PERGAMON

Computers & Geosciences 28 (2002) 143–154

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

Practical considerations to speed up Lagrangian stochastic particle models

Stephan Schwere[a], Andreas Stohl[b], Mathias W. Rotach[a,*]

[a] *Institute of Climate Research, Swiss Federal Institute of Technology, Winterthurerstr 190, CH-8057 Zurich, Switzerland*
[b] *Technical University of Munich, Germany*

Received 30 November 1999; received in revised form 6 December 2000; accepted 10 December 2000

Abstract

The most serious drawback of Lagrangian stochastic particle models is their excessive demand of computing time. Even with today's powerful computers this makes them unattractive for many practical applications, such as for the assessment of air pollution in a long-term (e.g., yearly) scenario. This contribution therefore focuses on two issues: firstly, on which specific tasks do these models 'waste' computing time? Secondly, methods are presented in order specifically to reduce the computing time of these tasks. It is shown that for 'simple' models (e.g. those suited for Gaussian homogeneous turbulence) a speed-up factor of 10–100 can be reached when following these reduction strategies. The overall speed-up factor is about two for a more sophisticated model, which takes into account skewed and inhomogeneous turbulence. The results from the 'speeded-up simulations' are compared with the original model's results and it is shown that the speed-up procedures do not significantly alter the concentration patterns. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Lagrangian particle models; Computing time; Random numbers; Time step

1. Introduction

Lagrangian stochastic particle models are widely used and generally accepted as being the most powerful tool to model the dispersion of atmospheric pollutants in the boundary layer (Wilson and Sawford, 1996). Many models for different scaling regimes (e.g., Gaussian turbulence in a neutral boundary layer or skewed vertical turbulence in the convective boundary layer, e.g., Luhar and Britter (1989)) have been presented over the last decade and their quality has continually improved. However, their complexity has also gradually increased which leads to their major disadvantage: their enormous requirement of computing time (CPU-time). Although much work on the quality of the model results has been published, little work has been done so far on the aspect of the required CPU-time. In this respect, simplifications in the physics of the model have been proposed in order to save CPU time. For example, Hurley and Physick (1993) use a vertically homogeneous skewness of the vertical velocity component rather than the more time consuming actually observed profile. This allows using a larger time step. Ryall and Maryon (1998), as another example, allow their model a transition to a random displacement procedure once the particles are sufficiently far from the source. Ermak and Nasstrom (2000) introduced a higher order non-Gaussian method to solve the random displacement equation and showed that their method is accurate and significantly more efficient than a lower order, Gaussian method. In the present contribution we focus on analyzing and improving the efficiency of the numerical procedures involved in calculating the particle trajectories, rather than simplifying the underlying physics in order to speed-up Lagrangian particle models.

*Corresponding author. Tel.: +41-1-635-5222; fax: +41-1-362-5197.
E-mail address: rotach@geo.umnw.ethz.ch (M.W. Rotach).

0098-3004/02/$ - see front matter © 2002 Elsevier Science Ltd. All rights reserved.
PII: S0098-3004(01)00023-1

Although CPU-time is not prohibitive for scientific applications of Lagrangian particle models (such as simulating some 10 releases of a tracer experiment), in practical applications simple analytical models (e.g. Gaussian plume models) are still widely used due to CPU restrictions. It is for such tasks, when 8760 h (of a year, say) and releases from multiple sources are to be simulated that the present paper aims at contributing a step towards faster performance of the more reliable Lagrangian dispersion models.

In Lagrangian stochastic models, the trajectories of the simulated particles are numerically generated by calculating each particle's acceleration for each time step. In order to obtain statistically reliable results, as many as $O(10^5)$ particles must be traced (van Dop, 1992; de Haan, 1999). This large number of particles together with the often-required small time step leads to the excessive CPU-time consumption. de Haan (1999) showed that when using a kernel method to determine the concentration fields instead of the most popular box-counting method, the required number of simulated particles can be reduced by about two orders of magnitude. Hence the CPU-time is reduced by the same factor. In the present work, three additional methods to further reduce the CPU-time are presented. Whereas two of them deal with the reduction of required operations for determining the particle's acceleration, the third is related to the time step used for the simulation. It is evident that these methods should not alter the quality of the model results, namely the characteristics of the original model must be preserved. In Section 2, a brief overview of the theoretical background of Lagrangian stochastic particle models together with the three example models used in this work is given. The three methods employed for speeding up the models are presented in Section 3. The results in terms of CPU-time savings and of the qualitative effects for the models are given in Section 4.

