3-D Data Assimilation of Ocean Gliders in the Ekman Layer

Adam Mallen

Marquette University

February 12, 2013

Motivation

- Data assimilation on glider location observations to help plan flight paths
- Glider location data is used in two ways:
  - Data is assimilated into a physical model of glider movement through the fluid.
  - Data is assimilated into a model of the background fluid flow.

What Can Go Wrong

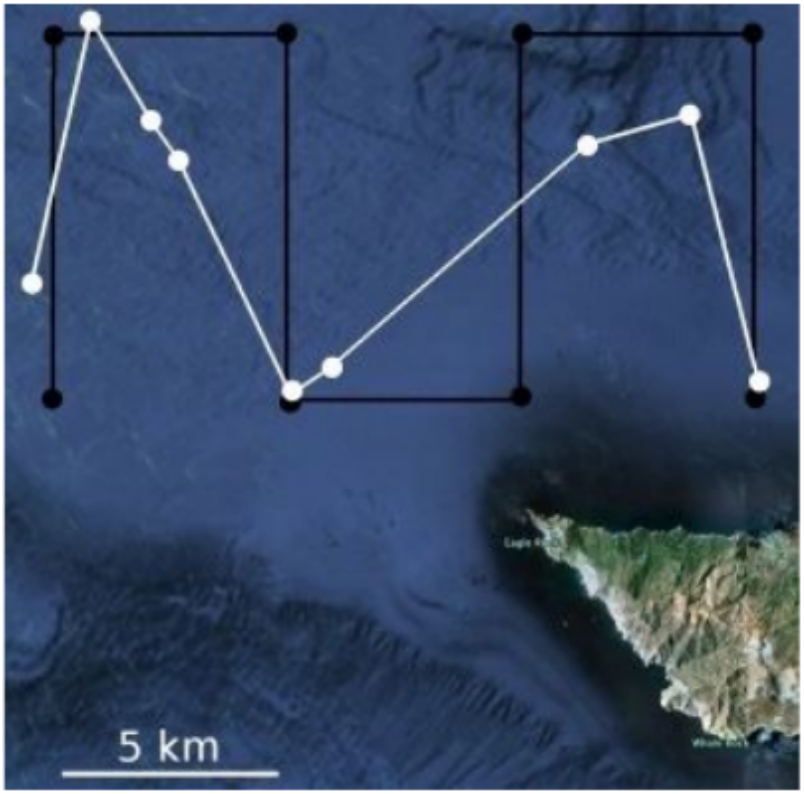


Figure : Example of a mismatch between planned flight path and true glider trajectory. (taken from Smith 2011).

Motivation

- Often data and control for Lagrangian instruments focuses only on horizontal glider movement.
- Often researchers assume perfect control over glider navigation (direction and velocity of flight).
- Our goal: explore assimilating glider location observations into a physical model of glider control with unknown parameters moving in a background flow with unknown parameters.
- How can we improve parameter estimates to help plan better flight paths?

Outline

- Glider model
  - Equilibrium equations for buoyancy, lift, and drag
- Simulation and control framework
- Background flow
  - Eckman layer model
- Data assimilation
- Future work

Glider Model

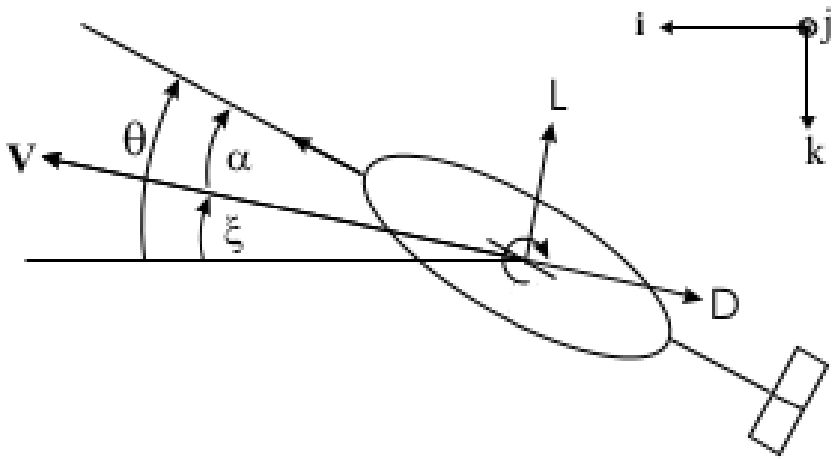


Figure : Model of angle variables and lift and drag operating on the glider. θ is the pitch angle. ξ is the glide angle. α is the angle of attack. (taken from Leonard 2001).

Glider Model

- We assume the glider can control pitch angle and ballast mass.
  - Pitch angle–with lift and drag–determine glide angle at equilibrium.
  - Ballast mass determines buoyancy force which plays a part in determining velocity at equilibrium.
- Equilibrium equations of buoyancy, lift, and drag relate glider control to velocity and flight direction (glide angle).
- The goal is to design a flight path by
  - choosing a desired glide angle and desired velocity
  - and computing the control that yields the desired glide angle and velocity at equilibrium.

Simulation Framework (for known background flow)

1. Given initial estimates of unknown parameters in glider model.
   - Here, these are just lift and drag coefficients.
2. Determine flight path.
   - Here, this is just the desired glide angle and velocity.
3. Compute glider control to yield the desired flight path.
4. Simulate true glider movement given control from step (3).
   - This involves computing the *true* glide angle and velocity.
   - Also involves the background flow's effect on glider movement.
   - Observe the glider's location (possibly with noise).
   - This is our data for assimilation.

Computing Glider Control (3)

- Pitch angle at equilibrium is a function of the desired glide angle and the angle of attack at equilibrium.

$$\theta = \xi_d + \alpha_d$$

- Angle of attack at equilibrium is a function of the desired glide angle and the lift and drag at equilibrium.
- Parameters involved in lift and drag approximations are unknown, though an initial estimate is known (possibly an estimate from a previous iteration of data assimilation).

