CAD Laboratory (CE4P001) — Assignment 1

Session: Autumn 2025

by Deekshansh Bhagat(22ce02012)

https://github.com/Deekshansh-sapper/CAD-Laboratory-CE4P001

School of Infrastructure Indian Institute of Technology, Bhubaneswar Odisha October 2025

Contents

1. Introduction

This report explains the assignment steps, the Julia code used for every question, and the meaning of the outputs (plots and numerical results). For each question I give:

- the underlying mathematical problem,
- a short, line-by-line explanation of the code blocks, and
- an interpretation of the generated outputs (what the plots show and how they verify the mathematics).

All plotting uses `Plots.jl`. Automatic symbolic/automatic differentiation operations use `CalculusWithJulia.jl` where noted.

2. Question 1 — Hill height scalar field

Given:

$$[\ h(x, y) = 200 - x^2 - 2y^2.\ ]$$

(a) 3D surface and 2D contour plots

Goal: Visualize the scalar field as a surface and contours.

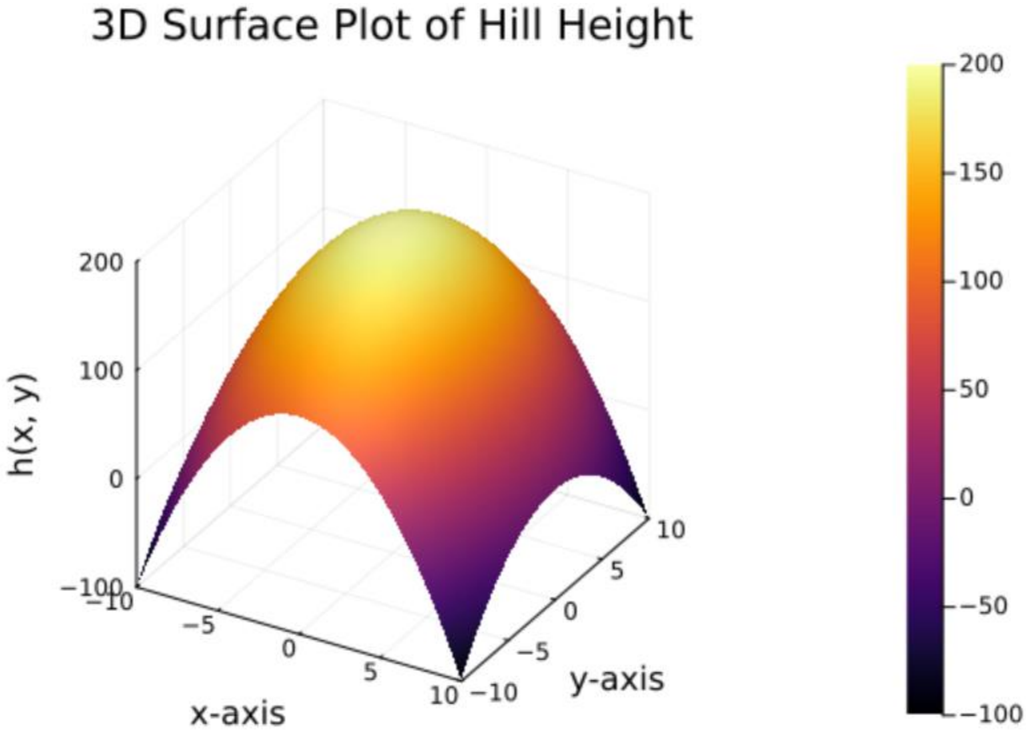
What the code does (stepwise):

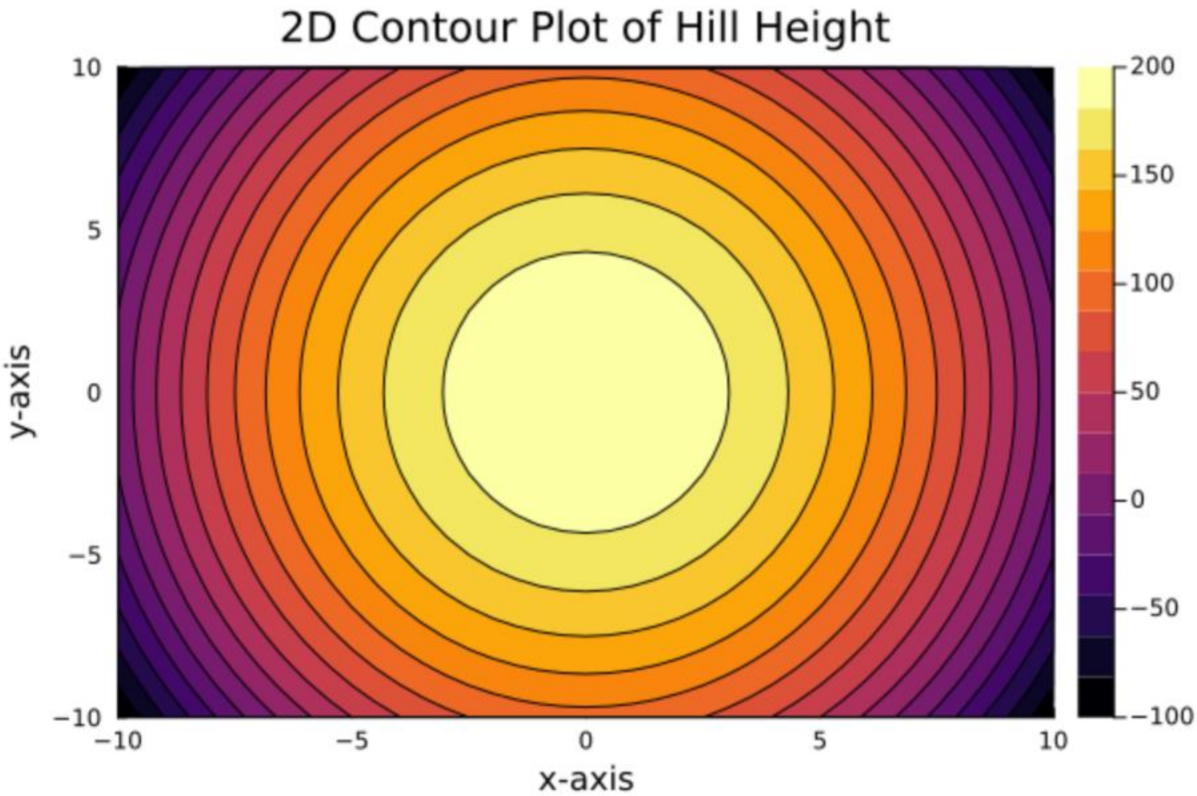
- Define the function `h(x,y)` that returns `200 - x^2 - 2y^2`.
- Create ranges for x and y (`xrange`, `yrange`), here typically from -10 to 10 with a step.
- Build a matrix `z = [h(x1,y1) for x1 in xrange, y1 in yrange]` to store values on the grid.
- Call `surface(xrange, yrange, z, ...)` to produce the 3D surface plot.
- Call `contour(xrange, yrange, z, fill=true, cbar=true, ...)` to produce the 2D contour map.