2. Theory of Lagrangian stochastic particle models

Lagrangian stochastic particle dispersion models are based on the assumption that the evolution of particle velocities and positions can be described as a Markov process. Thus, using the approach of Thomson (1987), the evolution of the velocity fluctuations u_i and the particle position x_i is governed by the stochastic differential equations

$$\begin{aligned} \mathrm{d}u_i &= a_i(\mathbf{x},\mathbf{u},t)\,\mathrm{d}t + b_{ij}(\mathbf{x},\mathbf{u},t)\,\mathrm{d}\xi_j,\\ \mathrm{d}x_i &= u_i\,\mathrm{d}t, \end{aligned} \tag{1}$$

where $\mathrm{d}t$ is the time step, $\mathrm{d}\xi_j$ are the increments of a Wiener process with zero mean and variance $\mathrm{d}t$. The functions a_i and b_{ij} are, in general, functions of the particle position in space ($\mathbf{x}$), the particle velocity components $\mathbf{u}(=(u_1,u_2,u_3))$ and time (t). A sample of trajectories can then numerically be calculated by replacing the infinitesimal increments $\mathrm{d}\mathbf{u}$, $\mathrm{d}\mathbf{x}$, $\mathrm{d}t$ and $\mathrm{d}\xi$ through the finite increments $\Delta\mathbf{u}$, $\Delta\mathbf{x}$, Δt and $\Delta\xi$.

To design a model for a specific situation the functions a_i and b_{ij} need to be specified. There are two widely used expressions for b_{ij}. Firstly, b_{ij} can be determined by using the velocity structure function for the inertial subrange (e.g., Sawford and Guest, 1988). It then can be shown that—if the frequency $(\mathrm{d}t)^{-1}$ falls into the inertial subrange—the following expression is valid:

$$b_{ij} = \tfrac{1}{2}\sqrt{C_0}\varepsilon\delta_{ij} \tag{2}$$

with C_0 being an universal constant, ε the ensemble-average rate of dissipation of turbulent kinetic energy and δ_{ij} the Kronecker delta function. Often, b_{ij} is alternatively determined using the Lagrangian integral time scale, $T_{L,i}$:

$$b_{ij} = \sqrt{\frac{2\sigma_i^2}{T_{L,i}}}, \tag{3}$$

where σ_i is the standard deviation of the corresponding velocity fluctuations. This expression is, strictly speaking, only valid for stationary and homogeneous Gaussian turbulence.

Thomson (1987) shows that the functions a_i must obey certain restrictions of which the most general is the 'well-mixed' condition. This condition can mathematically be expressed through the Fokker–Planck equation for process (1) with $P = P_a$:

$$\frac{\partial P}{\partial t} = -\frac{\partial}{\partial x_i}(u_i P) - \frac{\partial}{\partial u_i}(a_i P) + \frac{\partial^2}{\partial u_i \partial x_i}(B_{ij} P), \tag{4}$$

where P stands for the probability density function (pdf) of the particle velocities and P_a is that of the fluid elements. Here, B_{ij} is simply an abbreviation for $\frac{1}{2}b_{ik}b_{jk}$. Here, and in the following, small letters are used to denote the fluctuating wind speed components $U_i = \bar{u}_i + u_i'$, with U_i referring to the actual wind speed component and the overbar to the ensemble average. To develop a model for a specific situation, it is sufficient to specify an appropriate pdf, $P\,(=P_a)$ for this flow regime and then solve Eq. (4) for a_i.

