¡

Problem 1. Using the *ETOPO1 Global Relief Model*, compute the area of the Earth that is covered by oceans. Assume for simplicity that the Earth is a perfect sphere, so that the area element is $dA = r^2 \cos(\phi) d\phi d\lambda$.

Solution: □

Mathematically, finding the area of the ocean is equivalent to evaluating a double integral

$$A_{\text{ocean}} \;=\; \iint_{\mathcal{O}} dA,$$

where the limits of integration, $\mathcal{O}$, trace the irregular coastline. The mathematical expression for the coastline is complicated. Instead, we extend the integral over the entire sphere by replacing the integrand from 1 to a function that evaluates to 1 for points inside the ocean and 0 over land. There are multiple common notations used to express the integrand. Here are a few of them:

$$A_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \mathbf{1}_{\{\, h(\phi,\lambda)<0 \,\}} \; a^2 \cos\phi \; d\lambda \, d\phi$$

$$A_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-\infty}^{0} \delta\big(h(\phi,\lambda) - z\big) \, dz \; a^2 \cos\phi \; d\lambda \, d\phi$$

$$A_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} H\big(-\, h(\phi,\lambda)\big) \; a^2 \cos\phi \; d\lambda \, d\phi,$$

Here, the indicator function $\mathbf{1}_{\{h<0\}}$ (also written as the Iverson bracket $[h < 0]$ in computer science) is widely used in statistics and probability to mask integration domains, whereas the Dirac delta $\delta(\cdot)$ is a distributional tool favored in physics and engineering for imposing constraints inside continuous integrals; applied mathematicians and computational scientists choose between these notations depending on context.

Numerical implementation

In practice, the continuous integral

$$A_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \mathbf{1}_{\{h<0\}} \; a^2 \cos\phi \; d\lambda \, d\phi$$

is replaced by the discrete sum over the latitude–longitude grid:

$$A_{\text{ocean}} \approx \sum_{i=1}^{N_\phi} \sum_{j=1}^{N_\lambda} M_{ij} \Delta A_{ij}$$

where N_ϕ and N_λ are the number of grid points in latitude and longitude, and $\Delta\phi$, $\Delta\lambda$ are the uniform spacings.

$$\Delta A_{ij} = a^2 \cos\big(\phi_i\big)\,\Delta\phi\,\Delta\lambda, \qquad A_{\text{ocean}} = \sum_{i,j} \Delta A_{ij}\, M_{ij},$$

where

a Earth's radius.

ϕ_i, λ_j Latitude and longitude grid points (in radians).

$\Delta\phi$, $\Delta\lambda$ Uniform grid spacings.

M_{ij} Land–sea mask: $M_{ij} = 1$ if ETOPO1 height $h_{ij} < 0$, else 0.

Python Code: `prob1.py`

And now?

- **Global heat & mass budgets:** Ocean area sets the scale for heat uptake, evaporation, and air–sea gas exchange.
- **Radiation partitioning:** Accurate ocean fraction is required to split incoming solar radiation correctly between land and sea.
- **Model calibration:** Ensures that computed transports (e.g. meridional overturning) and biogeochemical fluxes have the correct global magnitude.

For students who want to push further:

- **Bonus problem:** Repeat the ocean-area calculation using an *oblate spheroid* Earth. Derive and implement the area element

$$dA \;=\; b^2\,\frac{(1-\varepsilon^2)\,\cos\phi}{\big(1-\varepsilon^2\sin^2\phi\big)^2}\,d\phi\,d\lambda, \quad \varepsilon^2 = \frac{b^2-c^2}{b^2},$$

where b and c are the equatorial and polar radii respectively, then compare your ocean fraction to the spherical result. You can find some notes relevant to this problem on Canvas in the solution set from a previous iteration of the course. `hw1prob1_sol.pdf`

Problem 2. Again, using the *ETOPO1* data, compute the volume of the world's oceans. Assume for simplicity that the volume element is $dV = a^2\cos(\phi)d\phi d\lambda dz$, where a is the mean radius of the earth.

Solution: □

Mathematically, computing the volume of the ocean is equivalent to evaluating a triple integral

$$V_{\text{ocean}} = \iiint_{\mathcal{O}} dV,$$

where the limits of integration, $\mathcal{O}$, trace the irregular ocean basin. Instead, we extend the integral over the entire sphere–depth domain by replacing the integrand with a mask and the local depth:

$$V_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-H(\phi,\lambda)}^{0} a^2 \cos\phi \, dz \, d\lambda \, d\phi,$$

where

$$H(\phi,\lambda) = \begin{cases} -h(\phi,\lambda), & h(\phi,\lambda) < 0, \\ 0, & \text{else}, \end{cases}$$

is the local ocean depth.
Equivalently,

$$V_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} -h(\phi,\lambda) \, \mathbf{1}_{\{h<0\}} \, a^2 \cos\phi \, d\lambda \, d\phi.$$

Numerical implementation

In practice, the continuous integral is replaced by the discrete sum over the ETOPO1 grid:

$$V_{\text{ocean}} \approx \sum_{i=1}^{N_\phi} \sum_{j=1}^{N_\lambda} \left(-h_{ij}\right) M_{ij} \, a^2 \cos(\phi_i) \, \Delta\phi \, \Delta\lambda,$$

where

N_ϕ, N_λ Number of latitude and longitude grid points.

