LES of a Cumulus Cloud with Explicit Lagrangian Droplets

I. S. Kang[1]*, Y. Noh[1], and S. Raasch[2]

[1]Department of Atmospheric Science, Yonsei University, Korea
[2] Institute of Meteorology and Climatology, University of Hannover, Germany
E-mail:ynoh@yonsei.ac.kr

Key Words: LES, cloud, suspended particles, particle settling, turbulence

ABSTRACT

LES model of a cumulus cloud is developed in which the motion of a large number of water droplets are explicitly simulated. Model provides various important information of the cloud process such as entrainment and internal mixing, the distribution of droplets within a cloud, and the evolution of particle size spectrum.

1. INTRODUCTION

Turbulence plays a critical role in cloud dynamics. It determines how dry air is entrained into a cloud and mixes within it, how droplets collide each other to increase the droplet size by coalescence, and how droplets are distributed within a cloud and ultimately precipitate.

Meanwhile, there have been few attempts to simulate a cloud by using LES, because of the difficulty in dealing with complex thermodynamics and microphysics relevant to cloud dynamics. In particular, there has been no LES model of a cloud with Lagrangian droplets so far.

2. MODEL

In the present study, we have developed a cumulus cloud LES model based on the PALM(Parallelized LES Model) whose parallelized code shows an excellent parallel performance up to a very large number of processors[1]. A new feature is included in the code to simulate a large number of Lagrangian droplets explicitly. Since it is not realistic to simulate all droplets in the cloud, we introduced the concept of weighting factor A_p which represents the ratio of real and simulated volumes of droplets, i.e.,

$$q_l = A_p \frac{1}{\rho}\left[\frac{\rho_L}{\Delta V}\sum_{i=1}^{N_p}\frac{4}{3}\pi r_i^3\right]$$

where N_p is the number of particles in a grid, q_l is the total liquid water content and ρ_L and r_i are the density and radius of a droplet. Each droplet experiences the increase of its radius by condensation and coalescence, while it is advected by the simulated turbulent field as

$$\frac{dV_i}{dt} = \frac{1}{\tau_p}(u_i - V_i - w_s\delta_{i3})$$

where V_i and u_i are the velocity of a droplet and fluid, respectively, τ_p is the inertial response time, and w_s is the settling velocity in the rest fluid. Here the process of condensation and coalescence are treated by the conventional approach of microphysics for a single droplet [2]. After coalescence and advection, A_p is modulated to conserve the total liquid water content.

The model domain was 6 x 1 x 6 km, and the grid size was 20 m in all directions. Experiments started with a two-dimensional thermal near the bottom with 2.4 x 10^4 droplets with r_i = 0.1μm. The relative humidity was given as 99% in the whole domain, and the total buoyancy was given by Q_0 = 0.3 m^3s^{-3}. We also simulated the case with a dry cloud without microphysics and the case with Q_0 = 2.4 m^3s^{-3} for comparison.

3. PARALLEL COMPUTATION

The most difficult part of the code development was the part to calculate the interaction of particles, which requires the particle concentration neighbouring each particle.

In the new algorithm, data of all particle features *i* (like the current particle coordinates or velocity components) are stored in 1d-arrays P_i dimensioned as $P_i(N)$ where N is the number of particles in use. At the initial stage of a model run, the particle data are sorted following the sequence of the grid point data stored in the memory. For example, $P_i(1:10)$ contains data of those particles located in grid box *(1,1,1)*, $P_i(11:20)$ those of particles in *(2,1,1)*, etc. That means that if e.g. the new particle coordinate *x* for the next time step is calculated in a loop like

```
DO  n = 1, number_of_particles
   Px(n) = Px(n) + delta_t * u_int
ENDDO
```

where `u_int` is the u-component of the velocity field of the fluid as interpolated from the neighbouring grid points, the u-velocity grid point data are accessed in the order as they are stored in memory. This is computationally very efficient because it takes advantage of the cache features of the processor avoiding time-consuming memory access.

During the simulation, due to the turbulent mixing, the correlation between the particles and the grid boxes gets lost. Although the data of the first particle are still stored in $P_i(1)$, the particle now probably stays in a box far away from that where it was initially started. Carrying out the above loop now requires a more or less random access to the memory in order to get the velocity data needed for interpolation. This causes a large number of cache misses which significantly slows down the computational speed, especially for large grids whose data does not fit into the cache.