Three different models are used to demonstrate the effects of the speed-up procedures presented in Section 3 (Table 1). Two of them are intended to be used for short-range dispersion problems, whereas the third is a long-range model. The model of Thomson (1987), henceforth denoted T87, is based on a Gaussian inhomogeneous pdf. In the present study, however, only the vertical variability is taken into account. It is quadratic in velocity and termed (by its author) the 'simplest' solution to Eq. (4) for this pdf and hence, it is

Table 1
Models used to demonstrate speed-up procedures presented in Section 3

			pdf	Reference(s)
T87	Short-range	Horizontally homogeneous[a], stationary	Gaussian	Thomson (1987)
R96	Short-range	Horizontally homogeneous, stationary	Continuous transition between Gaussian correlated (u and w) pdf to skewed pdf in w, uncorrelated to u	Rotach et al. (1996), de Haan and Rotach (1998)
S98	Long-range	Horizontally inhomogeneous, non-stationary	Gaussian	Stohl et al. (1998)

[a] Original model of Thomson (1987) uses Gaussian, inhomogeneous pdf. In this study, indicated restrictions are applied (basically due to lack of input data in example simulations presented).

optimally suited for the first speed-up procedure as introduced in Section 3. The model of Rotach et al. (1996), which will be termed R96 in the following, is two-dimensional and horizontally homogeneous. It is based on a pdf which continuously varies (as a function of stability) between 'Gaussian, correlated (u and w)' to 'skewed in w, uncorrelated (u and w)'. In the present study the three-dimensional version of this model, as described in de Haan and Rotach (1998), is used. The model has been tested successfully against tracer data (Rotach et al., 1996). The long-range model FLEXPART (of which the user's guide can be found in http://www.forst.uni-muenchen.de/LST/METEOR/stohl/flexpart.html) is based on data from the numerical weather prediction model of the European Center for Medium-Range Weather Forecasts (ECMWF, 1995). It will be termed S98 in the following. Like the short-range models, it is governed by the Langevin Equation (1). The discrete model formulation follows Wilson et al. (1983), with a modification by Stohl and Thomson (1999) to account for variations in air density. The three wind components evolve independently, apart from linkages caused by the fact that the coefficients in the equations depend on particle position. This involves neglecting the cross-correlations which have only minor effects on the results of large-scale dispersion models (Uliasz, 1994). Gaussian turbulence is assumed under all meteorological conditions, and the turbulent statistics are obtained using the scheme of Hanna (1982) with some modifications for convective conditions. An important difference to the short-range models is that the turbulence is (also horizontally) inhomogeneous and non-stationary. The model accounts for low-frequency wind variations ('meandering') by solving a second Langevin equation, assuming that meandering and turbulence can be treated independently. The FLEXPART model (S98) was evaluated with data from three large-scale tracer experiments comprising a total of 40 releases (Stohl et al., 1998). The first release of the European Tracer Experiment (ETEX-I) is reconsidered here for our purposes (for a description of the experiment see the Atmospheric Environment special issue on ETEX (1998, volume 32). In this exercise FLEXPART was run for a simulation time of $\sim$90 h, releasing 50,000 particles during the first $\sim$12 h.

3. Methods

The simulation with a particle dispersion model can roughly be split into three major parts: the calculation of the particle's acceleration a_i, the generation of the random numbers $d\xi$ and the evaluation of a time-step criterion. All of these three parts have to be completed for each particle and each time step. In this section, three methods are presented to reduce the required CPU-time for each of these parts. In Section 4, additional conditions or constraints will be presented for these methods, which must be followed in order to ensure that the results of the dispersion simulation are not affected by employing them.

3.1. Calculation of acceleration a_i

The expression for calculating the acceleration a_i in Eq. (1) can be written as a sum of products of two different types of functions: the first one is only a function of position $\mathbf{x}$ and the second one only a function of the actual particle's velocity $\mathbf{u}$. Hence, for the horizontally homogeneous models (T87, R96) considered in the present study, a_i is of the form

$$a_i = \sum_j f_{ij}(z) g_{ij}(\mathbf{u}), \tag{5}$$

where the index i refers to the three coordinate axes and j corresponds to the number of required summands in Eq. (5). If the functions $f_{ij}(z)$ are determined prior to the simulation once for all, the required CPU-time for each

particle at any one time will drastically be reduced. The derivation of the functions f_{ij} and g_{ij} for the models under consideration and hence the factorization of a_i is straightforward. For model T87, they are explicitly given in Rotach and Schwere (2000). Due to the more complicated form of a_i some additional terms appear in Eq. (5) for the model R96 and the functions g_{ij} become more complex, i.e. it is impossible to split completely the expression for a_i into the two above-mentioned kinds of functions. Therefore, all the mixed terms need to be incorporated in the functions g_{ij}.

