

The G-traj model*

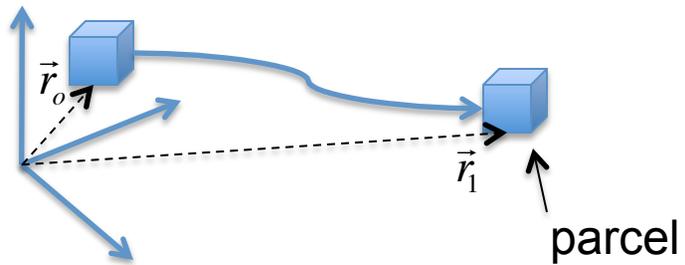
SIVO/ASTG

June 4 2009

*Developed by Schoeberl et al at NASA/GSFC Laboratory for Atmospheres.

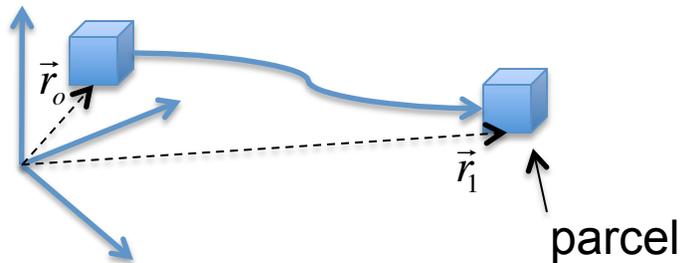
How does it work?

- What is a trajectory?



How does it work?

- What is a trajectory?



- To simulate a trajectory Gtraj uses prescribed wind fields to mathematically simulate the motion of air parcels.

$$\frac{d\vec{r}}{dt} = \vec{v}$$

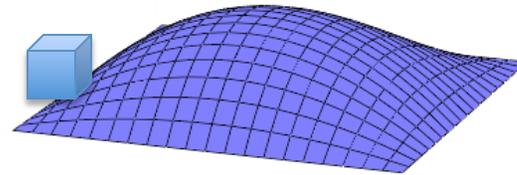
How does it work?

- Model can represent trajectory under various formulations:
- Kinematic Trajectory Models - use “w” field from 3D GCM or assimilation
 - Advantages
 - Consistent with CTM model transport
 - Can resolve mesoscale convective motions and cloud transport if 3D model can.
 - Disadvantages
 - Inadequate time resolution of adiabatic motion can lead to enhanced diabatic drift
- Isentropic/Diabatic Trajectory Models
 - Advantages
 - Conservation of entropy - adiabatic motions removed
 - Diabatic motion usually more accurate than vertical motion field since motion dependent on T not “w”
 - Disadvantages
 - What is the best way to treat surface interaction, convection and moist processes?

How does it work?

- Adiabatic (isentropic) trajectories
 - Adiabatic motion => isentropic coordinates

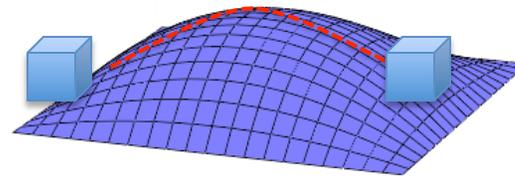
$$\frac{\theta}{T} = \left(\frac{p_o}{p} \right)^{\kappa}$$



How does it work?

- Adiabatic (isentropic) trajectories
 - Adiabatic motion => isentropic coordinates

$$\frac{\theta}{T} = \left(\frac{p_o}{p} \right)^\kappa$$



How does it work?