The main idea of our optimization is a re-sorting of the particles after each time step in order to avoid the cache miss. The sorting is carried out in a way that the original correlation between the sequence of the particle data and the sequence the grid point data is stored in memory is restored. This method yields a great advantage for the calculation of particle concentration. If the particles are not stored in a consecutive order, we have to find for each particle the particles residing in its vicinity by carrying out a loop like

```
DO  n = 1, number_of_particles
   DO  k = 1, number_of_particles
      IF ( k /= n ) THEN
         IF ( ABS( Px(k) -Px(n) ) < threshold ) THEN ...
      ENDIF
   ENDDO
ENDDO
```

which requires an order of N^2 operations. If particles are sorted according to the grid boxes, however, the particle concentration within these (and the neighbouring) boxes can be calculated very fast using a loop like:

```
DO  n = n_start(i,j,k), n_end(i,j,k)

ENDDO
```

where `n_start` and `n_end` are the lower and upper index of those particles within gridbox *(i,j,k)*. This method only requires operations in the order of *N*.

Using this sorting method, the CPU-time needed for the particle motion was reduced to12% of the total integration time in the present simulation, from 74% in the case of an old algorithm.

4. RESULT

Figure 1 shows the evolution of distributions of potential temperature and droplets during the development of a cumulus cloud. Contrary to the typical vortex-pair shape of a dry cloud, a strong core of buoyancy maximum is maintained with stronger upward motion. Distribution of the variance of temperature reveals that entrainment occurs mostly at the top of a cloud initially, and subsequently occurs at the side as well at the later stage.

The entrainment of dry air into a cloud near the top increases the variability in the droplet spectra, and it also causes the droplet size there by enhanced collision rate. The evolution of the droplet spectra also shows the slow growth by condensation at the initial stage, followed by the rapid growth by coalescence, which is in good agreement with the observational evidence (Fig. 2). Experiments with larger Q_0 shows much larger enhancement of buoyancy by microphysics and much faster droplet size growth at the initial stage.

More realistic simulation of a cumulus cloud is expected in the future with more elaborate treatment of droplet collision process and initial droplet spectrum.

REFERENCES

[1] Raasch, S. and M. Schr? er (2001). A large eddy simulation model performing on massively parallel computers. Z. Meteor. 10, 363-372.

[2] Roger, R. R. and M. K. Yau (1976). 'A Short Course in Cloud Physics' (Elsevier).

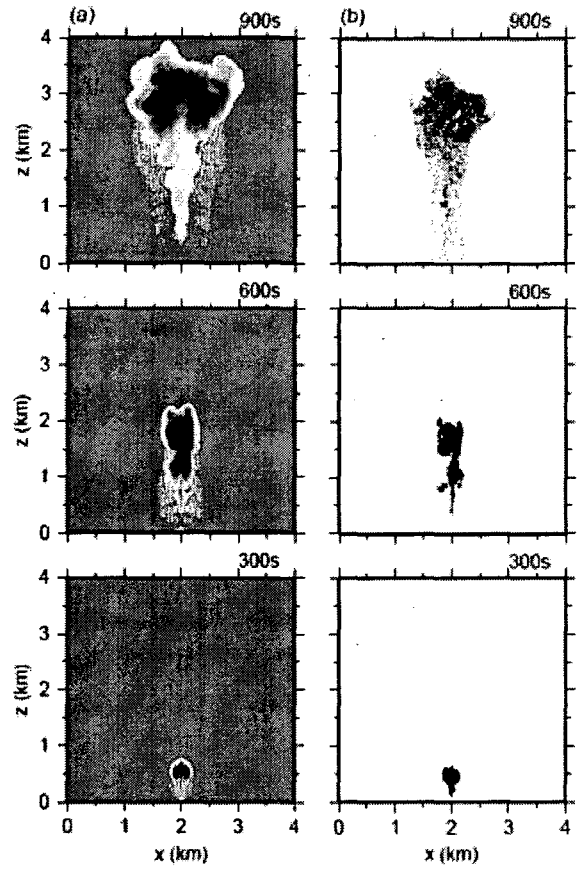

Fig. 1 Evolution of the distribution of (a) potential temperature and (b) droplets at the vertical cross-section.

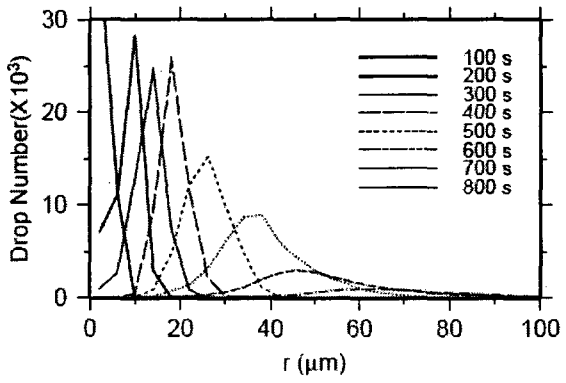

Fig. 2 Evolution of a droplet spectrum with time.