

SECTION 1

Simulation Set Up

In this section, I briefly describe the simulation set up from the most trivial case all the way up to the true problem. We start with a brief introduction about how the inversion process proceed.

SUBSECTION 1.1

Inversion Process

From hydrostatic balance, the surface pressure is proportional to the surface sea height (SSH).

$$\eta = \frac{p\{z=0\}}{g} \quad (1.1)$$

In all the QG models, the pressure is expanded asymptotically in the form of

$$p_{total} \sim p^0 + \epsilon p^1 \quad (1.2)$$

The first order pressure is directly related to the first order potential Φ^0 by the formula

$$p^0 = f\Phi^0$$

As from geostrophic balance we have

$$p_y^0 = -fu^0 = -f\partial_y\Phi^0 \quad p_x^0 = fv^0 = f\partial_x\Phi^0$$

Now we have

$$\eta^0 = \frac{f\Phi^0}{g}$$

Similarly from the first order correction we have

$$\eta^1 = \frac{p^1}{g}$$

So in general, considering the first order correction to the hydrostatic balance Eq 1.1, we have

$$\eta \sim \eta^0 + \epsilon\eta^1 = \frac{f\Phi^0}{g} + \epsilon\frac{p^1}{g} \quad (1.3)$$

This is the equation 40 in Ryan's note.

Now remember our goal is to use η to invert for Φ^0 . What is the relationship between p^1 and Φ^0 ? The relationship is given by eq ???. We copy it here

$$\boxed{\nabla^2 p^1 - f\zeta^1 = 2J(\Phi_x^0, \Phi_y^0)}$$

Where J is the Jacobian operator. This is a non-linear relationship, recall that ζ^1 is related to the potentials via Eq ???. So

$$\nabla^2 p^1 = f \left(\nabla^2 \Phi^1 + F_{zy}^1 - G_{zx}^1 \right) + 2J(\Phi_x^0, \Phi_y^0) \rightarrow \Phi^{0,s} + \epsilon \mathcal{N}(\Phi^{0,s}) = \eta(x, y)$$

Where \mathcal{N} is a non-linear operator. Here the relationship between Φ^1, F^1 and G^1 are given

by Eq ??, ?? and ???. Then the whole inversion problem is clear. Given $\eta(x, y)$ we wish to find a $\Phi^{0,s}$ ¹

¹given $\Phi^{0,s}$ we can reconstruct the 3D Φ already.

SUBSECTION 1.2

Pre-Defined Velocity Fields

We start with a very simple case to play around with this model.

1.2.1 Toy Model Setup

The toy model start with a pre-defined potential Φ^0 . I use a random generator here and make the spectrum follow a power law with slope -3, which is typical for 3D turbulence.

```

1 rng(42); % Set seed for reproducibility
2 % We produce a random $\Phi^0$ here.
3 k_peak = 4;
4 slope = -3; % This slope matches the energy cascade in 3D turbulence
5 phase = rand(N, N) * 2 * pi;
6 amplitude = (K ./ k_peak).^(slope) .* exp(-(K./k_peak).^2);
7
8 amplitude(1,1) = 0;
9 % Compute the hat
10 phi0_hat = amplitude .* exp(1i * phase);
11 % Inverse transform to get to the physical space
12 phi0_surf = real(ifft2(phi0_hat));
13 % Normalize
14 phi0_surf = phi0_surf / std(phi0_surf(:));

```

1.2.2 Forward Process to get the true velocity fields

Then some functions are defined for different purposes.

1. This function compute the 3D potential Φ^0 from the surface data. The fourier components has an exponential decay in the vertical direction.

```

1 phi0_3d_true = derive_phi0_3d(phi0_surf, K, z, Bu);

```

2. This function compute all the other first order potential F^1, G^1 and Φ^1 using Eq ??, ?? and ???. A possion problem is solved in the spectral space.

```

1 [F1_true, G1_true, Phi1_true] =
    calculate_higher_order(phi0_3d_true, K, kx, ky, z, Bu, N,
    nz);

```

3. This function computes the first order pressure p^1 using Eq ??.

```

1 p1_true = solve_p1(f, dx, dz, kx, ky, z, Bu, Ro,
    phi0_3d_true, F1_true, G1_true, Phi1_true);

```

4. With all the data above, we can compute the true surface sea height using Eq 1.3.

```

1 p1_surf = p1_true(:, :, end);
2 ssh_true = phi0_surf + Ro * p1_surf;

```

Here I use end because the z coordinate starts from the bottom to the top.

Now we have the surface sea height. This is where the inversion starts.

1.2.3 Inversion Process solving the Optimization Problem

The inversion process starts with η . We make an initial guess for Φ^0 , in my code I use a zero initial Φ^0 .

```
1 phi0_guess_flat = zeros(N, N);
```

Then use the function

```
1 cost_func = @(phi0_flat) sqg_cost_function(phi0_flat, f, ssh_true, K,
  kx, ky, z, Bu, Ro, N, nz, dx, dz);
```

To compute the cost. In this `sqg_cost_function` basically do the same thing as the forward process, compute the 3D potential Φ^0 first, then solve the Possion equation to get F^1, G^1, Φ^1 , then compute p^1 and finally compute the SSH. The difference is that at the final step, we compute the difference between the computed SSH and the true SSH to obtain the cost.

```
1 % Cost
2 difference = ssh_guess - ssh_obs;
3 cost = sum(difference(:).^2);
```

This is the returned value of that function. I then use a built in Matlab toolbox to solve the optimization problem.

```
1 num_iteration = 20;
2 options = optimoptions('fminunc', 'Display', 'iter', 'Algorithm',
  'quasi-newton', 'MaxIterations', num_iteration);
3
4 % Compute the cost function
5 cost_func = @(phi0_flat) sqg_cost_function(phi0_flat, f, ssh_true, K,
  kx, ky, z, Bu, Ro, N, nz, dx, dz);
6
7 % Run Optimization
8 tic;
9 try
10   [phi0_opt_flat, fval] = fminunc(cost_func, phi0_guess_flat,
  options);
11   phi0_surf_opt = reshape(phi0_opt_flat, N, N);
12   disp('Optimization Complete.');
13 catch ME
14   disp('Optimization failed or interrupted.');
15   disp(ME.message);
16   phi0_surf_opt = reshape(phi0_guess_flat, N, N); % Fallback
17 end
18 toc;
```

Then plot the results.²

²The plot part is written by Gemini