$\Delta\phi, \Delta\lambda$ Uniform spacings in radians.

h_{ij} ETOPO1 height (negative underwater).

$M_{ij} = \mathbf{1}_{\{h_{ij}<0\}}$ Land–sea mask.

Python Code: `prob2.py`

And now?

- **Compute the hypsometric curve:** Bin the ocean floor depths $h_{ij} < 0$ into intervals z_k, sum the corresponding cell areas $\Delta A_{ij} = a^2 \cos(\phi_i) \, \Delta\phi \, \Delta\lambda$ at each depth, and plot the normalized distribution of ocean area versus depth. Discuss the modal depths and compare to published global hypsometric curves. Have a look at my Youtube video showing how I did it in Julia as part of a previous iteration of this course Spherical Coordinates Part III

Problem 3. Use the *2023 World Ocean Atlas* objectively-analyzed annually averaged temperature and salinity data to compute the volumetric averages of temperature and salinity of the ocean.

Solution: □

Mathematically, the volumetric average of a field X (temperature T or salinity S) over the ocean is

$$\langle X\rangle = \frac{1}{V_{\text{ocean}}}\iiint_{\mathcal{O}} X(\phi,\lambda,z)\,dV,$$

where

$$dV = a^2\cos\phi\,d\phi\,d\lambda\,dz,\quad V_{\text{ocean}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2\pi}\int_{-H(\phi,\lambda)}^{0} a^2\cos\phi\,dz\,d\lambda\,d\phi,$$

and $H(\phi,\lambda)=\max\{0,-h(\phi,\lambda)\}$ is the local ocean depth.

$$\text{Equivalently, } \langle X\rangle = \frac{\displaystyle\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2\pi}\int_{-H(\phi,\lambda)}^{0} X(\phi,\lambda,z)\,a^2\cos\phi\,dz\,d\lambda\,d\phi}{\displaystyle\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2\pi}\int_{-H(\phi,\lambda)}^{0} a^2\cos\phi\,dz\,d\lambda\,d\phi}.$$

Numerical implementation

In practice, we replace the integrals by discrete sums over latitude (i), longitude (j), and depth (k):

$$\langle X\rangle \approx \frac{\sum_{i=1}^{N_\phi}\sum_{j=1}^{N_\lambda}\sum_{k=1}^{N_z} X_{ijk}\,M_{ij}\,\Delta V_{ijk}}{\sum_{i=1}^{N_\phi}\sum_{j=1}^{N_\lambda}\sum_{k=1}^{N_z} M_{ij}\,\Delta V_{ijk}},$$

with

$$\Delta V_{ijk} = a^2\cos(\phi_i)\,\Delta\phi\,\Delta\lambda\,\Delta z_k,\quad M_{ij} = \mathbf{1}_{\{h_{ij}<0\}}.$$

Python Code: `prob3.py`

And now?

- Compute basin-by-basin averages (Atlantic, Pacific, Indian) by applying geographic masks.
- Compare your global mean T and S to canonical values (~ 3.5°C, ~ 35 PSU).

Problem 4. Using the *2023 World Ocean Atlas* objectively analyzed monthly temperature and salinity data, plot vertical profiles of temperature and salinity for a water column in the North Atlantic ($\sim 45°$ N) during winter, spring, summer, and fall. Your plot should have depth along the vertical axis and the temperature or salinity along the horizontal axis. Make sure you label your axes clearly.

Solution: □

At the fixed location $(\phi_0, \lambda_0) \approx (45°\text{N}, \dots)$, we directly load the objectively-analyzed seasonal profiles from WOA:

$$T_{s,k} = T(\phi_0, \lambda_0, z_k, s), \qquad S_{As,k} = \mathrm{S}_\mathrm{A}\big(S_P(\phi_0, \lambda_0, z_k, s), p_k, \lambda_0, \phi_0\big),$$

where S_P is practical salinity and S_A the absolute salinity (g/kg) computed via the TEOS-10 GSW toolbox. Plot each $T_{s,k}$ and $S_{As,k}$ versus depth z_k, with depth (m) on the vertical axis and temperature (°C) or absolute salinity (g/kg) on the horizontal axis.

Numerical implementation

1. Read the seasonal WOA NetCDF files for $s \in \{\mathrm{DJF}, \mathrm{MAM}, \mathrm{JJA}, \mathrm{SON}\}$.

Python Code: `prob4.py`

And now?

- Compute mixed-layer depth for each season using a temperature-based criterion on the seasonal profiles.
- Compare your absolute-salinity profiles to the practical-salinity based profiles and discuss differences.
- Create a T–S diagram at 200 m for each season to illustrate water-mass changes.

