Jacobian Matrix Part 1

At 1:59 Andrei says, "you have to check that neither x nor y is zero." What is meant is: "you have to check that x and y are not both zero".

We're going to calculate, step by step, all the partial derivatives in H_j:

$$H_j = \begin{bmatrix} \frac{\partial \rho}{\partial p_x} & \frac{\partial \rho}{\partial p_y} & \frac{\partial \rho}{\partial v_x} & \frac{\partial \rho}{\partial v_y} \\ \frac{\partial \varphi}{\partial p_x} & \frac{\partial \varphi}{\partial p_y} & \frac{\partial \varphi}{\partial v_x} & \frac{\partial \varphi}{\partial v_y} \\ \frac{\partial \dot{\rho}}{\partial p_x} & \frac{\partial \dot{\rho}}{\partial p_y} & \frac{\partial \dot{\rho}}{\partial v_x} & \frac{\partial \dot{\rho}}{\partial v_y} \end{bmatrix}$$

So all of H_j's elements are calculated as follows:

$$\frac{\partial \rho}{\partial p_x} = \frac{\partial}{\partial p_x}(\sqrt{p_x^2 + p_y^2}) = \frac{2p_x}{2\sqrt{p_x^2 + p_y^2}} = \frac{p_x}{\sqrt{p_x^2 + p_y^2}}$$

$$\frac{\partial \rho}{\partial v_x} = \frac{\partial}{\partial v_x}(\sqrt{p_x^2 + p_y^2}) = 0$$

$$\frac{\partial \rho}{\partial v_y} = \frac{\partial}{\partial v_y}(\sqrt{p_x^2 + p_y^2}) = 0$$

$$\frac{\partial \varphi}{\partial p_x} = \frac{\partial}{\partial p_x} arctan(p_y/p_x) = \frac{1}{(\frac{p_y}{p_x})^2 + 1}(-\frac{p_y}{p_x^2}) = -\frac{p_y}{p_x^2 + p_y^2}$$

$$\frac{\partial \varphi}{\partial p_y} = \frac{\partial}{\partial p_y} arctan(p_y/p_x) = \frac{1}{(\frac{p_y}{p_x})^2 + 1}(\frac{1}{p_x}) = \frac{p_x^2}{p_x^2 + p_y^2}\frac{1}{p_x} = \frac{p_x}{p_x^2 + p_y^2}$$

$$\frac{\partial \varphi}{\partial v_x} = \frac{\partial}{\partial v_x} arctan(p_y/p_x) = 0$$

$$\frac{\partial \varphi}{\partial v_y} = \frac{\partial}{\partial v_y} arctan(p_y/p_x) = 0$$

$$\frac{\partial \dot{\rho}}{\partial p_x} = \frac{\partial}{\partial p_x}\left(\frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}}\right)$$

In order to calculate the derivative of this function we use the quotient rule.

Given a function z that is quotient of two other functions, f and g:

$$z = \frac{f}{g}$$

its derivative with respect to x is defined as:

$$\frac{\partial z}{\partial x} = \frac{\frac{\partial f}{\partial x} g - \frac{\partial g}{\partial x} f}{g^2}$$

in our case:

$$f = p_x v_x + p_y v_y$$

Their derivatives are:

$$\frac{\partial f}{\partial p_x} = \frac{\partial}{\partial p_x}(p_x v_x + p_y v_y) = v_x$$

$$\frac{\partial g}{\partial p_x} = \frac{\partial}{\partial p_x}\left(\sqrt{p_x^2 + p_y^2}\right) = \frac{p_x}{\sqrt{p_x^2 + p_y^2}}$$

Putting everything together into the derivative quotient rule we have:

$$\frac{\partial \dot{\rho}}{\partial p_x} = \frac{v_x\sqrt{p_x^2 + p_y^2} - \frac{p_x}{\sqrt{p_x^2+p_y^2}}(p_x v_x + p_y v_y)}{p_x^2 + p_y^2} = \frac{p_y(v_x p_y - v_y p_x)}{(p_x^2 + p_y^2)^{3/2}}$$

Similarly,

$$\frac{\partial \dot{\rho}}{\partial p_y} = \frac{\partial}{\partial p_y}\left(\frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}}\right) = \frac{p_x(v_y p_x - v_x p_y)}{(p_x^2 + p_y^2)^{3/2}}$$

$$\frac{\partial \dot{\rho}}{\partial v_x} = \frac{\partial}{\partial v_x}\left(\frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}}\right) = \frac{p_x}{\sqrt{p_x^2 + p_y^2}}$$

$$\frac{\partial \dot{\rho}}{\partial v_y} = \frac{\partial}{\partial v_y}\left(\frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}}\right) = \frac{p_y}{\sqrt{p_x^2 + p_y^2}}$$

So now, after calculating all the partial derivatives, our resulted Jacobian, H_j is:

$$H_j = \begin{bmatrix} \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} & 0 & 0 \\ -\frac{p_y}{p_x^2+p_y^2} & \frac{p_x}{p_x^2+p_y^2} & 0 & 0 \\ \frac{p_y(v_x p_y - v_y p_x)}{(p_x^2+p_y^2)^{3/2}} & \frac{p_x(v_y p_x - v_x p_y)}{(p_x^2+p_y^2)^{3/2}} & \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} \end{bmatrix}$$

Jacobian Matrix Programming Quiz

FIll in the missing code in the `CalculateJacobian()` function to return the correct Jacobian matrix.

- [ ] Make sure you don't divide by zero, if you do print out an error message and just return Initialized `Hj`
- [ ] Calculate the Jacobian matrix `Hj` using the above equation

```cpp
#include <iostream>
#include <vector>
#include "Dense"

using Eigen::MatrixXd;
using Eigen::VectorXd;
using std::cout;
using std::endl;

MatrixXd CalculateJacobian(const VectorXd& x_state);

int main() {
  /**
   * Compute the Jacobian Matrix
   */

  // predicted state example
  // px = 1, py = 2, vx = 0.2, vy = 0.4
  VectorXd x_predicted(4);
  x_predicted << 1, 2, 0.2, 0.4;

  MatrixXd Hj = CalculateJacobian(x_predicted);

  cout << "Hj:" << endl << Hj << endl;

  return 0;
}
```

Search or ask questions in Knowledge.

Ask peers or mentors for help in Student Hub.

NEXT