Computing Glider Control (3)

- We approximate the lift and drag operating on the glider as

$$L = (K_{L_0} + K_L \alpha) V^2 \tag{1}$$

$$D = (K_{D_0} + K_D \alpha^2) V^2 \tag{2}$$

- When we balance the forces at equilibrium in the x and z directions we get

$$0 = [\cos(\xi), \sin(\xi)][D, L]^T \tag{3}$$

$$m_0 g = [-\sin(\xi), \cos(\xi)][D, L]^T \tag{4}$$

- m_0 is the excess mass and $m_0 g$ is the buoyancy force.

Computing Glider Control (3)

- Equation (3) yields the following quadratic in α

$$0 = \alpha^2 + \frac{K_L}{K_D}\tan(\xi)\alpha + \frac{1}{K_D}(K_{D_0} + K_{L_0}\tan(\xi)) \qquad (5)$$

- Compute α_d using equation (5), the desired glide angle, ξ_d, and the initial estimates of the lift and drag coefficients.
- Compute the pitch angle, θ,

$$\theta = \xi_d + \alpha_d \qquad (6)$$

Computing Glider Control (3)

- Let m be the mass of the displaced fluid, m_b be the controllable ballast mass, and m_h be the remaining mass of the glider hull. Then,

$$m_0 = m_b + m_h - m \tag{7}$$

- and equation (4) becomes

$$m_b = (m - m_h) + \frac{1}{g}(-\sin(\xi)(K_{D_0} + K_D\alpha^2) + \cos(\xi)(K_{L_0} + K_L\alpha)V^2 \tag{8}$$

- Compute m_b from equation (8) using ξ_d, V_d, α_d, and the estimates of the lift and drag coefficients.

Simulate True Glider Movement (4)

- To simulate the true movement of the glider we need to compute the true flight path, ξ_t and V_t, given the control $[\theta, m_b]$ and the true lift and drag coefficients.
- To find ξ_t we can substitute $\xi_t = \theta - \alpha_t$ into equation (5) and solve for α_t

$$0 = {\alpha_t}^2 + \frac{K_L}{K_D}\tan(\theta - \alpha_t)\alpha_t + \frac{1}{K_D}(K_{D_0} + K_{L_0}\tan(\theta - \alpha_t)) \quad (9)$$

- To find V_t we can solve equation (8) for V_t with ξ_t and α_t now known.

Simulate True Glider Movement (4)

- Let $U(x, y, z, t) = [u_x, u_y, u_z]^T$ be the background flow velocity field.
- We simulate the true glider movement by evolving the glider through time according to

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = U(x, y, z, t) + V_t \tag{10}$$

Simulate True Glider Movement (4)

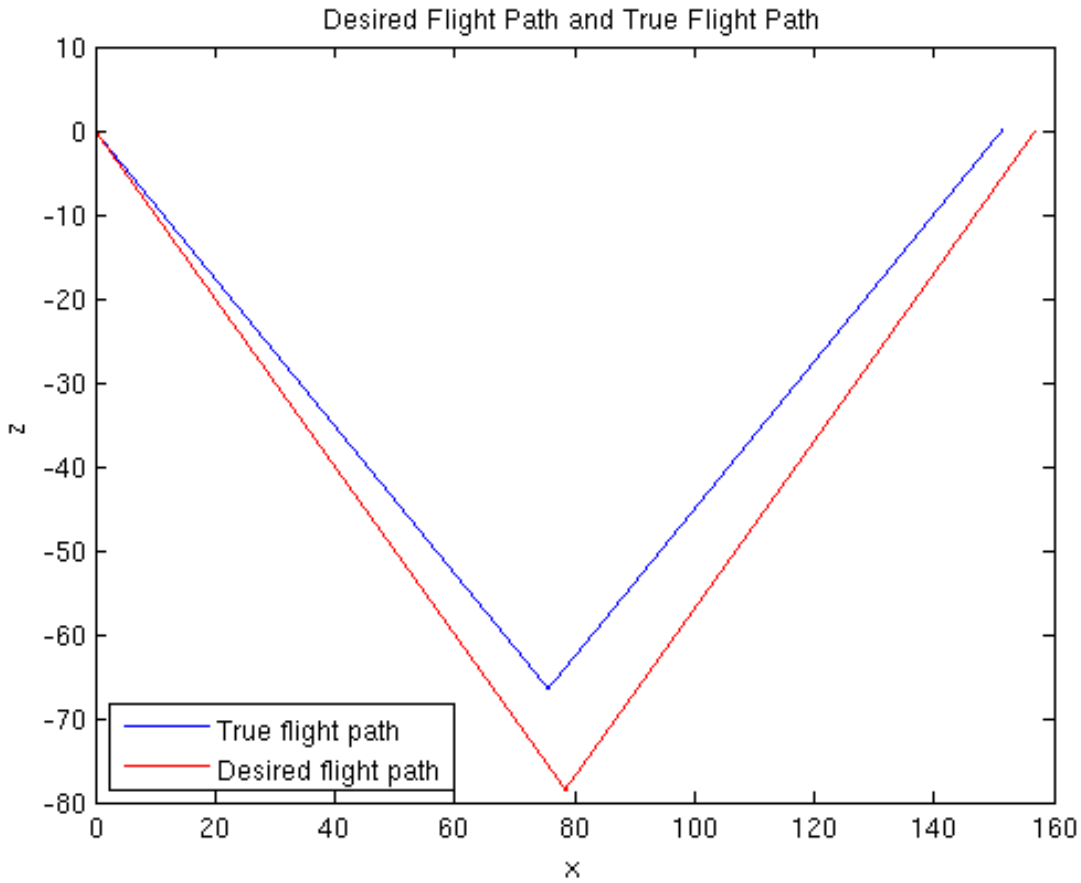


Figure : Errors in the lift and drag parameter estimates can cause a discrepancy between the desired glide angle and the true glide angle. The same goes for the desired velocity and true velocity.

Ekman Layer Model

- The Ekman layer is a model of the fluid velocity in the top layer of the ocean caused by wind stress on the surface and the Coriolis force.
- It is parameterized by the Ekman depth, d, which is the scale of the depth at which the wind stress affects fluid flow (depth of the boundary layer). Here it is considered unknown.
- The Ekman depth, d, is a function of the viscosity and rotation of the system.
- The Ekman layer velocity field is z-dependent but has no vertical velocity component.