  Interpretation of outputs:
- The surface plot is an inverted elliptical paraboloid (a hill with a peak). The maximum height occurs at (0,0) with `h(0,0)=200`.
- Contour lines are concentric ellipses centered at the origin because the level sets satisfy `x^2 + 2y^2 = constant`.

- The z-axis range and camera angle affect the visual emphasis; the code sets camera = (30,30) for a clear perspective.

Mathematical check: level curves defined by x^2 + 2y^2 = C → ellipses, consistent with plots.





(b) Automatic gradient (quiver using CalculusWithJulia.jl)

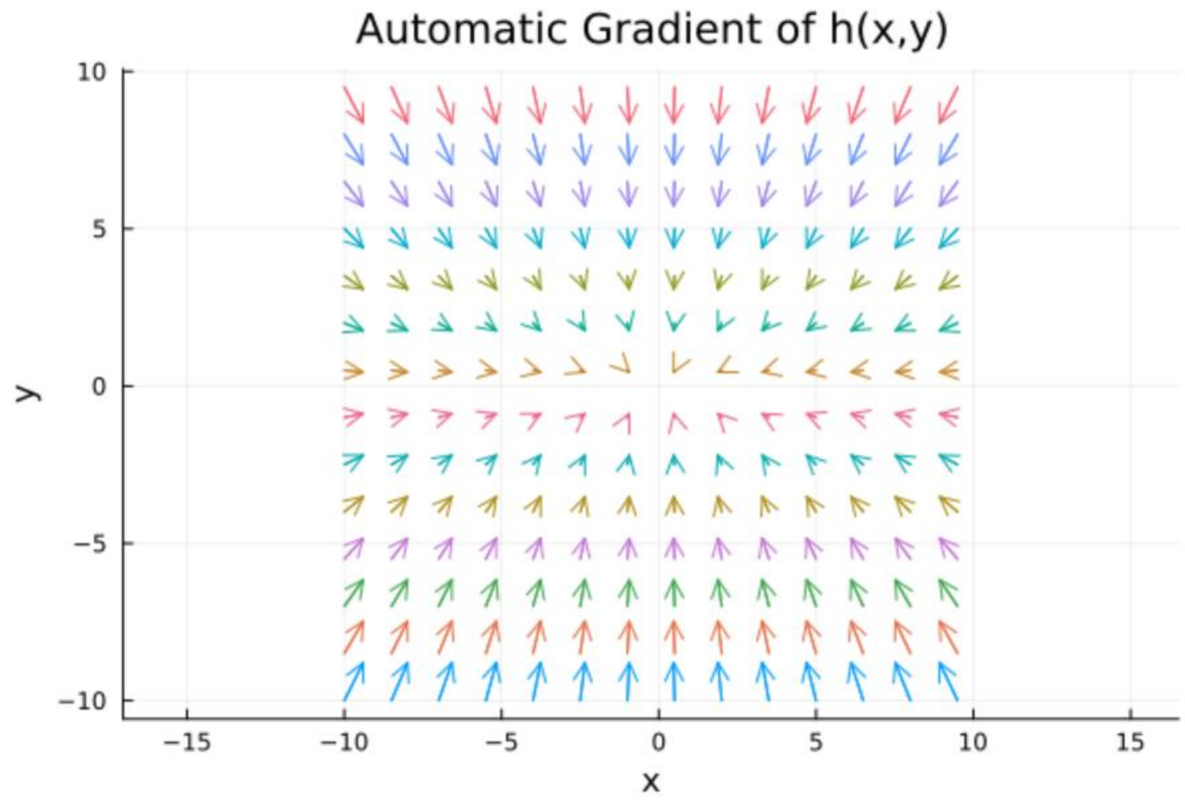
Goal: Plot the gradient vector field using an automatic gradient routine.

What the code does:

- Re-defines h accepting a vector $h(vec) = 200 - vec[1]^2 - 2 * (vec[2]^2)$ so the automatic gradient routine can call it.
- Uses `grad_h(x,y) = gradient(h,[x,y])` from `CalculusWithJulia.jl` to compute the 2-component gradient at each point.
- Builds grid arrays X and Y for plotting quiver arrows.
- Extracts components u and v from the automatic gradient and scales them for better visualization.
- Uses `quiver(X, Y, quiver=(u_scaled, v_scaled), ...)` to show arrows representing ∇h.

Interpretation of outputs:

- The gradient `∇h = (∂h/∂x, ∂h/∂y)` points in direction of maximum increase of h. For this hill:
    - `∂h/∂x = -2x`
    - `∂h/∂y = -4y`
- Arrows point toward the origin when x or y are positive (since derivatives are negative away from origin), meaning the steepest ascent from a point moves towards the peak at the origin.
- Arrow lengths reflect slope magnitude; farther from origin arrows are longer (steeper slope).