Problem 5. Using the *TEOS-10 Thermodynamic Equation of Seawater-2010* and the *2023 World Ocean Atlas* objectively analyzed temperature and salinity, compute:

a. the volumetrically averaged thermal expansion coefficient for seawater in the top 200 m of the ocean

b. the volumetrically average haline contraction coefficient for seawater in the top 200 m of the ocean

c. the total mass of salt in the ocean

d. the total mass of freshwater in the ocean

Solution: □

Using TEOS-10, we define at each point the thermal expansion and haline contraction coefficients

$$\alpha = \alpha\big(SA, CT, p\big), \qquad \beta = \beta\big(SA, CT, p\big),$$

and compute their volumetric averages over the top 200 m:

$$\langle \alpha \rangle_{0-200} = \frac{1}{V_{0-200}} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-200}^{0} \alpha\big(SA, CT, p\big)\ a^2 \cos\phi \, dz \, d\lambda \, d\phi,$$

$$\langle \beta \rangle_{0-200} = \frac{1}{V_{0-200}} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-200}^{0} \beta\big(SA, CT, p\big)\ a^2 \cos\phi \, dz \, d\lambda \, d\phi,$$

where

$$V_{0-200} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-200}^{0} a^2 \cos\phi \, dz \, d\lambda \, d\phi$$

is the ocean volume in the top 200 m.

The total salt mass is

$$M_{\text{salt}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-H(\phi,\lambda)}^{0} SA(\phi, \lambda, z)\, \rho\big(SA, CT, p\big)\ a^2 \cos\phi \, dz \, d\lambda \, d\phi,$$

and the total water mass is

$$M_{\text{water}} = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{0}^{2\pi} \int_{-H(\phi,\lambda)}^{0} \rho\big(SA, CT, p\big)\ a^2 \cos\phi \, dz \, d\lambda \, d\phi,$$

so that the freshwater mass is $M_{\text{water}} - M_{\text{salt}}$.

Numerical implementation

Discretize over latitude i, longitude j, and depth levels k, with $\Delta V_{ijk} = a^2 \cos(\phi_i)\, \Delta\phi\, \Delta\lambda\, \Delta z_k$ and land–sea mask $M_{ij} = \mathbf{1}_{\{h_{ij}<0\}}$. Compute:

$$\langle \alpha \rangle_{0-200} \approx \frac{\sum_{i,j,k: z_k \geq -200} \alpha_{ijk}\, M_{ij}\, \Delta V_{ijk}}{\sum_{i,j,k: z_k \geq -200} M_{ij}\, \Delta V_{ijk}},$$

$$\langle \beta \rangle_{0-200} \approx \frac{\sum_{i,j,k: z_k \geq -200} \beta_{ijk}\, M_{ij}\, \Delta V_{ijk}}{\sum_{i,j,k: z_k \geq -200} M_{ij}\, \Delta V_{ijk}},$$

$$M_{\text{salt}} \approx \sum_{i,j,k} SA_{ijk}\, \rho_{ijk}\, M_{ij}\, \Delta V_{ijk}, \qquad M_{\text{water}} \approx \sum_{i,j,k} \rho_{ijk}\, M_{ij}\, \Delta V_{ijk}.$$

Python Code: `prob5.py`

And now?

- Test sensitivity of $\langle \alpha \rangle$ and $\langle \beta \rangle$ to the depth threshold (e.g. 100 m, 500 m).
- Map the spatial distribution of local α and β at selected depths.
- Compare total salt mass estimate to climatological values ($\sim 3.5 \times 10^{16}$ kg).

Problem 6. Let $\boldsymbol{x} = (x, y, z)$ be a fixed spatial position in a body of fluid. The coordinate $\boldsymbol{x}$ is called an *Eulerian* coordinate system. Suppose we assign a unique label to every fluid particle. This can be done by using the coordinate of the particle at time $t = 0$ as the label, $\boldsymbol{X} = \boldsymbol{X}(t, \boldsymbol{x}) = (X, Y, Z)$, where $X|_{t=0} = x$, $Y|_{t=0} = y$, and $Z|_{t=0} = z$.

The coordinate $\boldsymbol{X}$ is referred to as a *Lagrangian* coordinate system. As the fluid particles move around their Eulerian coordinates change, but each particle keeps its Lagrangian label. In general, each coordinate system may be transformed into the other:

$$\begin{aligned} \text{Eulerian: } & \boldsymbol{x}, t \quad \boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t) \\ \text{Lagrangian: } & \boldsymbol{X}, t \quad \boldsymbol{X} = \boldsymbol{X}(\boldsymbol{x}, t) \end{aligned} \tag{1}$$

Suppose that at time $t = 0$, the temperature varies in space according to

$$T_0(\boldsymbol{x}) = b + cy, \tag{2}$$

where $b = 20°\text{C}$ and $c = (-10°\text{C})/(100\text{km})$. Suppose also that the Eulerian velocity field (in polar coordinates) is given by

$$\begin{aligned} v_r &= 0, \\ v_\theta &= \Gamma \times \begin{cases} r/a^2 & r \le a, \\ 1/r & r > a, \end{cases} \end{aligned} \tag{3}$$

for $a = 100$ km, and $\Gamma = 2\pi a^2/(1 \text{ day})$.

(a) Assuming $\partial \boldsymbol{v}/\partial t = 0$, compute $D\boldsymbol{v}/Dt$.

Solution: □

The velocity field is purely in the tangential direction with no radial component. Moreover, $\partial \boldsymbol{v}/\partial t = 0$ indicates that the velocity field is independent of time. As a result, material particles exhibit uniform circular motion. The acceleration for uniform circular motion is v_θ^2/r directed radially inward toward the center of the circle. Thus,

$$\begin{aligned} \frac{D\boldsymbol{v}}{Dt} &= -\frac{v_\theta^2}{r}\hat{\mathbf{r}} \\ &= \Gamma^2 \times \begin{cases} r/a^4 \text{ for } r \le a, \\ 1/r^3 \text{ for } r > a, \end{cases} \end{aligned} \tag{4}$$

(b) Assuming $DT/Dt = 0$, make a plot of T as a function of t for $t \in [0, 2 \text{ days}]$ following the particle that started at $(x, y) = (125 \text{ km}, 0)$ at $t = 0$.