Ekman Layer Model

- Let u,v be the fluid flow velocity in the x,y direction.
- Let $\bar{u}$,$\bar{v}$ be the z-independent velocity of an underlying flow.
- Let τ/ρ be the wind stress per unit mass at the surface.
- Let f be the Coriolis parameter.
- The Ekman layer velocity field is

$$u = \bar{u}(x,y) + \frac{\sqrt{2}}{fd} e^{z/d} \left[\frac{\tau^x(x,y)}{\rho} \cos\left(\frac{z}{d} - \frac{\pi}{4}\right) - \frac{\tau^y(x,y)}{\rho} \sin\left(\frac{z}{d} - \frac{\pi}{4}\right) \right]. \tag{11}$$

$$v = \bar{v}(x,y) + \frac{\sqrt{2}}{fd} e^{z/d} \left[\frac{\tau^x(x,y)}{\rho} \sin\left(\frac{z}{d} - \frac{\pi}{4}\right) + \frac{\tau^y(x,y)}{\rho} \cos\left(\frac{z}{d} - \frac{\pi}{4}\right) \right]. \tag{12}$$

Ekman Spiral

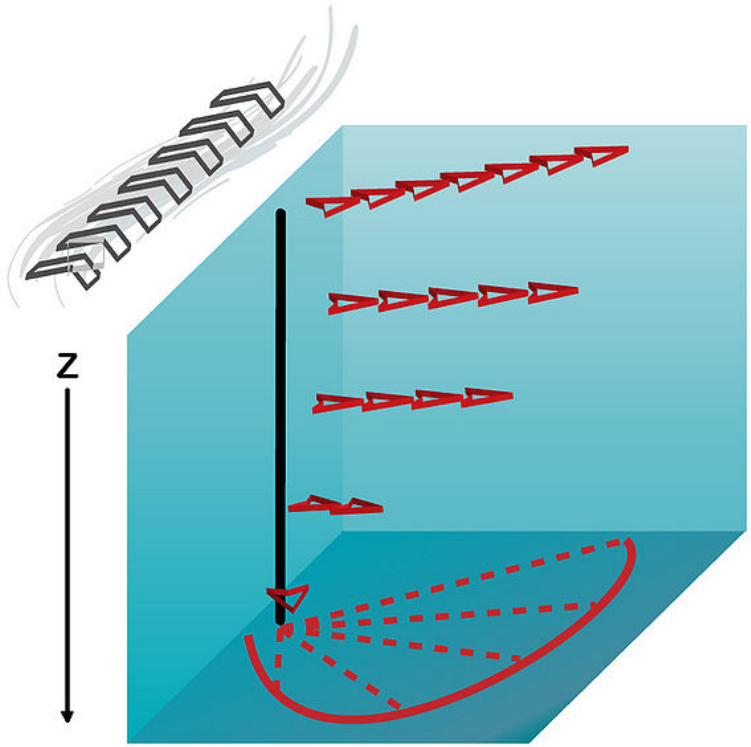


Figure : Example velocity field for a column of water in the Eckman layer. (Taken from "Ekman Layer" on Wikipedia)

Simulation Framework with Ekman Layer

1. Given initial estimates of unknown parameters in glider model and background flow model.
   - Here, these are just lift and drag coefficients for the glider model
   - and the Ekman depth for the Ekman Layer model.
2. Determine flight path.
   - Here, this is just the desired glide angle and velocity.
   - These will depend on the estimate of the background flow.
3. Compute glider control to yield the desired flight path.
4. Simulate true glider movement given control from step (3).
   - Involves using the true background flow.

Data Assimilation

- The (x, y, z) location of the glider's path during the dive is our observation data.
- The goal is to improve our estimates of the lift and drag coefficients as well as our estimate of the Ekman depth.
- Can use many assimilation techniques.
- We use a simple implementation of a particle filter.
  - Take a large number of random estimates of the unknown parameters ('particles').
  - Compute each particle's (x, y, z) trajectory.
  - Weight each particle's parameter estimates according to the likelihood of its trajectory yielding the observed data.
  - These estimates, along with their associated weights, represent an approximation to the posterior distribution of parameter estimates.