(c) Manual gradient and plotting

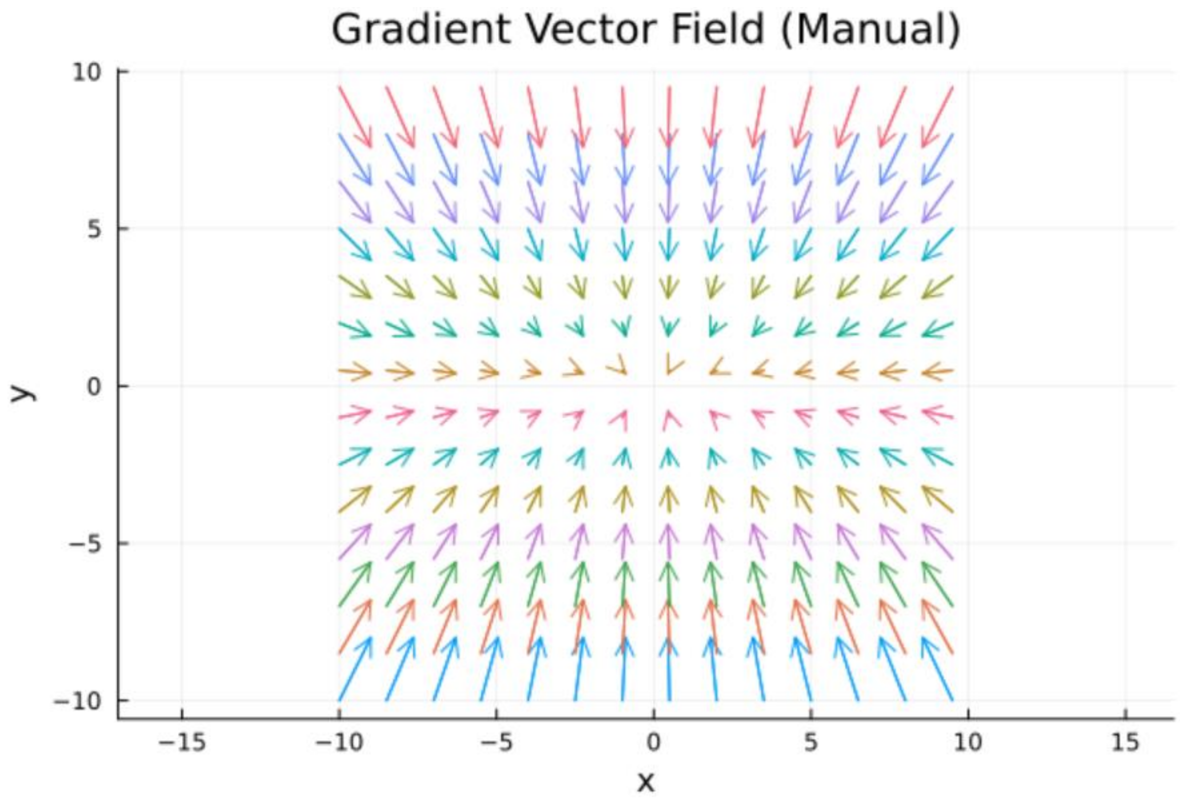
Goal: Compute the gradient analytically and plot it to compare with automatic gradient.

What the code does:

- Declares `A(x,y) = -2*x` and `B(x,y) = -4*y` which are the components of the gradient.
- Builds grid X and Y and component arrays U and V from A and B.
- Scales vectors (`U_scaled`, `V_scaled`) and uses `quiver` to plot.

Interpretation and verification:

- The manual quiver field must match the automatic quiver field from (b). They should be identical up to scaling used for plotting.
- This confirms the correctness of both the symbolic derivative of h and the autograd routine.



3. Question 2 — Cyclone vector field

Given:

$$[\, v(x, y) = x\, e_1 - y^2\, e_2. \,]$$

This represents a 2D vector field: (v(x,y) = (x, -y^2)).

(a) Plot the vector field

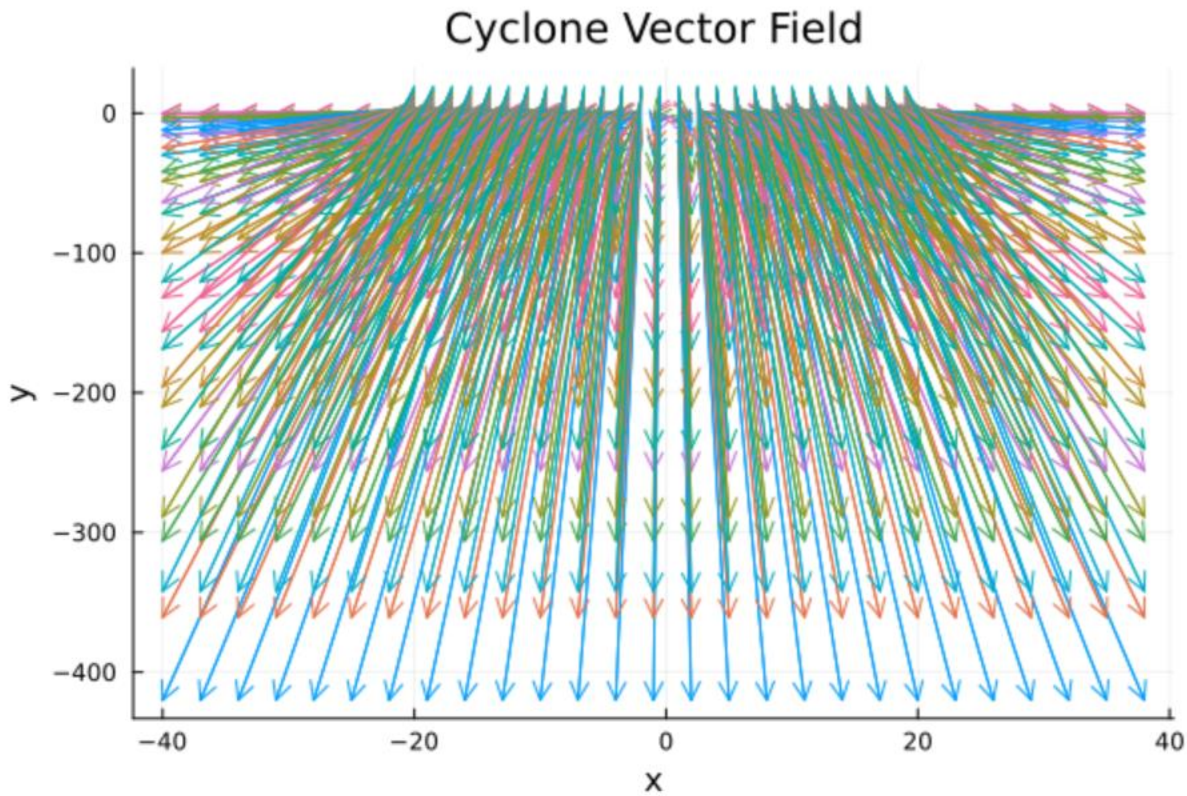
What the code does:

- Defines `vx(x,y) = x` and `vy(x,y) = -y^2`.
- Builds grid ranges (in example -20:1.5:20) and calls `quiver` to plot the vectors at each grid point.

