Differentiation, Divergence, and Curl

Numerical Differentiation

When we use numerical differentiation for approximating the derivative at a certain value, we need to consider both mathematical and floating point errors. The h value must be between what each approach considers a good h value. The general formula when choosing your h value is: $h = x^{n+1}\sqrt{\epsilon}$. ϵ is the machine epsilon for our purposes it is the rounding error on double precision floating point numbers.

To reduce error further, we can sometimes use the centered difference approach which combines the forward and backward difference with this formula:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

This breaks down faster with increasing numbers of decimal places due to rounding error and me must also consider how complicated the function is when using this method

We can also use the approach where we create an approximation with the Taylor series for higher order differentiation.

With a Richardson extrapolation where we use two series to increase the order of our approximation, we run into the same problem as the mathematical and floating point error with accuracy vs computation efficiency. For Richardson extrapolation in Python, we should not increase the order to more than four.

We can get a very accurate answer using a tabulated approach where we calculate the result of a derivative for different values of h and then iterate the Richardson extrapolation until we have a sufficient number of correct decimal places.

Automatic differentiation is a technique for evaluating derivatives in Python that is very accurate and relies on using non real numbers which in this case are dual numbers ($\epsilon^2 = 0$). We get a really accurate answer because when we use the Taylor series to make an expression for the derivative of a function $f(x+\epsilon)$, the error component includes ϵ^2, ϵ^3, etc, so it becomes zero.

Divergence of a Vector Field

The divergence of a vector field will tell you if a field is acting inwards, outwards, or net zero effect from a point you choose. If the result is positive, it is acting outwards, if it is negative, inwards, and if it is zero then there is no net field. The divergence here can be calculated by $\nabla * \vec{F}\ (x, y) = \frac{\partial \vec{F}(x,y)}{\partial x} + \frac{\partial \vec{F}(x,y)}{\partial y}$. It can also be calculated by summing the centred difference for each partial derivative. Here is a sample question. Determine the divergence of a function at the point (-2,9): $\vec{F}\ (x, y) = 4x^2\ \hat{\imath} + 3y\ \hat{\jmath}$.

This example is quite simple so we can just evaluate the partial derivatives:

$$\nabla * \vec{F}\ (x, y) = \frac{\partial}{\partial x}(4x^2 + 3y) + \frac{\partial}{\partial y}(4x^2 + 3y)$$

$$\nabla * \vec{F}\ (x, y) = 8\text{x} + 3$$

The sum of the partial derivatives gives us this equation which we can substitute the values for x and y at our given point

$$\nabla * \vec{F}\ (-2,9) = 8(\text{-}2) + 3$$

Note that here the final answer will only depend on x

$$\nabla * \vec{F}\,(-2,9) = \text{-}13.$$

Curl of a Vector Field

The curl of a vector field will yield a vector orthogonal to the other two components involved, $\nabla \times \vec{F}\,(x,y)$ would give us a vector in the $\hat{z}$ direction. This can be calculated with:

$$\nabla \times \vec{F}\,(x,y) = \left(\frac{\partial F_y(x,y)}{\partial x} - \frac{\partial F_x(x,y)}{\partial y}\right)\hat{z}\,.$$

For a computer, we would use the centred difference of each partial derivative to calculate the curl:

$$\nabla \times \vec{F}\,(x,y) = \left(\frac{F_y(x+h,y) - F_y(x-h,y)}{2h} - \frac{F_x(x,y+h) - F_x(x,y-h)}{2h}\right)\hat{z}$$

For a counter clockwise field, the curl will go out of the page (positive), and vice versa for clockwise (negative). These can be easily determined with the right-hand rule.