Particle Filter

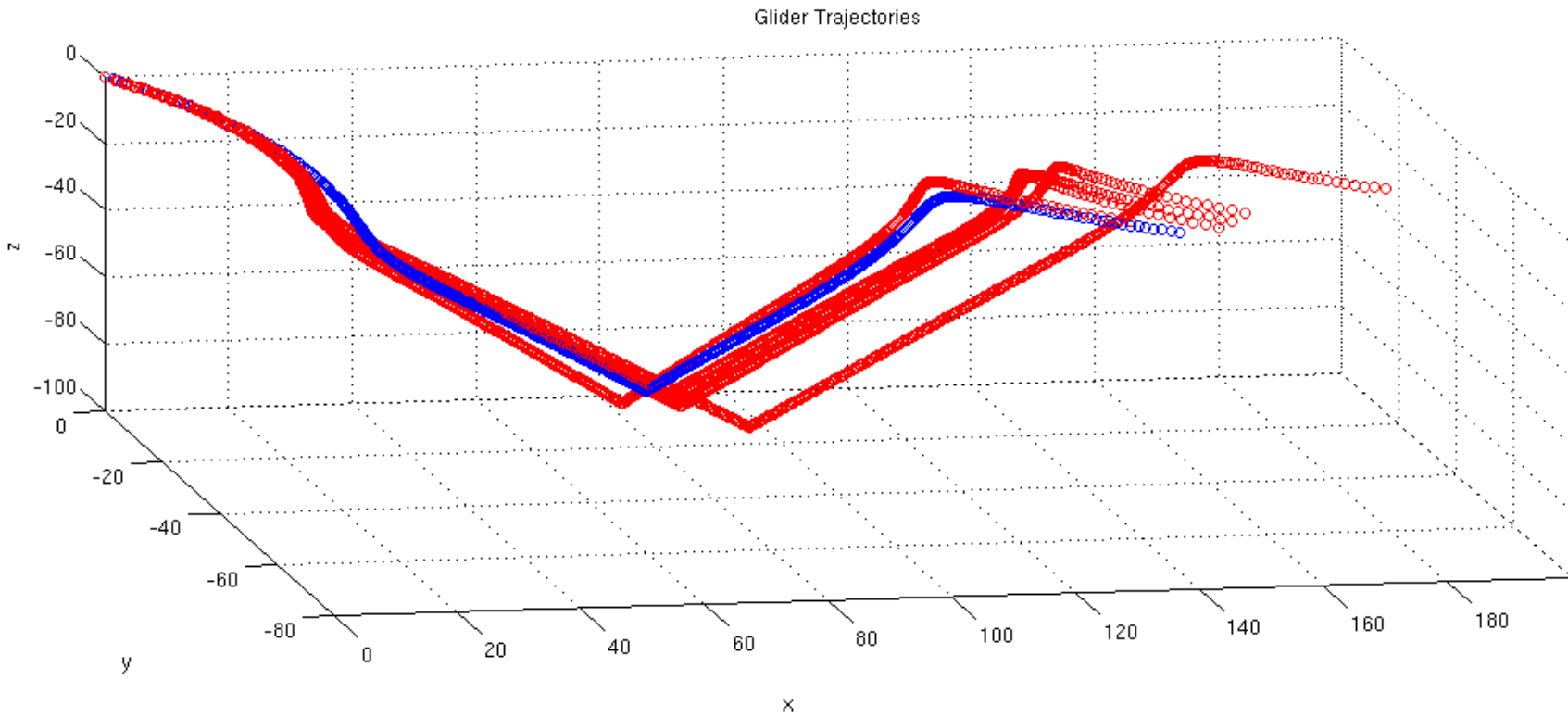


Figure : True glider trajectory given in blue. 5 particles' trajectories are given in red.

Results

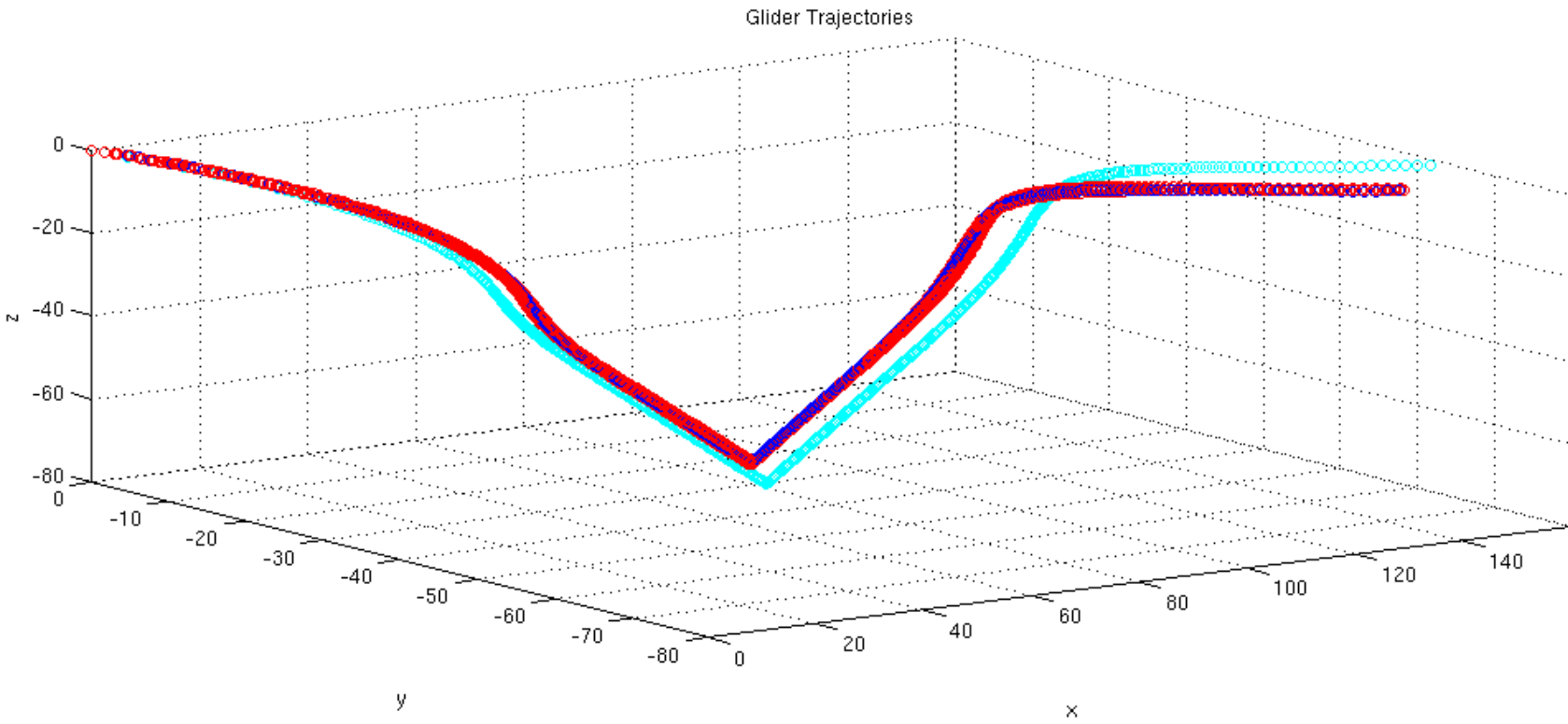
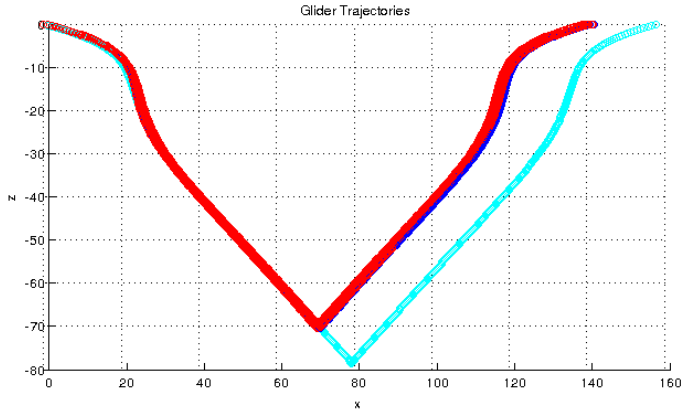
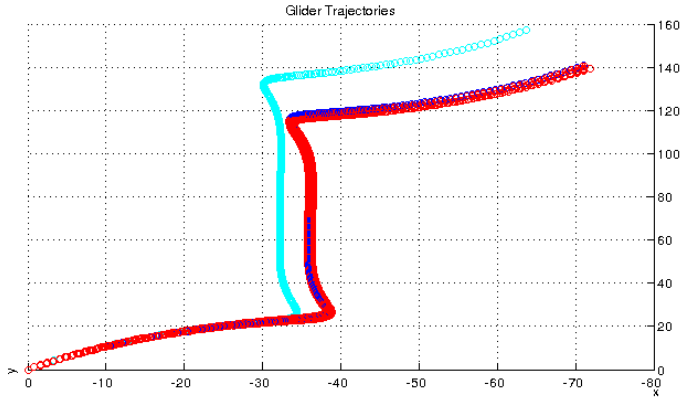
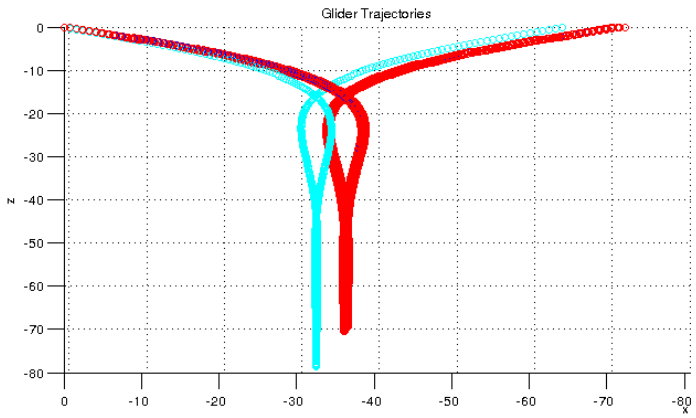