Interpretation:

- The x-component grows linearly with x; arrows point to the right for x>0 and to the left for x<0.
- The y-component is always non-positive (`-y^2 ≤ 0`) — pointing downward or zero on the y-axis.

- Combined, this produces a flow where horizontal motion depends on x, and vertical motion always points downward with magnitude growing with |y|.



(b) Divergence — automatic and manual

Mathematical divergence:

$$\left[\nabla \cdot v = \frac{\partial}{\partial x(x)} + \frac{\partial}{\partial y(-y^2)} = 1 - 2y. \right]$$

Automatic calculation: the code defines `velocity(v) = [v[1], -v[2]^2]` and calls `divergence(velocity,[x,y])` from `CalculusWithJulia.jl`.

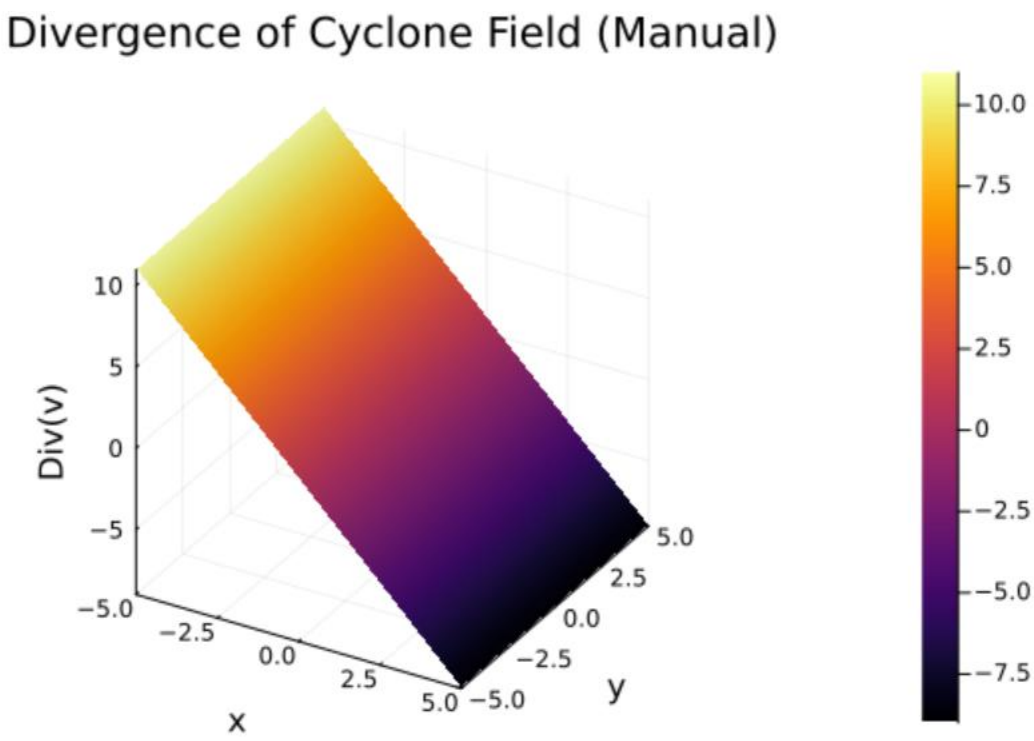
Manual calculation and plot:

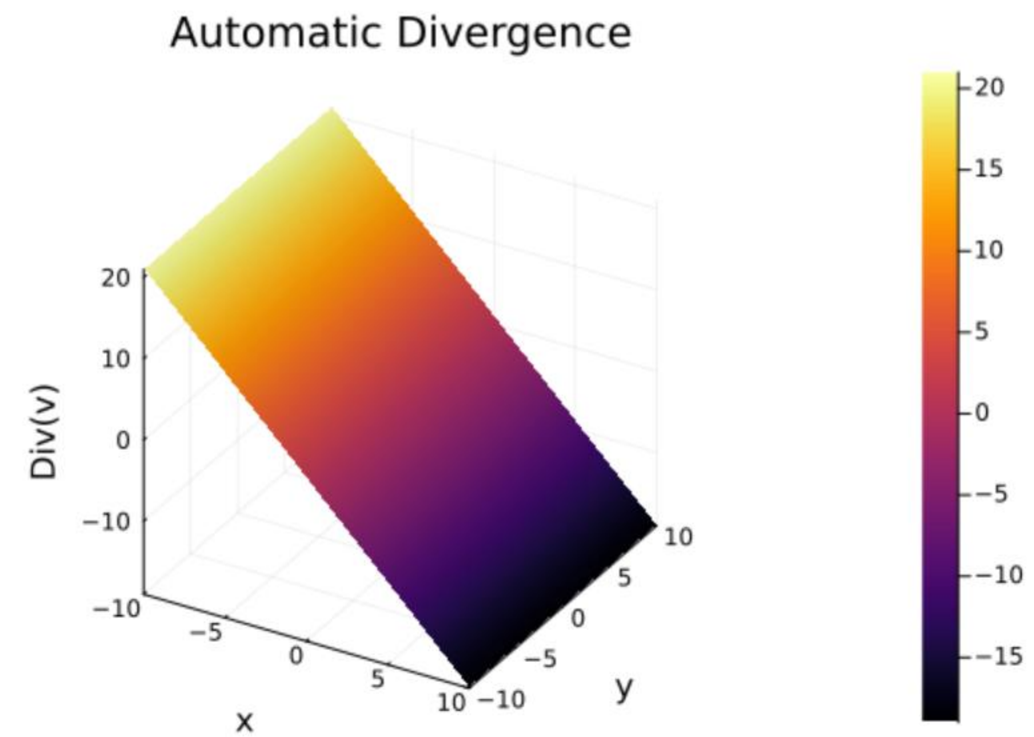
- A function `div_manualq2(x,y) = 1 - 2*y` is evaluated on a grid and surface-plotted.

Interpretation and comparison:

- Both the automatic and manual divergence surfaces should match numerically: a plane that varies linearly with y with a unit offset in x.

- Where `y < 0.5`, divergence is positive (net source); for `y > 0.5` it becomes negative (net sink). This tells how the cyclone field expands or contracts locally depending on the y-coordinate.