- Adiabatic (isentropic) trajectories
 - Adiabatic motion => isentropic coordinates

$$\frac{\theta}{T} = \left(\frac{p_o}{p} \right)^{\kappa}$$

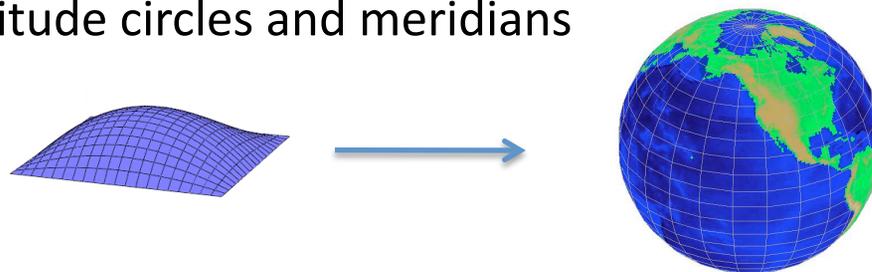
- Diabatic trajectories
 - parcels transported along isentropic surfaces by the large-scale wind field
 - parcels moved across isentropic surfaces by the net diabatic heating
- Kinematic trajectories
 - parcels transported horizontally on pressure surfaces
 - parcels moved vertically by the omega field (dp/dt , where p is the pressure)

How does it work operationally?

- Given met fields specified on a regular lat-lon grid and at regular time intervals. How do we use this to advect parcels?
 - Perform vertical interpolation of met fields from pressure surfaces to the Θ surface of interest => met data on an isentropic surface



- Interpolate isentropic surface to parcel position doing interpolation along latitude circles and meridians



- Finally interpolate met fields linearly in time to current model time

How does it work operationally?

- Massive numbers of trajectories give insight into the overall transport properties of meteorological fields (whether from a data assimilation system or a general circulation model).
- Schoeberl et al. used diabatic and kinematic trajectories to compare tropical transport produced by UKMO and GEOS data assimilation systems.

Schoeberl et al., “A comparison of the lower stratospheric age spectra derived from a general circulation model and two data assimilation system”, *J. Geophys. Res.*, 108, 2003

How does it work operationally?

- After determining winds, model moves parcels isentropically forward (or backward) in time by solving

$$\vec{r}(t_1 + \Delta t) = \vec{r}(t_1) + \int_{t_1}^{t_1 + \Delta t} \vec{v} dt$$

How does it work operationally?

- After determining winds, model moves parcels isentropically forward (or backward) in time by solving

$$\vec{r}(t_1 + \Delta t) = \vec{r}(t_1) + \int_{t_1}^{t_1 + \Delta t} \vec{v} dt$$

↑
integrate

4th-order Runge-Kutta Method

$$\begin{aligned}k_1 &= hf(x_n, y_n) \\k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\k_3 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\k_4 &= hf(x_n + h, y_n + k_3) \\y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)\end{aligned}\tag{16.1.3}$$

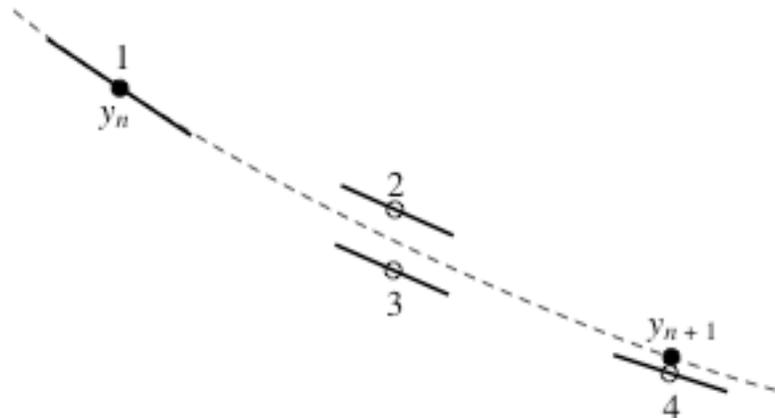


Figure 16.1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)

4th-order Runge-Kutta Method

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 &= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \end{aligned} \quad (16.1.3)$$