The advantage of this factorizing of the functions a_i becomes evident when bearing in mind that in the present horizontally homogeneous situation the functions f_{ij} are only a function of the particle height. Due to this property, they can be evaluated prior to the actual simulation once for all leading to an enormous reduction in required CPU-time for each particle at each time step. Here, the functions f_{ij} are calculated in intervals of $\Delta z = 1$ m from the surface up to the upper end of the boundary layer at $z = z_i$ and stored. This method is chosen for simplicity: when determining the acceleration a_i during the simulation, the actual particle height leads immediately to the appropriate index of the arrays in which the f_{ij} are stored. However, the f_{ij} could generally be determined to any desired spatial discretization. The net effect of this method is that only a few operations to calculate a_i (mainly some multiplications and additions) remain to be executed during the simulation.

For the long-range model (S98), a factorization of the a_i would be much less efficient, since the particles are usually distributed rather inhomogeneously over the grid. Factorizing the a_i for the whole grid would mean attributing a large amount of computation time to grid cells where no particles reside. Furthermore, the factorized a_i would occupy a large amount of storage space, higher by a factor $2*nx*ny$ than that required for the short-range models (nx and ny being the number of grid points in the horizontal directions). The factor 2 results from non-stationarity, which requires two wind fields to be kept in memory at a time for interpolation. A possible solution would be to factorize the a_i only for those grid cells where particles actually reside, but this requires extra computation time. As shown later there are other possibilities for speeding up long-range models. Hence, factorization was not taken into account within the model S98.

3.2. Random numbers

For each particle and each time step, three random derivatives $\mathrm{d}\xi_j$ in Eq. (1)—one for each of the three coordinate directions—must be generated. This can easily be done by using, for example, a standard routine (e.g. *ran2* and *gasdev*) from Press et al. (1992). It is important to use a qualitatively 'good' random number generator (e.g., van Dop, 1992).

Instead of generating the random numbers anew for each time step and particle, a 'pool' of random numbers with the required properties can be calculated prior to the actual simulation and stored in an array. Now, during the simulation, the first three random numbers (one for each of the three coordinate directions) for a particle i can be taken from this pool by randomly picking three indices from this array. The next set of three random numbers for the same particle i is then obtained by simply increasing all of the three indices by 1. This procedure is repeated for up to 400 time steps, before starting a new sequence by randomly picking three initial indices. The smaller the ratio between the size of the pool and the actually required random numbers for the original models, the bigger is the saving of CPU-time.

3.3. Time step

An actual simulation is carried out by replacing the inifinitesimal quantities in Eq. (1) by finite differences. In order to get physically realistic particle trajectories, some constraints concerning the time step Δt need to be taken into account (see e.g. time-step criteria in Thomson, 1987 or Rotach et al., 1996). In this investigation, the following time-step criterion is used for the short-range models:

$$\Delta t_{\max} = \min\left(\frac{0.01\varepsilon}{\left|w\partial\varepsilon/\partial z\right|}, \frac{0.1\sigma_w}{|a_w|}, \frac{0.1\sigma_u}{|a_u|}, \frac{0.01\sigma_u}{\left|w\partial\sigma_u/\partial z\right|}, \frac{0.01\sigma_w}{\left|w\partial\sigma_w/\partial z\right|}, \left|\frac{u}{w\partial\bar{u}/\partial z}\right|, \frac{0.01\sigma_w^2}{B}\right), \tag{6}$$

where the subscript 'i' denotes the respective component, which is to be treated. In the following, the individual terms in Eq. (6) are referred to as criteria 1–7, respectively in the same order as they appear in Eq. (6). Criteria 1–7, again, are a function of height z but also of the fluctuating velocity components. Fig. 1 shows them as a function of z/z_i as well as for different combinations of u and w for model T87. Only very small time steps, such as $\Delta t = 0.01$ s, are allowed near the surface, whereas for regions higher up in the boundary layer larger time steps would be sufficient. To meet all the criteria, a very small time step needs to be prescribed, which leads to an increase in CPU-time for a simulation.