Solution: □

Because $DT/Dt = 0$ for all t, T is a constant following the particle. Thus if $\Theta(t)$ is the temperature of the particle then

$$\frac{d\Theta}{dt} = 0.$$

Integrating w.r.t. t yields $\Theta(t) = A$ where A is the constant of integration. To determine A we use the initial condition. At time $t = 0$ the particle is at the point $(x, y) = (125 \text{ km}, 0)$ where it's temperature is

$$T_0(125 \text{ km}, 0) = 20°\text{C}.$$

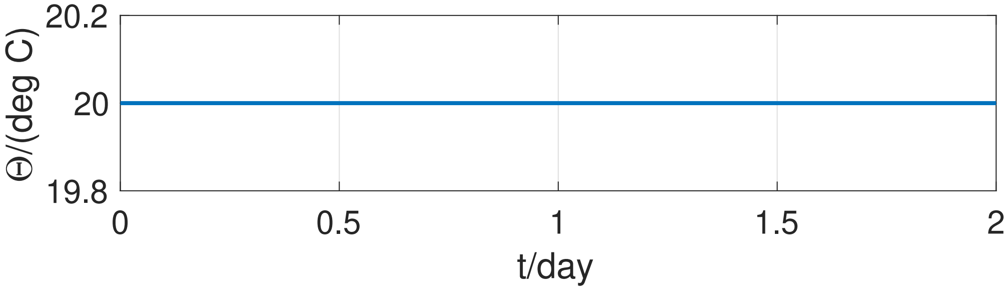

Figure 1: Temperature, $\Theta(t)$, of the Lagrangian particle located at $(x, y) = (125$ km, $0)$ at time $t = 0$, plotted as a function of time. The graph is a constant because $DT/Dt = 0$, which means that Lagrangian particles conserve their temperature.

(c) Still assuming that $DT/Dt = 0$, and also assuming $\partial\boldsymbol{v}/\partial t = 0$, make a plot of T as a function of t at position $(x, y) = (125 \text{ km}, 0)$ for $t \in [0, 2 \text{ days}]$.

Solution: □

Given a position described by the Cartesian coordinates (x, y) we can compute the polar coordinates $(\theta, r) = (\text{atan2}(y, x), \sqrt{x^2 + y^2})$. Given the polar coordinates, we can compute the point's tangential velocity, $v_\theta(r)$, and the angular frequency $\omega(r) = v_\theta(r)/r$. The initial θ coordinate of the particle at (x, y) at time t is thus

$$\theta_0(x, y, t) = \text{atan2}(y, x) - \omega(r)t,$$

so that the temperature of the particle is

$$T(x, y, t) = b + cr\sin(\theta_0(t)).$$

The Matlab script does the rest.

Matlab script

```
% Temperature field at time t = 0
b = 20; %  ( deg C)
c = -10 / 300; %  ( deg C/km)
T0 = @(x,y) b + c * y;
% Velocity field (Rankine vortex)
a = 100; %(km)
Gamma = 2 * pi * a^2 / ( 24 * 60^2 ); % (s^-1)
v_theta  = @( r ) Gamma * ( ( r / a^2 ) .* ( r <= a ) + ( 1 ./ r) .* ( r > a ) );
% cartesian to polar coordinate conversion functions
r = @( x, y ) sqrt( x.^2 + y.^2 );
theta = @( x, y ) atan2( y, x );
% angular frequency in radians as a function of radius
omega = @( r ) v_theta( r ) ./ r;
% initial theta coordinate as a function of (x,y,t)
theta_0 = @( x, y, t ) theta( x, y ) - omega( r( x, y ) ) * t;
% initial X and Y coordinate as a function of (x,y,t)
Y_0 = @( x, y, t ) r( x, y ) .* sin( theta_0( x, y, t) );
X_0 = @( x, y, t ) r( x, y ) .* cos( theta_0( x, y, t) );
% temperature as a function of (x,y,t)
T = @( x, y, t ) T0( X_0( x y, t ), Y_0( x, y, t ) );
% make a mesh
x = -300 : 1 : 300; % (km)
y = -300 : 1 : 300; % (km)
[X,Y] = meshgrid( x, y );
t = 24 * 60^2; % seconds
% make the plot
contourf( X, Y, T( X, Y, t ),[ 0 : 1 : 40 ]);
xlabel( 'x/km' );
ylabel( 'y/km' );
H = colorbar;
ylabel( H, 'deg C' )
set( gca, 'FontSize', 16 );
```

(d) Make a contour plot of the T field at time $t = 1$ day.