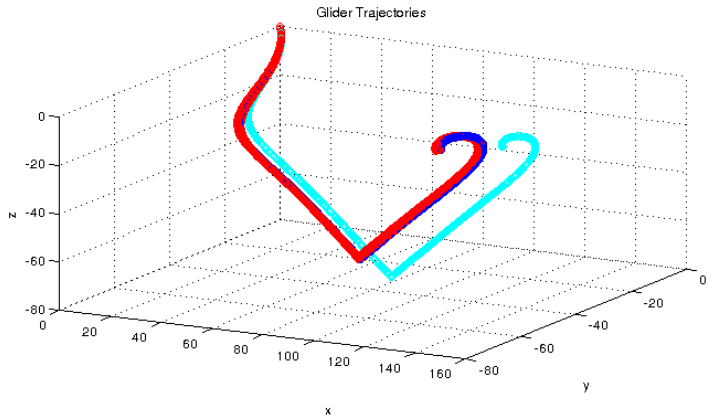
Figure : Desired flight path given in cyan. True glider trajectory given in blue. 5 best particles' trajectories are given in red.

Results


Glider Trajectories
z
x


Glider Trajectories
y
x


Glider Trajectories
z
x


Glider Trajectories
z
x
y

Results

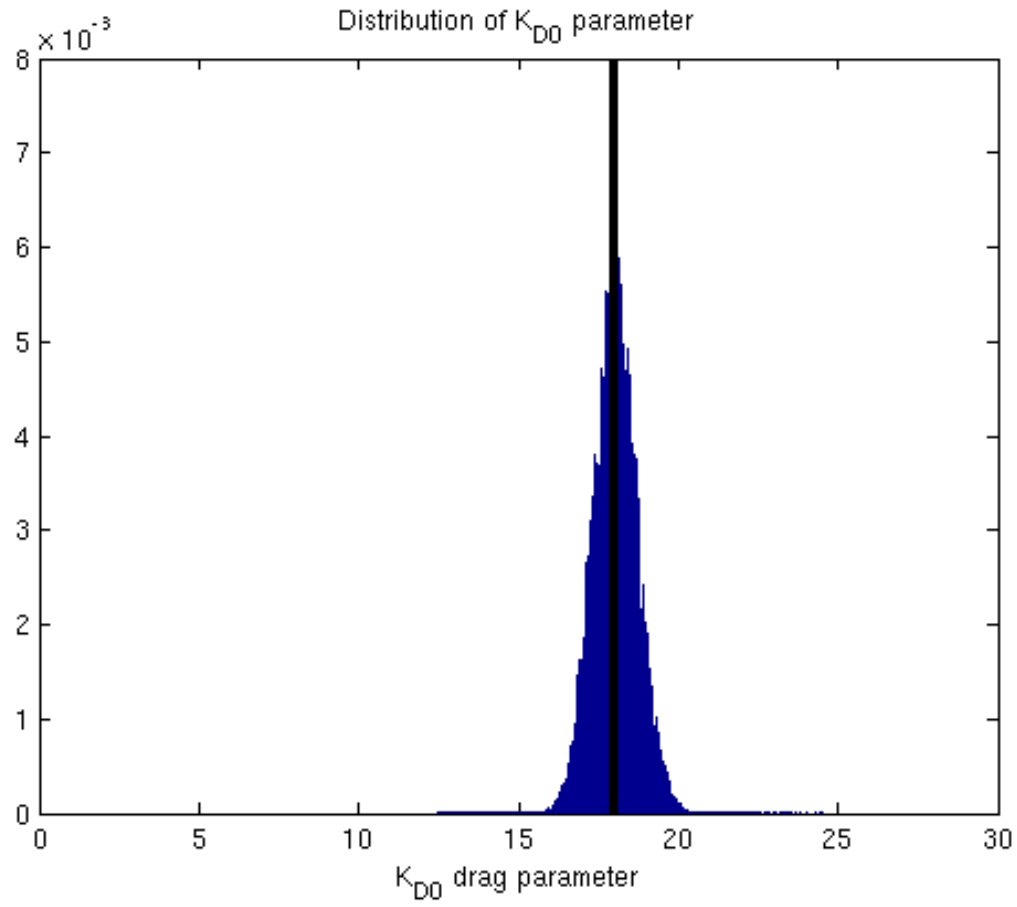


Figure : Distribution of K_{D_0} parameter for 5000 particles. Distribution mean given in black. True value of parameter given in red.