Instead of using a constant time step Δt, which satisfies all the criteria at any possible position and for all likely values of fluctuating velocities, a variable time step Δt^* can be used. In practice, this means for our approach that a default time step Δt^{def} is chosen which is considerably larger than the maximum allowed as suggested by an analysis as exemplified in Fig. 1. For

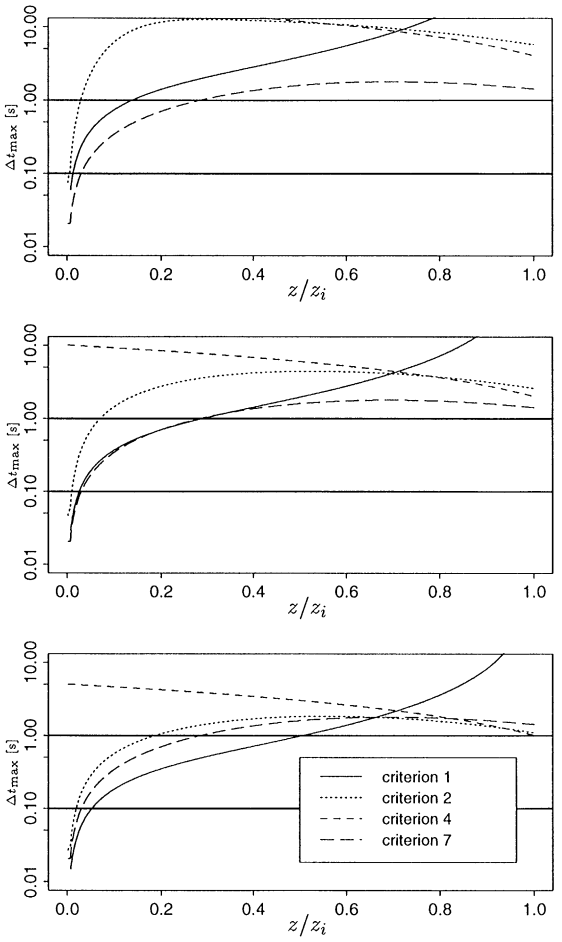

Fig. 1. Analysis of time step criteria of Eq. (6) for one (namely July 6, 1979) of Copenhagen tracer experiments (Gryning, Lyck, 1984). Criteria 3, 5 and 6 are not shown in this figure because they never determine Δt_{max}. Criteria are plotted for different values of vertical velocity: $w = 1\,\mathrm{m/s}$ (upper panel), $w = 2\,\mathrm{m/s}$ (middle panel) and $w = 4\,\mathrm{m/s}$ (lower panel). Since Δt_{max} proved not to be sensitive to u, a typical value of $u = 4\,\mathrm{m/s}$ was chosen. Two thick horizontal lines mark time steps of 0.1 and 1.0 s, respectively, as mentioned in text and in Table 2.

each particle and each time step it then should be checked whether Δt^{def} is not too large and thus violates one or more of the criteria of Eq. (6). If Δt^{def} is too large for a particle i, it is divided into smaller parts which fulfill the time step criteria (see Rotach et al., 1996 for details). Therefore, there is a trade-off between gaining computing speed by selecting a large Δt^{def} and the CPU-time spent on checking the criteria.

Fig. 2 ('original version' of the models) shows that it is rather 'expensive' in terms of CPU-time to check all these criteria. A closer analysis of these criteria shows that for a given Δt^{def} only some of them are relevant to determine Δt_{max}, i.e., only those criteria need to be evaluated (see Fig. 1). Moreover, the following properties can be observed from Fig. 1:

- Close to the surface, those criteria involving the dissipation of turbulent kinetic energy ε (criteria 1 and 7) are the most stringent.
- Criterion 1 dominates over criterion 7 for large values of w.
- For regions close to the mixing layer height z_i and large values of w, criteria 2 and 4 determine Δt_{max}.