Solution: □

Given a position described by the Cartesian coordinates (x, y) we can compute the polar coordinates $(\theta, r) = (\text{atan2}(y, x), \sqrt{x^2 + y^2})$. Given the polar coordinates, we can compute the point's tangential velocity, $v_\theta(r)$, and the angular frequency $\omega(r) = v_\theta(r)/r$. The initial

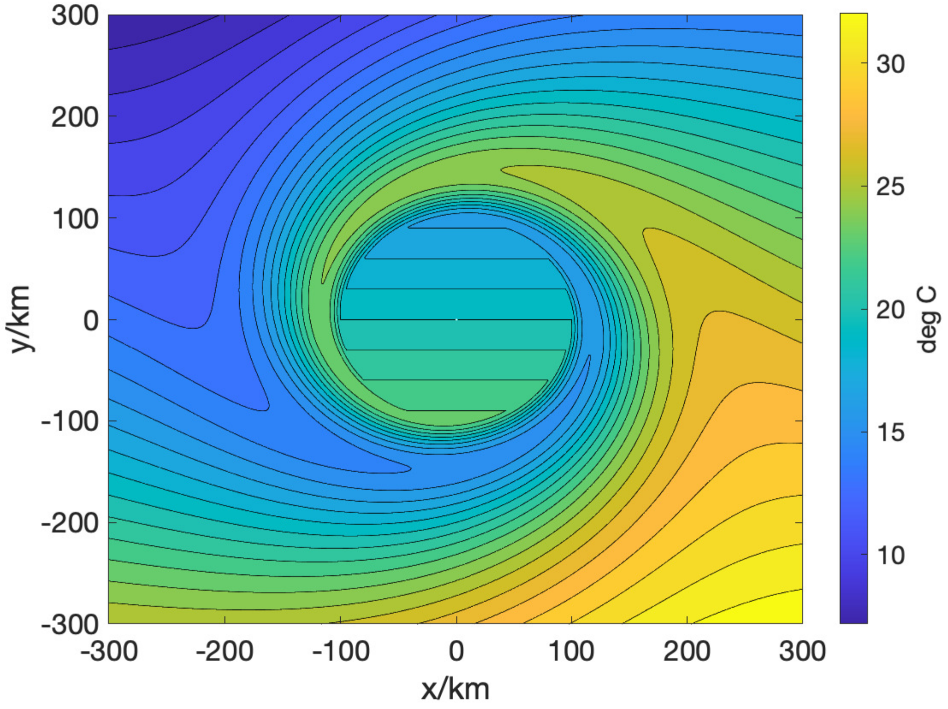

Figure 2: Temperature field at time $t = 1$ day. Notice that the fluid located in the disk with $r < a$ rotates as a solid body at a rate of 2π/day and therefore, after 1 day, the temperature in the inner disk has returned to its starting value. **Notice that for this figure I chose $c = -10°\mathbf{C}/300\mathbf{km}$ rather than $c = -10°\mathbf{C}/300\mathbf{km}$. I really meant to have 300 km in the denominator to keep the temperature in a plausible range.**

θ coordinate of the particle at (x, y) at time t is thus

$$\theta_0(x, y, t) = \text{atan2}(y, x) - \omega(r)t,$$

so that the temperature of the particle is

$$T(x, y, t) = b + cr \sin(\theta_0(t)) .$$

The Matlab script does the rest.

Matlab script

```
% Temperature field at time t = 0
b = 20; %  ( deg C)
c = -10 / 300; %  ( deg C/km)
T0 = @(x,y) b + c * y;
% Velocity field (Rankine vortex)
a = 100; %(km)
Gamma = 2 * pi * a^2 / ( 24 * 60^2 ); % (s^-1)
v_theta  = @( r ) Gamma * ( ( r / a^2 ) .* ( r <= a ) + ( 1 ./ r) .* ( r > a ) );
% cartesian to polar coordinate conversion functions
r = @( x, y ) sqrt( x.^2 + y.^2 );
theta = @( x, y ) atan2( y, x );
% angular frequency in radians as a function of radius
omega = @( r ) v_theta( r ) ./ r;
% initial theta coordinate as a function of (x,y,t)
theta_0 = @( x, y, t ) theta( x, y ) - omega( r( x, y ) ) * t;
% initial X and Y coordinate as a function of (x,y,t)
Y_0 = @( x, y, t ) r( x, y ) .* sin( theta_0( x, y, t) );
X_0 = @( x, y, t ) r( x, y ) .* cos( theta_0( x, y, t) );
% temperature as a function of (x,y,t)
T = @( x, y, t ) T0( X_0( x y, t ), Y_0( x, y, t ) );
% make a mesh
x = -300 : 1 : 300; % (km)
y = -300 : 1 : 300; % (km)
[X,Y] = meshgrid( x, y );
t = 24 * 60^2; % seconds
% make the plot
contourf( X, Y, T( X, Y, t ),[ 0 : 1 : 40 ]);
xlabel( 'x/km' );
ylabel( 'y/km' );
H = colorbar;
ylabel( H, 'deg C' )
set( gca, 'FontSize', 16 );
```

(e) Make a contour plot of the magnitude of the gradient of T at time $t = 1$ day.

The following Matlab script makes a plot of T at $t = 0$ and a quiver plot of $\boldsymbol{v}$.