Results

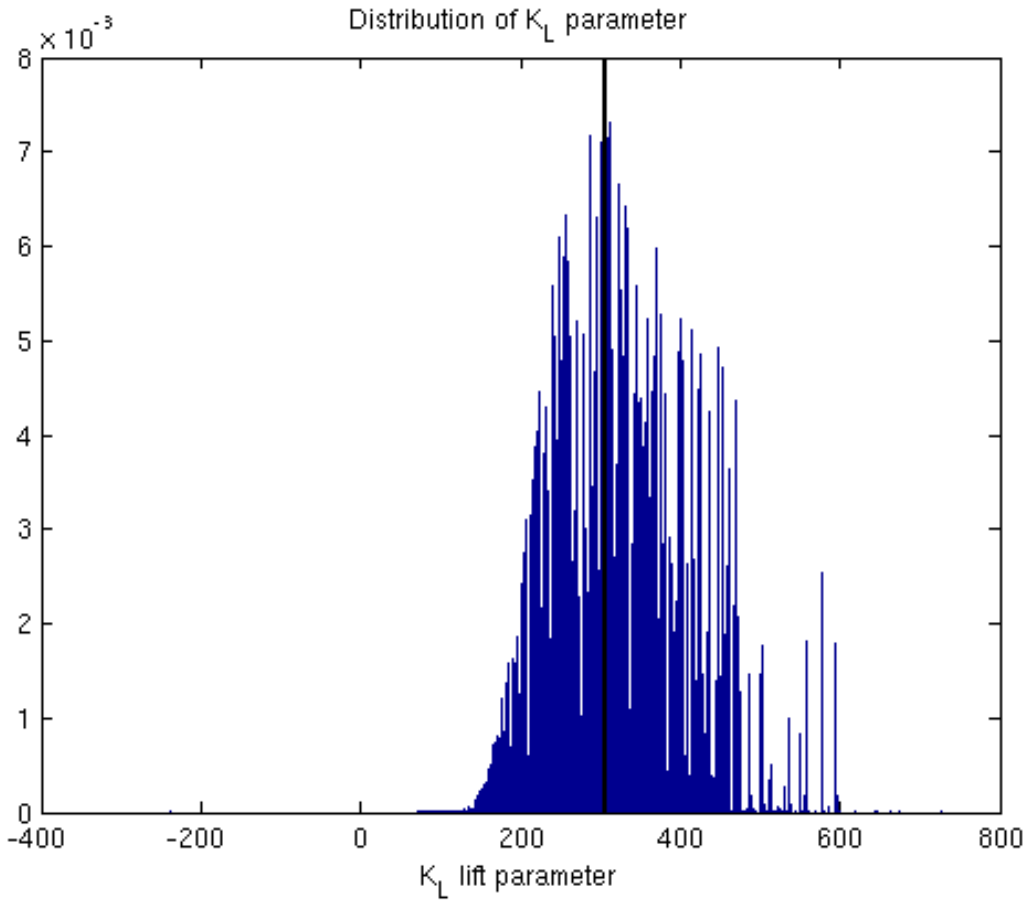


Figure : Distribution of K_L parameter for 5000 particles. Distribution mean given in black. True value of parameter given in red.

Results

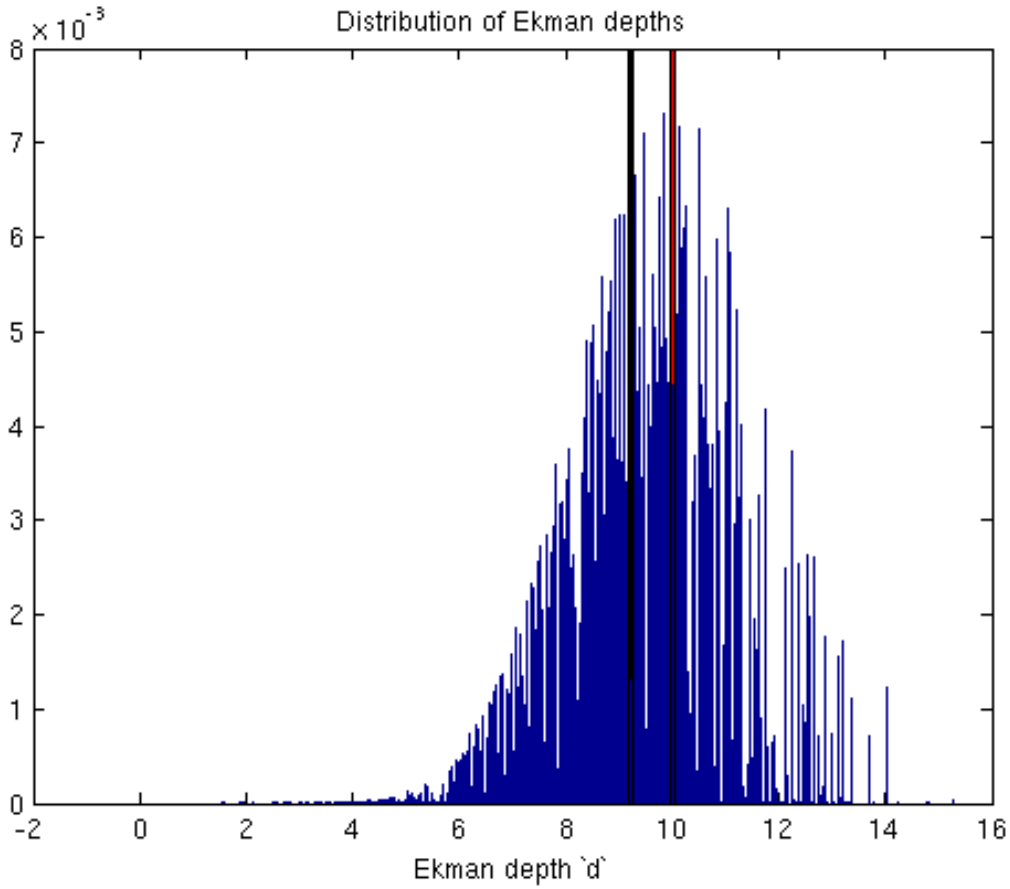


Figure : Distribution of Ekman depth parameter, d for 5000 particles. Distribution mean given in black. True value of parameter given in red.