Furthermore, it is found that Δt_{max} is not sensitive to u (not shown). If Eq. (3) instead of Eq. (2) is used to determine b_{ij}, those criteria involving ε (criteria 1 and 7) need not be taken into account. This leads to a substantial increase in Δt_{max}. However, this affects the results of the models as will be shown in Section 4.

A graphical analysis as presented in Fig. 1 can be used to reduce the full time step criterion (Eq. (2)) to the relevant criteria. The resulting time-step criterion used for an actual simulation (for the same meteorological situation as shown in Fig. 1) is given in Table 2. It shows, for example, that—if b_{ij} is given through Eq. (2) and for a chosen default time step $\Delta t^{def} = 0.1$ s—none of the criteria has to be evaluated for $z \geqslant 0.09 z_i$ and only for $z < 0.09 z_i$ criteria 1 and 7 need to be considered. In summary, the procedure to reduce the CPU-time in the evaluation of allowed time steps includes two stages: the selection of an appropriate default time step, Δt^{def}, and performing a graphical analysis for the meteorological situation under consideration resulting in the specification of those criteria, which need to be checked at certain positions in space.

Due to a different model architecture in the long-range model, S98, somewhat different criteria are used to determine the time step for the vertical component of the wind:

$$\Delta t_i = 0.01 \min\left(T_{L,w}, \frac{z_i}{2w}, \frac{0.5}{\partial \sigma_w / \partial z}\right). \quad (7)$$

Longer time steps for the horizontal wind components may be chosen, here by a factor of 4, to save computation time. Further, the time-step criteria are checked only every fourth time step. It is clear that with the criteria of Eq. (7), not much CPU-time can be saved by invoking a procedure as described here. Therefore, the effect of choosing (massively larger) fixed time step as an alternative is considered. Here, we use 15 min, which significantly exceeds the Lagrangian time scales for practically all situations, reducing the model to a random displacement one, where the turbulent particle velocities are no longer auto-correlated.

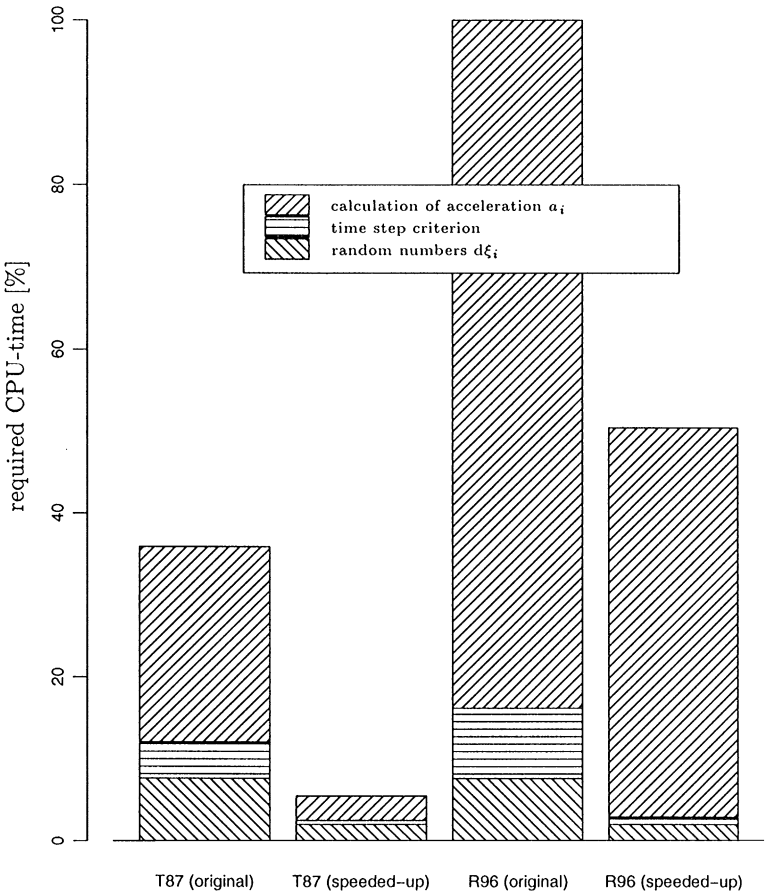

Fig. 2. Required CPU-time for calculation of acceleration a_i, generation of random increments $d\xi_j$ and evaluation of time step criterion. These three parts are shown for original version of short-range models under consideration and for respective speeded-up versions. All simulations have been performed with time step of 0.1 s. Speed-up resulting by introducing larger time step is not shown in figure.