(c) Curl — automatic and manual

Mathematical curl (2D scalar curl / out-of-plane):

For 2D vector field (v=(v_x,v_y)) the scalar curl is

$$\left((\nabla \times v)_z = \frac{\partial\, v_y}{\partial x} - \frac{\partial\, v_x}{\partial y}. \right)$$

Here:

[$\frac{\partial v_y}{\partial x}$ = 0, \quad \partial v_x/\partial y = 0, \Rightarrow \text{curl} = 0.]

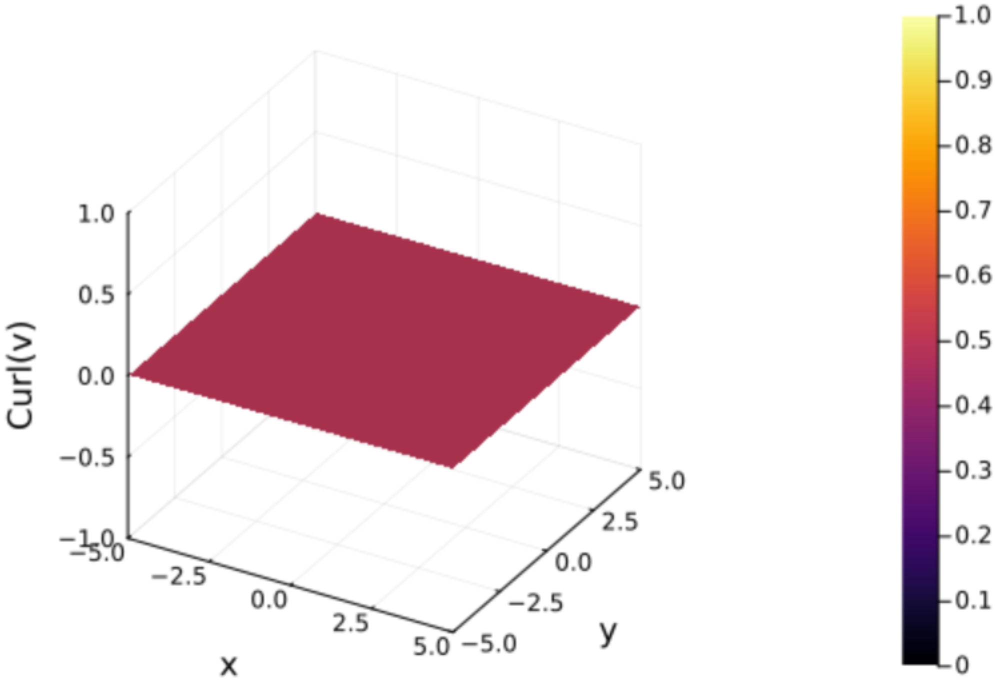
What the code does:

- Uses `curl(vel,[x,y])` to compute curl automatically (returns a scalar field for 2D case) and plots it.
- Implements `curl_manual(x,y) = 0.0` and plots a zero surface.

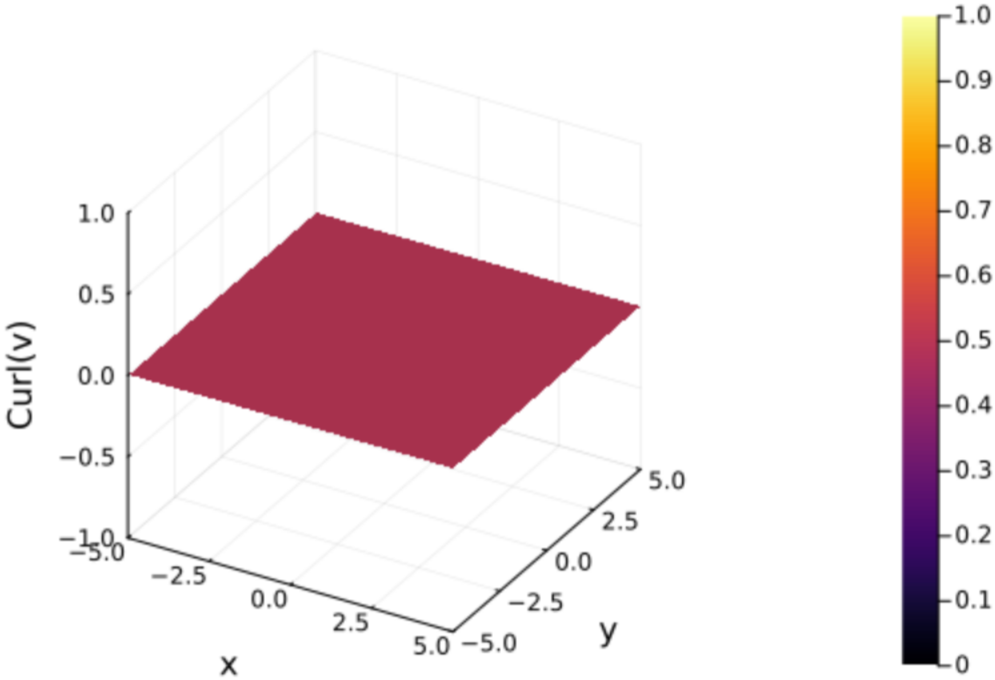
Interpretation:

- Both the automatic and manual calculations show nearly zero curl across the plane, confirming the field is irrotational (no local rotation), even though the name "cyclone" might suggest rotation — this particular model has no curl.

Automatic Curl



Curl (Manual Calculation)



4. Question 3 — River velocity field

Given:

$$\left[\, f(x,y) \;=\; e^{\{x\}}\, y^2\, e_1 + \;(x \;+\; 2y)e_2. \,\right]$$

So $\left(f = \left(e^{\{x\}}y^2,\, x + 2y\right).\, \right)$

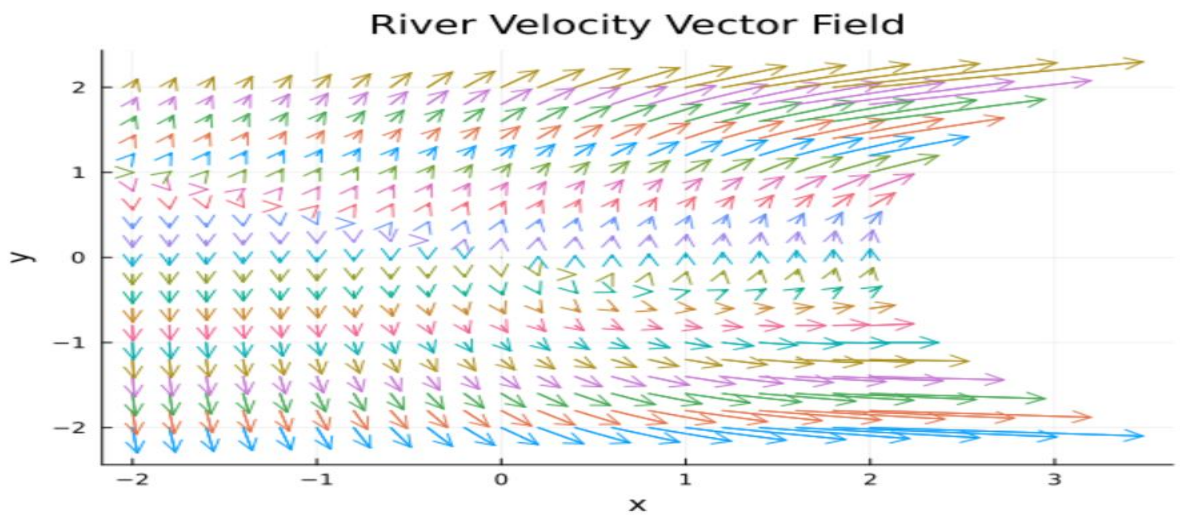
(a) Vector field plot

What the code does:

- Defines `fx(x,y)=exp(x)*y^2` and `fy(x,y)=x+2*y`.
- Builds grid and uses a scaled `quiver` to plot the flow field.

Interpretation:

- The x-component depends exponentially on x and quadratically on y — so for positive x and larger |y| the horizontal velocity grows fast.
- The y-component is linear in x and y and contributes to vertical motion.
- The pattern will show strong rightward flow in regions of positive x and significant vertical variation due to y.



(b) Divergence — automatic and manual

Mathematical divergence:

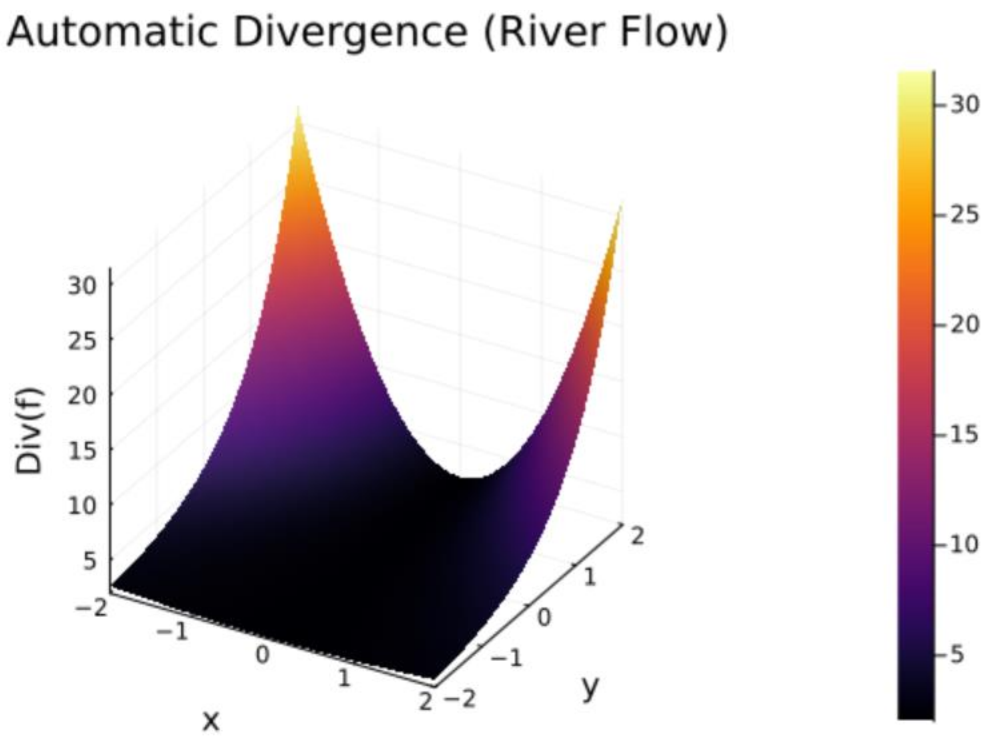
$$\left[\nabla \cdot f \ = \frac{\partial}{\partial x}\left(e^{\{x\}} y^2\right) + \frac{\partial}{\partial y}(x + 2y) = \ e^{\{x\}} y^2 + \ 2. \right]$$

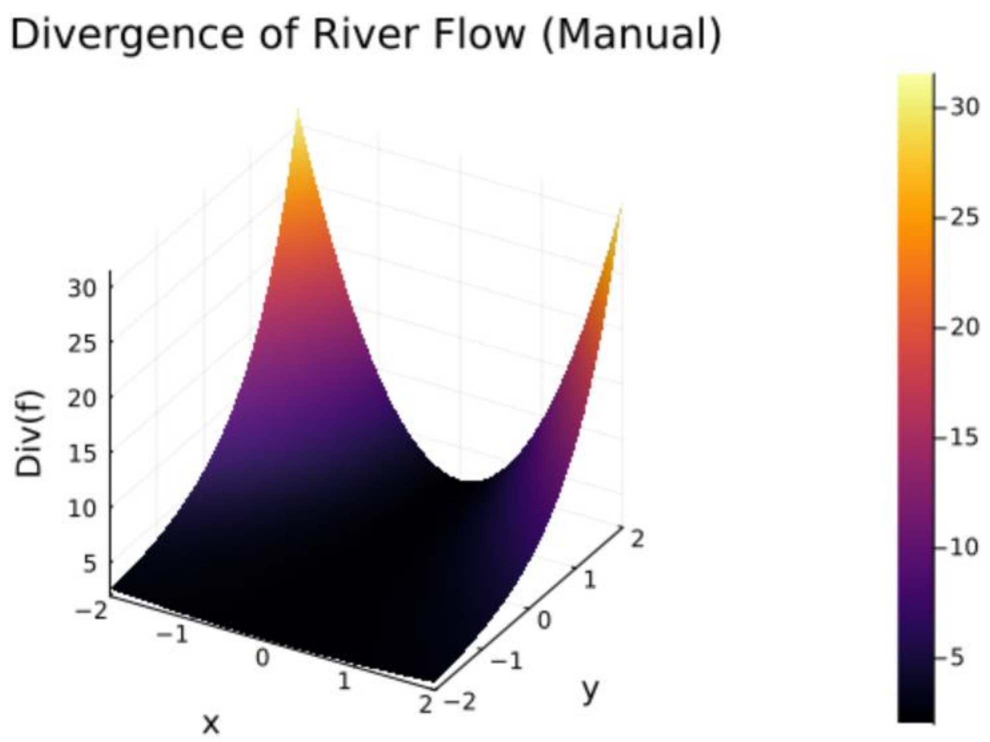
What the code does:

- Uses `divergence(f_vec, [x,y])` for automatic computation and surface plot.
- Implements manual function `div_f_manual(x,y) = exp(x)*y^2 + 2` and surface plots it.