Matlab script

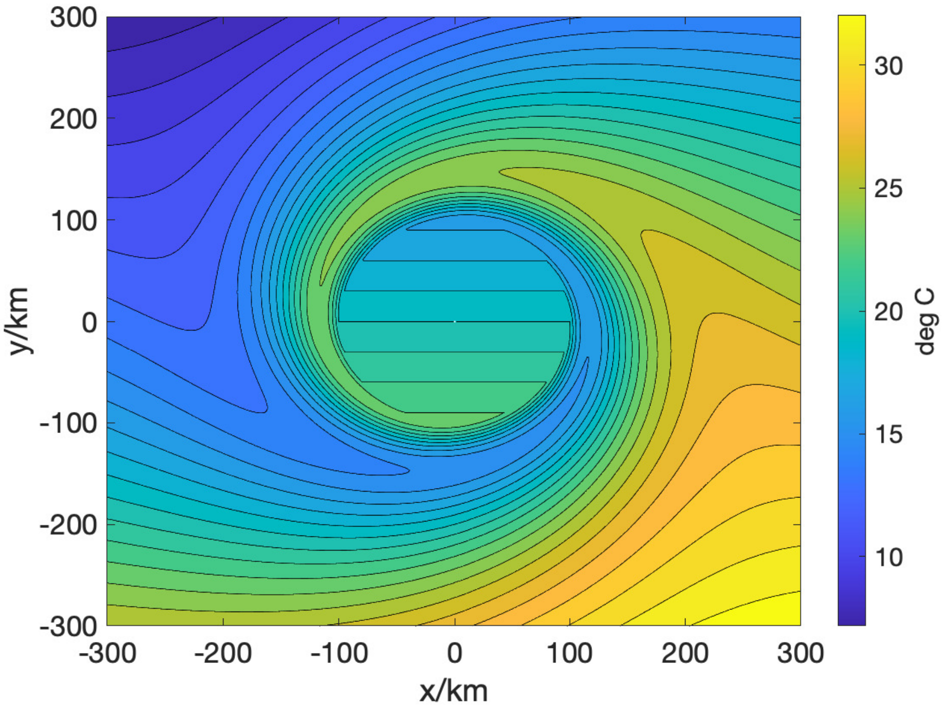

Figure 3: Temperature field at time $t = 1$ day. Notice that the fluid located in the disk with $r < a$ rotates as a solid body at a rate of 2π/day and therefore, after 1 day, the temperature in the inner disk has returned to its starting value. **Notice that for this figure I chose $c = -10°\mathbf{C}/300\mathbf{km}$ rather than $c = -10°\mathbf{C}/100\mathbf{km}$. I really meant to have 300 km in the denominator to keep the temperature in a plausible range.**

```
% make a mesh
x = -300:50:300; % (km)
y = -300:50:300; % (km)
[X0,Y0] = meshgrid(x,y);

% Temperature field at time t = 0
b = 20; %  ( deg C)
c = -10/300; %  ( deg C/km)
T0 = @(x,y) b+c*y;

figure(1)
contourf(X0,Y0,T0(X0,Y0));
H = colorbar;
ylabel(H,'deg C')
xlabel('x/km');
```

```
ylabel('y/km');
set(gca,'FontSize',16);
axis image

% Velocity field (Rankine vortex)
a = 100; %(km)
Gamma = 2*pi*a^2/(24*60^2); % (s^-1)
v_theta  = @(r) Gamma*( ( r / a^2 ) .* ( r <= a ) + ( 1 ./ r) .* ( r > a ) );

R = @(x,y) sqrt( x.^2 + y.^2);
s = @(x,y) y./R(x,y);
c = @(x,y) x./R(x,y);
u = @(x,y) -v_theta(R(x,y)).*s(x,y);
v = @(x,y)  v_theta(R(x,y)).*c(x,y);

figure(2)
quiver(X0,Y0,u(X0,Y0),v(X0,Y0));
xlabel('x/km');
ylabel('y/km');
set(gca,'FontSize',16);
axis image
```

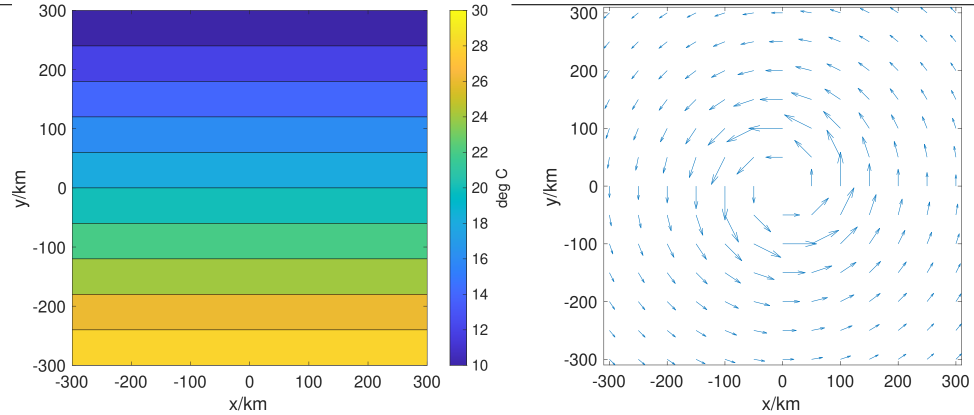

Figure 4: Left: Temperature field in °C. Right: Velocity field. The length of the arrows is proportional to the magnitude of the velocity.

Solution: □ I'm going to do this problem in two ways. The first way will use a *numerical approximation* called finite differences and the second way will use our recently honed calculus skills. (For more on finite-differences take ESS212.)