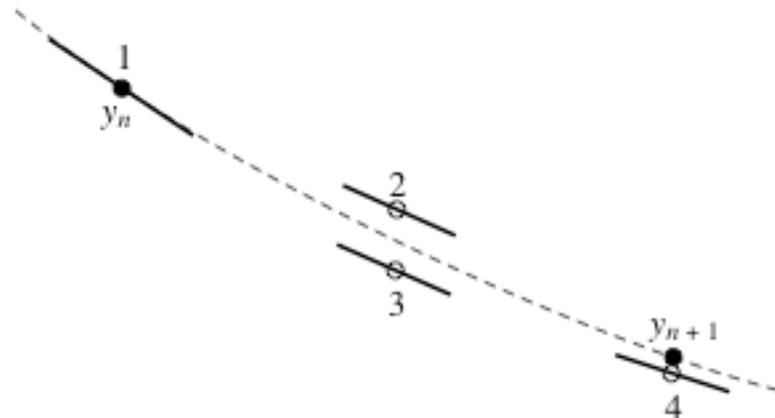
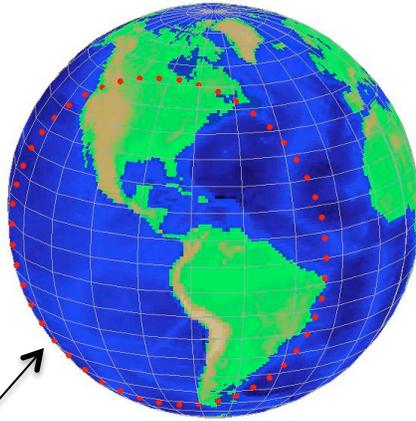


Figure 16.1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)

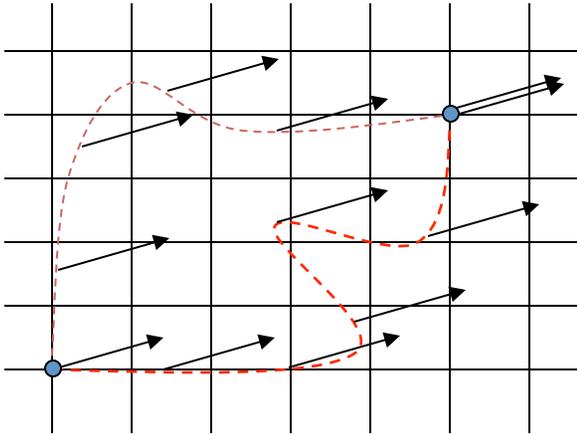
Computation of trajectories occurs on a sphere



geodesic

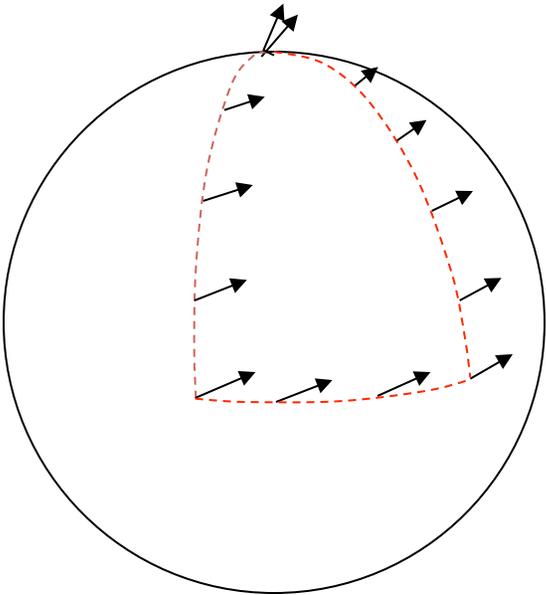
Curvature

In flat space:



parallel-transported
vectors are the same
regardless of the path

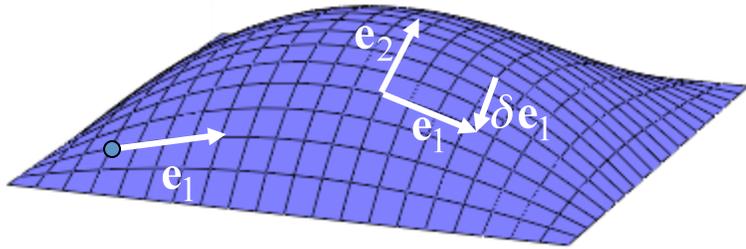
In curved space:



they are not

The Connection

How does a basis vector change when you move from one point to another?