Future Work

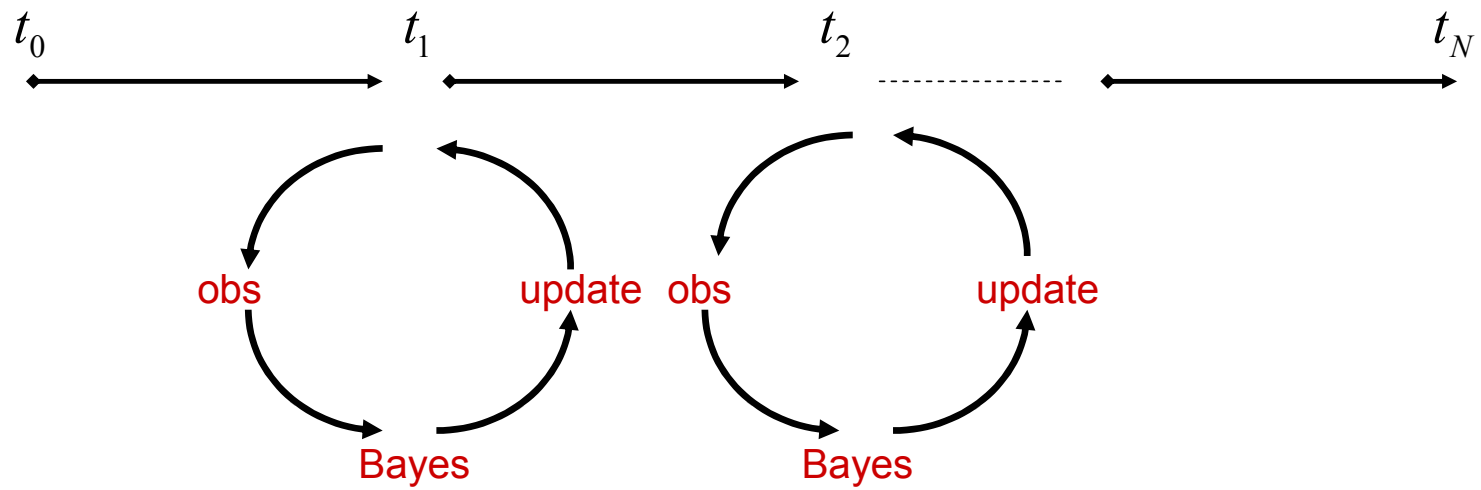
- Currently using whole glider trajectory for observations. Using just the glider's surfacing position is more realistic.
- Our current Ekman layer model is relatively simple and is only a function of the depth of the glider. We are looking for a more sophisticated model of the background flow.
  - This could just be a 2-D horizontal flow model that sits underneath the Ekman spiral caused by wind stress.
  - Could be spatially or temporally varying flow.
  - Could contain spatially or temporally varying wind speed.
- Using data assimilation to make better flight plans and keep true trajectories closer to desired flight plans.

Questions?

Thanks!
Any Questions... ?

Bibliography

- Smith, Ryan N. et al. "Persistent Ocean Monitoring with Underwater Gliders: Adapting Sampling Resolution." Journal of Field Robotics 28.5 (2011): 714 - 741.
- Naomi Ehrich Leonard and Joshua G. Graver. "Model-Based Feedback Control of Autonomous Underwater Gliders." IEEE Journal of Oceanic Engineering, vol. 26, No. 4, October 2001: 633 - 645.
- "Ekman Layer." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 26 October 2012. Web. 9 February 2013.



$x =$ state

$Y =$ obs

$$p(x \mid Y) \propto p(Y \mid x)\, p(x)$$

Key question: how do we obtain the distributions on RHS?

We're interested in the probability of a state X_j as it evolves over time. Recall, for independent random variables, we have

$$p(x_{0:n}) = p(x_0) \prod_{j=1}^{n} p(x_j | x_{1:j-1})$$

For our case, we'll have some distribution of initial conditions $\mu(x_0)$ (background) and a model to move our state forward in time,

$$X_j | (X_{j-1} = x_{j-1}) \sim m(x_j | x_{j-1})$$

where $m(x_j | x_{j-1})$ is the *transition probability* or the probability that our model would take use from state x_{j-1} to state x_j.

Combining the ideas above gives us

$$p(x_{0:n}) = \mu(x_0) \prod_{j=1}^{n} m(x_j | x_{j-1})$$

Recall, our observations will be related to the state variable by some observation function $y = H(x)$. We can think of observations as random variables distributed as

$$Y_j|(X_j = x_j) \sim g(y|x_j).$$

Or, $Y_j = H(X_j) + \text{“}noise\text{”}$.

$g(y|x)$ is the *likelihood* — how likely was an observation given the possible states?

With a whole set of observations $\{Y_j\}$ we can write down the likelihood for the time-series of observations

$$p(y_{1:j}|x_{1:j}) = \prod_{j=1}^{n} g(y_k|x_k)$$

Given a background distribution of initial conditions, $\mu(x_0)$, and observations, $Y_{1:n}$, we want to infer the distribution of physical states $X_{0:n}$.