First method: Centered finite differences

The partial derivatives of field evaluated on a mesh of equally spaced points can be approximated using finite differences:

$$\frac{\partial f}{\partial x}(x,y) \approx \frac{f(x+\Delta x, y) - f(x-\Delta x, y)}{2\Delta x},$$
$$\frac{\partial f}{\partial y}(x,y) \approx \frac{f(x, y+\Delta y) - f(x, y-\Delta y)}{2\Delta y}.$$

As long as Δx and Δy are small compared to the length scales over which f varries, the finite difference approximation will be accurate.

The Matlab script shows how this approximation can easily be implemented on a computer. **Second method:** Analytic derivative using the chain rule

As we have already deduced, the particles move around in circles centered at the origin. As they move they carry their initial temperature with them. We can therefore express the temperature field at any time using the inverse function $\boldsymbol{X}^{-1}$ to recover the initial position of each Lagrangian particle. In general this function is not easily obtained, but for the particular flow field given in this problem the flow is simple enough that we can construct $\boldsymbol{X}^{-1}$ by simply rotating each point through an angle $|tv_\theta(r)/r|$ in the clockwise direction. This

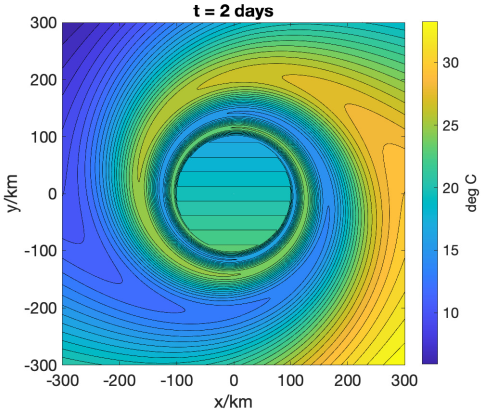

Figure 5: For reference, the temperature field in °C after 2 days, plotted with a resolution of 1 km × 1 km, assuming $DT/Dt = 0$ and $\partial \boldsymbol{v}/\partial t = 0$.

rotation is given by the following expression

$$
\begin{aligned}
x_0 &= x\cos\left(-\frac{v_\theta(r)}{r}t\right) - y\sin\left(-\frac{v_\theta(r)}{r}t\right)\\
y_0 &= x\sin\left(-\frac{v_\theta(r)}{r}t\right) + y\cos\left(-\frac{v_\theta(r)}{r}t\right)
\end{aligned}
$$

where $r = \sqrt{x^2 + y^2}$. The negative sign in the argument of the sine and cosine functions is needed for the clockwise rotation. Using the above expression we can express T at time t as follows

$$T(x, y) = T_0(y_0(x, y, t)).$$

The gradient can then be computed using the chain rule

$$
\begin{aligned}
\nabla T =& \left(D_1T_0(x_0, y_0)\right)\nabla x_0 + D_2T_0(x_0, y_0)\nabla y_0\\
=& c\nabla y_0.
\end{aligned}
$$

We must therefore compute the gradient of $y_0(x, y)$. Using the chain rule again, we get

$$
\begin{aligned}
\nabla y_0 = &\left[\sin\left(-\frac{v_\theta(r)}{r}t\right) + \left(x\cos\left(-\frac{v_\theta(r)}{r}t\right) - y\sin\left(-\frac{v_\theta(r)}{r}t\right)\right)t\frac{\partial}{\partial r}\left(\frac{v_\theta(r)}{r}\right)\frac{\partial r}{\partial x}\right]\boldsymbol{i}\\
&+\left[\cos\left(-\frac{v_\theta(r)}{r}t\right) + \left(x\cos\left(-\frac{v_\theta(r)}{r}t\right) - y\sin\left(-\frac{v_\theta(r)}{r}t\right)\right)t\frac{\partial}{\partial r}\left(\frac{v_\theta(r)}{r}\right)\frac{\partial r}{\partial y}\right]\boldsymbol{j},
\end{aligned}
$$

The above expressions show that we must also evaluate the derivative of $v_\theta(r)/r$ with respect to r and the Cartesian components of the gradient of r:

$$\frac{\partial}{\partial r}\left(\frac{v_\theta(r)}{r}\right) = \Gamma \times \begin{cases} 0 \text{ if } r \leq a \\ -2/r^3 \text{ if } r > a \end{cases},$$
$$\frac{\partial r}{\partial x} = \frac{x}{r},$$
$$\frac{\partial r}{\partial y} = \frac{y}{r}.$$

The Matlab script puts it all together.