$$\delta \mathbf{e}_i = \frac{\partial \mathbf{e}_i}{\partial x^j} \delta x^j \quad \frac{\partial \mathbf{e}_i}{\partial x^j} = \Gamma_{ij}^k \mathbf{e}_k$$

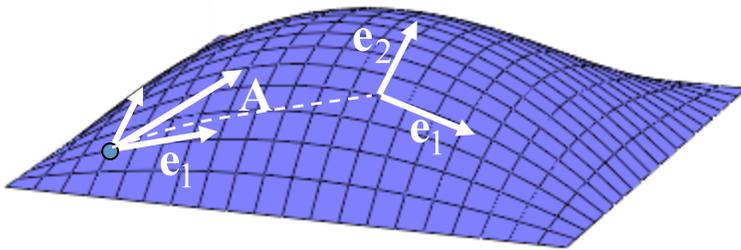
$$\Gamma_{ij}^k = \frac{1}{2} g^{kn} \left(\frac{\partial g_{in}}{\partial x^j} + \frac{\partial g_{jn}}{\partial x^i} - \frac{\partial g_{ij}}{\partial x^n} \right)$$

Thus, for any vector: $\mathbf{A} = A^i \mathbf{e}_i$

$$d\mathbf{A} = \frac{\partial \mathbf{A}}{\partial x^k} dx^k = \frac{\partial (A^i \mathbf{e}_i)}{\partial x^k} dx^k = \left(\frac{\partial A^i}{\partial x^k} \mathbf{e}_i + A^i \Gamma_{ik}^j \mathbf{e}_j \right) dx^k = \left(\frac{\partial A^i}{\partial x^k} + \Gamma_{jk}^i A^j \right) dx^k \mathbf{e}_i$$

Parallel Transport

Move a vector from one place to another without changing it



$$d\mathbf{A} = 0$$

$$d\mathbf{A} = \frac{\partial \mathbf{A}}{\partial x^k} dx^k = \left(\frac{\partial A^i}{\partial x^k} + \Gamma_{jk}^i A^j \right) dx^k \mathbf{e}_i = 0 \quad \longrightarrow \quad dA^i = -\Gamma_{jk}^i A^j dx^k$$

old

```
void RungeKutta :: solve( real t, real h, VectorArray *positions ) {
    real newTime;
    VectorArray*k1, *k2, *k3, *k4;
    real t_new;

    *F = field_object -> compute( t, positions, h );
    *k1 = (*F) * h;
    t_new = t + h * ((real) 0.5l);
    *pos_new = (*positions) + (*k1) * ((real) 0.5l);

    (*F) = field_object -> compute( t_new, pos_new, h );
    (*k2) = (*F) * h;
    t_new = t + h * ((real) 0.5l);
    *pos_new = (*positions) + (*k2) * ((real) 0.5l);

    (*F) = field_object -> compute( t_new, pos_new, h );
    *k3 = (*F) * h;
    t_new = t + h;
    *pos_new = (*positions) + (*k3);

    (*F) = field_object -> compute( t_new, pos_new, h );
    *k4 = (*F) * h;

    (*positions) = (*positions) + (*k1)*RECIPROCAL_6 + (*k2)*RECIPROCAL_3
        + (*k3)*RECIPROCAL_3 + (*k4)*RECIPROCAL_6;
}
```