Interpretation:

- Divergence is always shifted by +2 and increases with `exp(x)*y^2`. Regions of large positive x and large |y| show strong positive divergence (sources); negative x reduces the exponential term.
- Agreement between automatic and manual plots confirms the analytic differentiation.





(c) Curl — automatic and manual

Mathematical curl (2D scalar z-component):

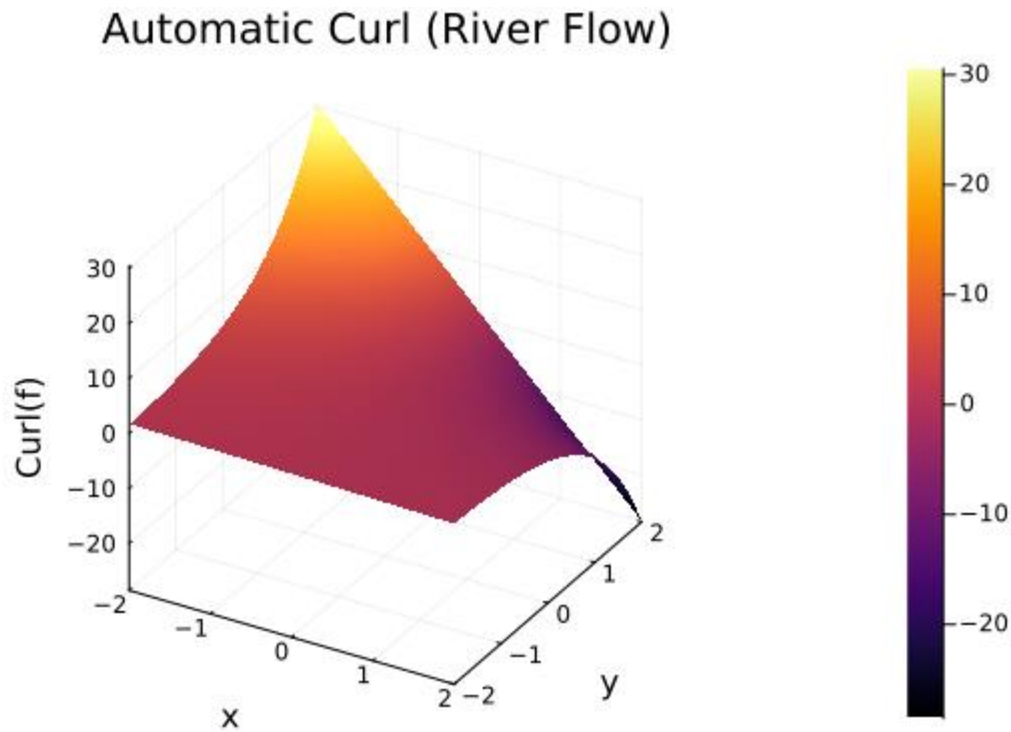
$$\left[(\nabla \times f)_z = \left[\frac{\partial}{\partial x}(x+2y), -\frac{\partial}{\partial y}\left(e^{\{x\}}y^2\right)\right] = \left[1\,, -2y\, e^{\{x\}}.\ \right] \right]$$

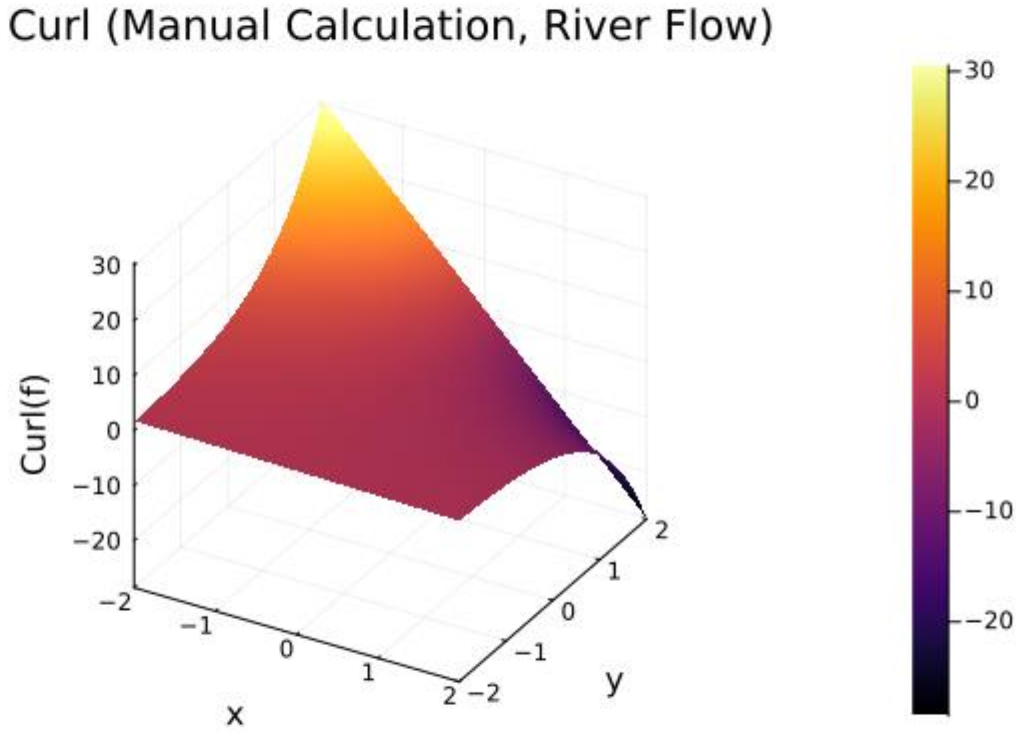
What the code does:

- Uses `curl(f_vec,[x,y])` to compute the curl automatically and creates a surface plot.
- Implements `curl_f_manual(x,y) = 1 - 2*y*exp(x)` and plots.