Matlab script

```matlab
% Temperature field at time t = 0
b = 20; %  ( deg C)
c = -10 / 300; %  ( deg C/km)
T0 = @(x,y) b + c * y;

% Velocity field (Rankine vortex)
a = 100; %(km)
Gamma = 2 * pi * a^2 / ( 24 * 60^2 ); % (s^-1)
v_theta  = @( r ) Gamma * ( ( r / a^2 ) .* ( r <= a ) + ( 1 ./ r) .* ( r > a ) );

% Cartesian to polar coordinate conversion functions
r = @( x, y ) sqrt( x.^2 + y.^2 );
theta = @( x, y ) atan2( y, x );
% angular frequency in radians as a function of radius
omega = @( r ) v_theta( r ) ./ r;

% make a mesh
dx = 2; %(km)
dy = 2; %(km)
x = -300 : dx : 300; % (km)
y = -300 : dy : 300; % (km)
[X,Y] = meshgrid( x, y );
t = 24 * 60^2; % seconds

% make successive plots of T as t increases from 0 to 1 day.
dt = 60*60;
R = sqrt( X.^2 + Y.^2);
omega = v_theta( R ) ./ R ;

t = 0;
```

```
for k = 1:24
    t = t+dt;
    x0 = X .* cos( -omega * t ) - Y .* sin( -omega * t );
    y0 = X .* sin( -omega * t ) + Y .* cos( -omega * t );
    T = b + c .* y0;
    figure(1)
    contourf( X, Y, T, [0:0.5:40]);
    H = colorbar;
    ylabel( H, 'deg C / km ' )
    set( gca, 'FontSize', 16 );
    title(num2str(t/(24*60^2)));
    xlabel('x/km');
    ylabel('y/km');
    axis image
    drawnow
end

%
% Use centered finite differences to compute the gradient
%
t = 24 * 60 * 60;
R = sqrt( X.^2 + Y.^2 );
omega = v_theta( R ) ./ R;
y0 = X .* sin( - omega * t ) + Y .* cos( -omega * t );
T = b + c .* y0;
D_x = @(f,dx) ( f( 2 : end - 1, 3 : end     ) - f( 2 : end - 1, 1 : end - 2 ) ) / (2*dx);
D_y = @(f,dy) ( f( 3 : end    , 2 : end - 1 ) - f( 1 : end - 2, 2 : end - 1 ) ) / (2*dy);
%
T_xfd = D_x(T,dx);
T_yfd = D_y(T,dy);
mag1 = sqrt( T_xfd.^2 + T_yfd.^2 );

% Use the chain rule to manage the complexity of computing the gradient analytically
% _x denotes partial w.r.t. x
% _y denotes partial w.r.t. y
% _r denotes partial w.r.t. r
R_x = X ./ R;
R_y = Y ./ R;
omega_r = Gamma * ( ( -2 ./ R.^3) .* ( R > a) );
omega_x = omega_r .* R_x;
omega_y = omega_r .* R_y;
y0_x = sin( -omega * t ) - t * ( X .* cos( -omega * t ) - Y .* sin( -omega * t ) ) .* omega_x ;
```

```
y0_y = cos( -omega * t ) - t * ( X .* cos( -omega * t ) - Y .* sin( -omega * t ) ) .* omega_y ;
T_x = c .* y0_x;
T_y = c .* y0_y;
mag2 = sqrt( T_x.^2 + T_y.^2);

figure(2)
subplot(1,2,1)
contourf( X(2:end-1,2:end-1), Y(2:end-1,2:end-1), mag1 );
H = colorbar;
ylabel( H, 'deg C / km ' )
title('FD, what''s the difference?');
set( gca, 'FontSize', 16 );
axis image

subplot(1,2,2)
contourf( X, Y, mag2 );
H = colorbar;
ylabel( H, 'deg C / km ' )
title('The chain rule rules');
set( gca, 'FontSize', 16 );
axis image
```

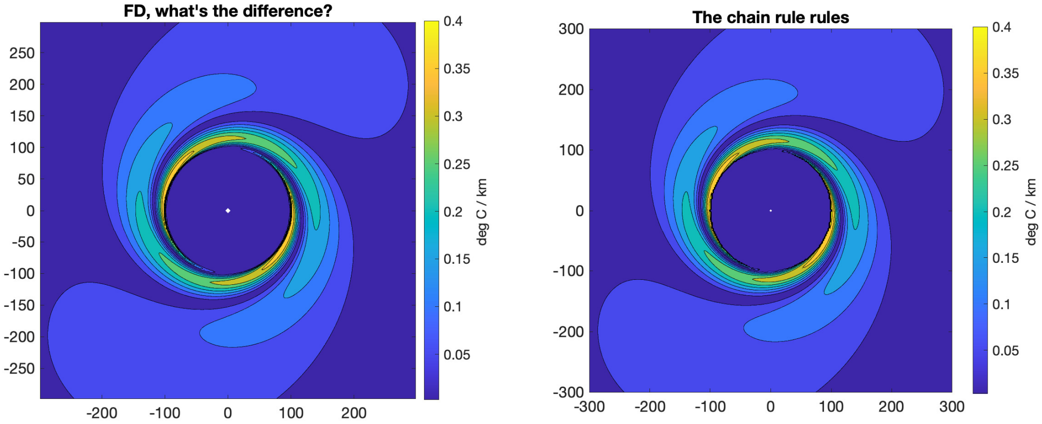

Figure 6: Contour plot of the $|\nabla T|$ at $t = 1$ day computed using centered finite differences with $\Delta x = \Delta y = 2$ km (left panel) and computed analytically using the chain rule. The agreement is pretty good at $t = 1$ day. However, at larger t the magnitude of the gradient continues to increase and the finite difference approximation will incur a larger error unless the size of Δx and Δy is decreased. The region with the largest $|\nabla T|$ are those where the diffusive fluxes of temperature will be largest. Also notice that the mesh points on the border of the domain are missing for the finite difference approximations. The centered difference method doesn't work on the boundaries. There are however one-sided schemes that would allow us to evaluate the gradient on the boundary of the domain.