- Prior

$$p(x_{0:n}) = \mu(x_o) \prod_{j=1}^{n} m(x_j|x_{j-1})$$

- Likelihood

$$p(y_{1:n}|x_{1:n}) = \prod_{j=1}^{n} g(y = H(x_j)|x_j)$$

- Posterior, obtained by Bayes' rule

$$p(x_{1:n}|y_{1:n}) = \frac{p(y_{1:n}|x_{1:n})p(x_{0:n})}{p(y_{1:n})}$$

recall, $p(y_{1:n}) = \int p(y_{1:n}|x_{1:n})p(x_{0:n})dx_{1:n}$

A Monte Carlo simulation or really sampling $p(x_{1:n}|y_{1:n})$

- takes a discrete set of samples from $X_0 \sim p(x_0)$
- moves them forward accord to the model, e.g. samples $X_{0:j} \sim p(x_j|x_{0:j-1})$
- evaluates likelihood between samples and observations

Note, after a few (say $k = 2$ or 3 observations) you will have samples from $X_{0:k} \sim p(x_{0:k}|y_{1:k})$ but they will not be useful.

Idea — normalize at every step, treat that posterior distribution as an *importance* prior distribution for the next step.

1. Start with $X_o \sim p(x_o)$, each particle $X_o^{(k)}$ has weight $w_1^{(k)} = 1/N$

Idea — normalize at every step, treat that posterior distribution as an *importance* prior distribution for the next step.

1. Start with $X_o \sim p(x_o)$, each particle $X_o^{(k)}$ has weight $w_1^{(k)} = 1/N$
2. Transition each $X_0^{(k)}$ forward, this gives sample $X_1^{(k)} \sim p(x_1|x_0) = m(x_1|x_0)$

Idea — normalize at every step, treat that posterior distribution as an *importance* prior distribution for the next step.

1. Start with $X_o \sim p(x_o)$, each particle $X_o^{(k)}$ has weight $w_1^{(k)} = 1/N$
2. Transition each $X_0^{(k)}$ forward, this gives sample $X_1^{(k)} \sim p(x_1|x_0) = m(x_1|x_0)$
3. Evaluate the likelihood function of each sample ("particle") $X_1^{(k)}$ against Y_1, $g(Y_1|X_1^{(k)})$

Idea — normalize at every step, treat that posterior distribution as an *importance* prior distribution for the next step.

1. Start with $X_o \sim p(x_o)$, each particle $X_o^{(k)}$ has weight $w_1^{(k)} = 1/N$
2. Transition each $X_0^{(k)}$ forward, this gives sample $X_1^{(k)} \sim p(x_1|x_0) = m(x_1|x_0)$
3. Evaluate the likelihood function of each sample ("particle") $X_1^{(k)}$ against Y_1, $g(Y_1|X_1^{(k)})$
4. *Weight* each particle by

$$w_1^{(k)} = \frac{g(Y_1|X_1^{(k)})w_0^{(k)}}{\sum_{k=1}^{N} g(Y_1|X_1^{(k)})w_0^{(k)}}$$

Idea — normalize at every step, treat that posterior distribution as an *importance* prior distribution for the next step.

1. Start with $X_o \sim p(x_o)$, each particle $X_o^{(k)}$ has weight $w_1^{(k)} = 1/N$
2. Transition each $X_0^{(k)}$ forward, this gives sample $X_1^{(k)} \sim p(x_1|x_0) = m(x_1|x_0)$
3. Evaluate the likelihood function of each sample ("particle") $X_1^{(k)}$ against Y_1, $g(Y_1|X_1^{(k)})$
4. *Weight* each particle by

$$w_1^{(k)} = \frac{g(Y_1|X_1^{(k)})w_0^{(k)}}{\sum_{k=1}^{N} g(Y_1|X_1^{(k)})w_0^{(k)}}$$

Repeat process transition from X_{j-1} to X_j instead of 0 to 1.
$\pi(x_j|Y_{1:j}) = \{x_j = X_j^{(k)}, w^{(k)}\}$

With a large number of samples, SIS works pretty well on moderate (small) dimensional deterministic (perfect model) problems.

Problem:

- A significant problem, though, is that most (or all) of the weight can be taken over by *one particle*

Solution:

- Resampling, e.g., bootstrapping

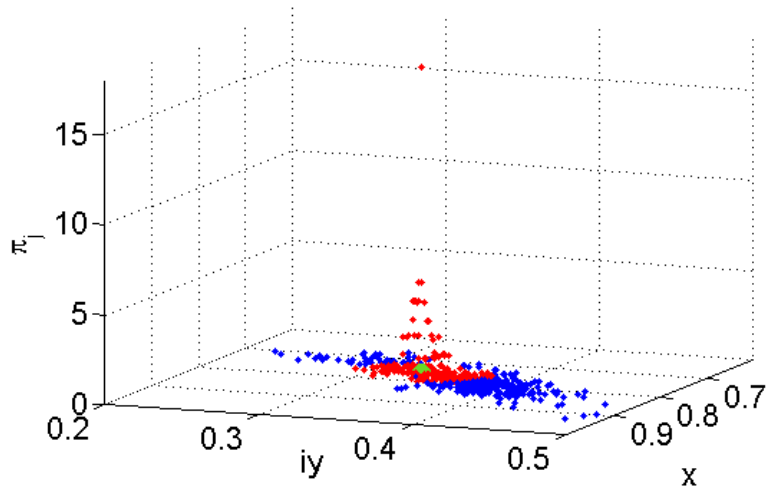
Strategy:

- Monitor weights, if problematic
- *Resample* or "bootstrap" by treating $\tilde{\pi}_j(x_{0:k}|Y_{1:k})$ as an *importance* empirical distribution
- Set all weights to $w_j^{(k)} = 1/N$
- Transition $j+1$ step, repeating resampling as necessary

The strategy is referred to an *SIR (sequential importance resampling) filter* and also goes by the names *particle filter*, *bootstrap filter*, and *sequential Monte Carlo*.

idea:

- pick subset of "best" particles $k = 1, \ldots, M$
- make m_k copies of each particle where $m_k \propto w_j(x_j^{(k)})$ where $\sum m_k = N$



reasonable:

- stochastic evolution to t_{j+1} "spreads out" cloud
- add "jitter" to each particle for deterministic evolution