}_b)^T \boldsymbol{B}_0^{-1} \cdot \delta \boldsymbol{x}_0 + 2(\boldsymbol{x}_1 - \Psi(\boldsymbol{x}_0))^T \boldsymbol{B}_1^{-1} \Pi_{1;0}^T \cdot \delta \boldsymbol{x}_0 + 2(\boldsymbol{y}_0 - \boldsymbol{H}_0 \boldsymbol{x}_0)^T \boldsymbol{R}_0^{-1} \boldsymbol{H}_0 \cdot \delta \boldsymbol{x}_0\end{aligned}$$

and with $\boldsymbol{x}_1$, we get

$$\begin{aligned}\nabla J_1 \cdot \delta \boldsymbol{x}_1 &= \nabla J_0 \cdot \delta \boldsymbol{x}_1 - \lambda^T \cdot \delta \boldsymbol{x}_1 \\ &= 2(\boldsymbol{x}_1 - \Psi(\boldsymbol{x}_0))^T \boldsymbol{B}_1^{-1} \cdot \delta \boldsymbol{x}_1 + 2(\boldsymbol{y}_1 - \boldsymbol{H}_1 \boldsymbol{x}_1)^T \boldsymbol{R}_1^{-1} \boldsymbol{H}_1 \cdot \delta \boldsymbol{x}_1 - \boldsymbol{\lambda}^T \cdot \delta \boldsymbol{x}_1\end{aligned}$$

We now have the equation

$$\begin{aligned}0 &= \nabla J_1 \cdot \boldsymbol{x} \\ &= 2(\boldsymbol{x}_0 - \boldsymbol{x}_b)^T \boldsymbol{B}_0^{-1} \cdot \delta \boldsymbol{x}_0 + 2(\boldsymbol{x}_1 - \Psi(\boldsymbol{x}_0))^T \boldsymbol{B}_1^{-1} (\boldsymbol{I} \cdot \delta \boldsymbol{x}_1 + \Pi_{1;0}^T \cdot \boldsymbol{x}_0) \\ &- (\boldsymbol{y}_0 - \boldsymbol{H}_0 \boldsymbol{x}_0)^T \boldsymbol{R}_0^{-1} \boldsymbol{H}_0 \cdot \delta \boldsymbol{x}_0 - (\boldsymbol{y}_1 - \boldsymbol{H}_1 \boldsymbol{x}_1)^T \boldsymbol{R}_1^{-1} \boldsymbol{H}_1 \cdot \delta \boldsymbol{x}_1 - \boldsymbol{\lambda}^T \cdot \delta \boldsymbol{x}_1.\end{aligned}$$

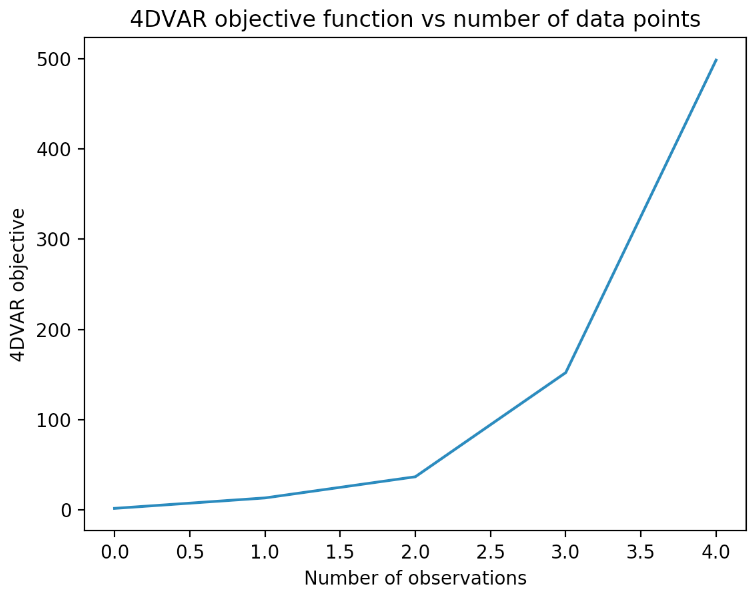
Dividing out differentials and rearranging terms, we see that

$$\begin{aligned}\boldsymbol{\lambda} = 2\boldsymbol{B}_0^{-T}(\boldsymbol{x}_0 - \boldsymbol{x}_b) + 2(\boldsymbol{I} + \Pi_{1;0}^T)^T \boldsymbol{B}_1^{-T}(\boldsymbol{x}_1 - \Psi(\boldsymbol{x}_0)) \\ - \boldsymbol{H}_0^T \boldsymbol{R}_0^{-T}(\boldsymbol{y}_0 - \boldsymbol{H}_0 \boldsymbol{x}_0) - \boldsymbol{H}_1^T \boldsymbol{R}_1^{-T}(\boldsymbol{y}_1 - \boldsymbol{H}_1 \boldsymbol{x}_1)\end{aligned}$$

The difference between this and the gradient in the notes is that here, we have a term depending on the actual value of $\boldsymbol{x}_1$ (instead of the flow map from $\boldsymbol{x}_0$ to time 1), and we only need to compute one $\boldsymbol{\lambda}$, instead of two (for two times) because of the constraint that $\boldsymbol{x}_1 = \Psi(\boldsymbol{x}_0)$.

2

(Note: code attached to the back of the assignment)



3

Letting $\boldsymbol{x} = (x, y)^T$, we have $x_{j+1} = a_j x_j + y_j + \eta_0$, $y_{j+1} = x_j + 2y_j$, and $a_{j+1} = a_j$. Since a is independent of x, differentiating each equation w.r.t. each element of (x, y, a) gives the following Jacobian matrix:

$$\begin{bmatrix} a & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Code

```
import matplotlib.pyplot as plt
import numpy as np

# x_t -> x_{t+1}
def xmap(x):
    return 3.95 * x * (1 - x)
```

```
def psi(x0,t):
    i = 0
    x = x0
    while i < t:
        x = xmap(x)
        i += 1
    return x

def H(x):
    return x + np.random.normal(0, 0.01)


def obs_error(y, x, t):
    xt = psi(x,t)
    return y - H(xt)

def varobjective4d(x0, y):
    xb = 1./3.
    b0 = 0.1 ** 2

    R = 0.01 ** 2

    J = (1/b0) * (x0 - xb) ** 2

    for i in range(len(y)):
        e = obs_error(y[i], x0, i)

        J += (1./R) * (e ** 2)

    return J

if __name__ == '__main__':

    true_obs = [0.25]
    for i in range(4):
        true_obs.append(xmap(true_obs[-1]))

    obs = [H(x) for x in true_obs]

    Jvals = []
```

```
for i in range(1,len(obs)):
    Jvals.append(varobjective4d(obs[0], obs[:i]))


plt.plot(Jvals)
plt.title("4DVAR objective function vs number of data points")
plt.xlabel("Number of observations")
plt.ylabel("4DVAR objective")
plt.show()
```