Interpretation:

- Curl is a function of both x and y. It generally decreases with increasing `y*e^{x}`. For small y and small/negative x, curl near 1 indicates a local clockwise/anticlockwise sense depending on sign convention; where `2y e^{x} > 1`, curl changes sign.
- Matching automatic and manual plots demonstrates correctness.


Automatic Curl (River Flow)
Curl(f)
x
y
30
20
10
0
−10
−20
−2
−1
0
1
2


Curl (Manual Calculation, River Flow)
Curl(f)
x
y
30
20
10
0
−10
−20
−2
−1
0
1
2

5. Question 4 — Beam problem (Fig 1)

Problem description: A beam of length `L_total = 1.25 * l` is loaded with a uniformly distributed load q over part or whole of beam and reactions RA and RB computed accordingly. The code is written to accept parameters `l` and q.

What the code does:

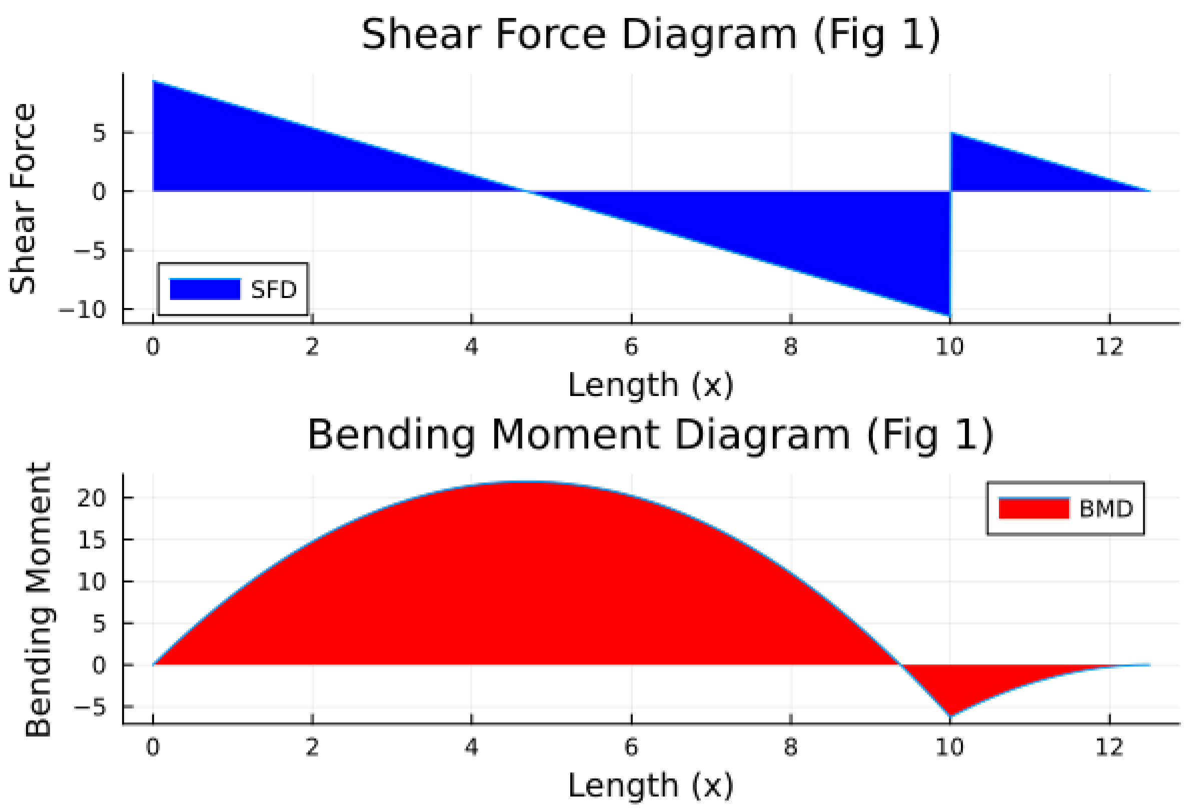
- `solve_beam_1(l,q)` constructs the geometry: `L_total = 1.25*l`.
- Reaction RB is computed by static equilibrium for the portion loaded. RA is found by subtracting RB from total vertical load.
- Defines local functions `shear_force(x)` and `bending_moment(x)` using piecewise expressions depending on x region.
- Evaluates `V_vals = shear_force.(x_vals)` and `M_vals = bending_moment.(x_vals)` on a dense x grid and plots SFD and BMD stacked vertically.

Interpretation of outputs:

- The Shear Force Diagram (SFD) shows linear segments between supports and load discontinuities caused by supports or concentrated loads.
- The Bending Moment Diagram (BMD) is quadratic in regions under uniform load (since integration of a linear shear yields a quadratic moment) and shows maxima/minima where shear crosses zero.
- The code uses fill colors to highlight area under the curves, and uses `display(plot(...,layout=(2,1)))` to show both diagrams together for easy reading.

Verification:

- Check equilibrium: `RA + RB = q * L_total` and moment equilibrium about a point should hold to numerical precision.
- Check where shear zero-crossing occurs; that x is where bending moment reaches an extremum.



6. Question 5 — Beam problem (Fig 2)

Problem description: A more complex beam with a concentrated load `P = 0.8*q*l` at `x_E = 0.4*l`, an UDL of length `l_BC = x_C - x_B` between x_B and x_C, and supports at positions x_B and x_C (and other geometry as shown in the figure).

What the code does:

- Computes P, locations `x_E, x_D, x_B, x_C` as fractions of `l`.
- Computes reactions RA, RC, and RB using static equilibrium (sum of moments and forces); details shown in code formulae.
- Implements piecewise `shear_force(x)` and `bending_moment(x)` according to the intervals: before x_E, between x_E and x_B, and between x_B and x_C.
- Evaluates arrays over `x_vals` and plots SFD and BMD similarly to Q4.

Interpretation:

- The SFD shows jumps at concentrated loads (P) and ramps over UDL regions.
- The BMD shows piecewise-polynomial shapes: linear in regions with no distributed load, quadratic under UDL, and local extrema at locations where shear equals zero.

- Reaction calculations should be validated by substituting numeric l and q and verifying equilibrium.