new

```
void RungeKutta::integrate(real time, real dt, Position* location) {
    real newTime;
    Vector k1, k2, k3, k4;
    Vector *f = new Vector();

    F = flowField -> computeFlow( time, location );
    F.getElement(0,f);
    k1 = (*f) * dt;

    F = flowField -> computeFlow(time + dt/2, location, k1 * ((real) 0.5));
    F.getElement(0,f);
    k2 = (*f) * dt;

    F = flowField -> computeFlow(time + dt/2, location, k2 * ((real) 0.5));
    F.getElement(0,f);
    k3 = (*f) * dt;

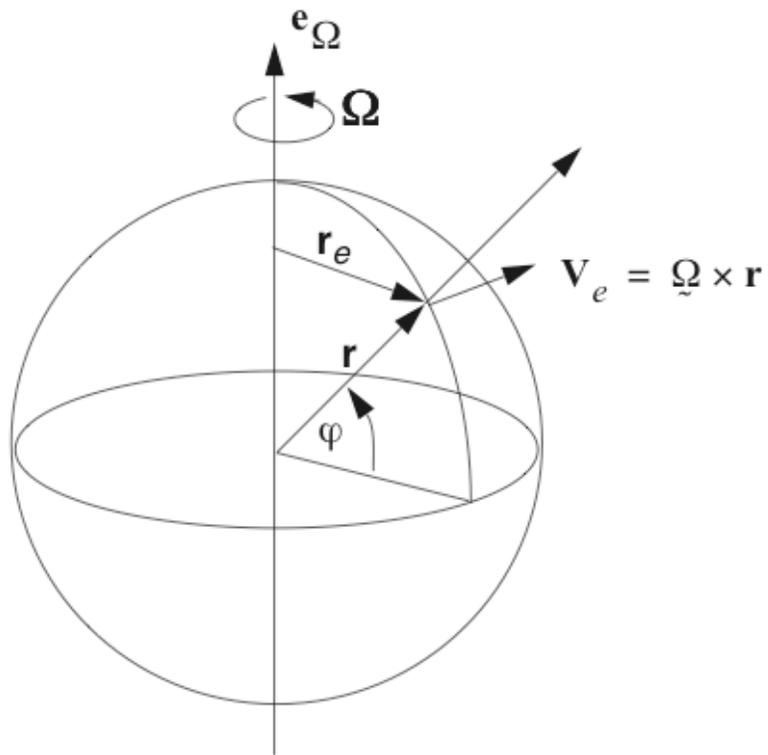
    F = flowField -> computeFlow(time + dt, location, k3 );
    F.getElement(0,f);
    k4 = (*f) * dt;

    VectorArray sumOfKValues =
        (k1 * k1Weight) +
        (k2 * k2Weight) +
        (k3 * k3Weight) +
        (k4 * k4Weight);
    location->move(sumOfKValues);
}
```

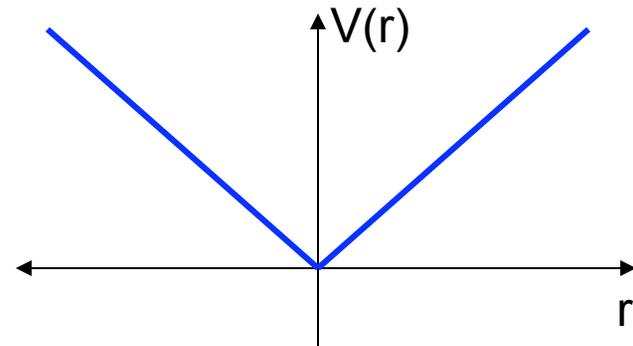
Tests

- Integration in 1,2,3D (in Cartesian coordinates)
- Spherical position
- Solid body rotation (SBR)
- SBR with $\Omega(t)$

Solid Body Rotation Tests



$$V(r) = \Omega r, \quad \Omega = \text{constant}$$



Demos...

Code can be checkout code and demos build from src/ directory:

```
>make demos
```

```
>cd apps
```

```
>demos.x
```

That will generate several text files that can be used to plot the test results.

The End