3.4. Tasks specific for the long-range model S98

3.4.1. Preparation of the meteorological data fields and interpolation

As input, the long-range model needs five three-dimensional and 22 two-dimensional data fields. Additional fields of frequently used variables (e.g. density) are derived from these data, and all data are transformed to a terrain-following coordinate system. In our experiments, the dimensions of meteorological fields are 101×56 grid points and 28 layers.

The large-scale model applies to horizontally inhomogeneous conditions. Thus, to determine meteorological parameters at a particle position, interpolation between grid point values is necessary. Although high-order interpolation procedures are more accurate (Stohl et al., 1995), they also consume more computation time than lower-order methods. As a compromise, tri-linear space and linear time interpolation is used in S98. To avoid repeated interpolation of (almost) the same values, interpolation is done only every few minutes (here 15), except for the vertical interpolation of the grid-scale wind, which must be done at every time step because of the fast vertical fluctuations of the particles. The actual choice of the time steps for the simulations in the present paper depend on the numerical experiment considered. Various assumptions will be tested as outlined in Section 4.2.

3.4.2. Particle counting

In the long-range model, tracer densities are evaluated on a three-dimensional grid using a uniform square kernel, which reduces the number of particles needed to obtain stable results by a factor of two as compared to simple particle counting, without affecting the spatial resolution of the output. The CPU-time needed for this task depends on the number of particles and the interval at which concentrations are to be determined. Here, concentrations are calculated every 15 min (averaged to

Table 2
Time-step criterion used for actual simulation for same tracer experiment as in Fig. 1[a]

b_{ij}	Δt^{def} (s)	Conditions of z, w	Evaluated time step criteria	Required CPU-time (%)
Eq. (2)	0.1	$z \geqslant 0.09 z_i$	—	8%
		$z < 0.09 z_i$	1, 7	
		$z \geqslant 0.3 z_i$ and $\lvert w \rvert \leqslant 2$	—	13%
Eq. (2)	1.0	$z < 0.3 z_i$ and $\lvert w \rvert \leqslant 2$	7	
		$\lvert w \rvert > 2$	1, 2, 4	
		$z \geqslant 0.05 z_i$ and $\lvert w \rvert \leqslant 2$	—	10%

[a] z and w are, respectively, actual height and velocity of particle under consideration, z_i is boundary layer height. Time-step criteria are numbered in same order as they appear in Eq. (6). Last column shows required CPU-time in percent of CPU-time when all criteria of Eq. (6) are checked.

3-h average concentrations), yielding a total of 360 concentration calculations.

4. Results

4.1. The short-range models

All the simulations have been performed using the Copenhagen data set (Gryning and Lyck, 1984) in order to use realistic meteorological input data. For the qualitative evaluation, the crosswind integrated ground level concentration (CIC), the arcwise concentration maximum (ARCMAX) and the lateral spread (SIGMA Y) are compared. A statistical analysis of the speeded-up models compared to the original models will be performed by using the *normalized mean squared error* (NMSE), the *fractional bias* (FB) and the *correlation* (CORR) as defined, e.g. in de Haan and Rotach (1998).

The most efficient method to reduce the required CPU-time in the short-range models is that of factorizing the expression for the acceleration a_i. A reduction of almost 90% can be achieved for the model T87 whereas, as expected, the reduction for the model R96 amounts to only about 50% (Fig. 2). However, even for this latter complex model the factorizing proves to be an effective method. The time needed for the calculation of the function f_{ij} ('once for all', see Section 3.1) can be neglected when simulating more than 100 time steps. In Table 3 (and Fig. 3 as one example) the simulation results of the original models are compared to those of a respective version, when only factorizing is applied. Note that we compare here 'model vs. model' rather than 'model vs. measurements' in order to show that the factorizing does not alter the model predictions (how accurate they ever may be when compared to measurements). Both Fig. 3 and the statistical data of Table 3 show that factorizing does not diminish the qualitative aspects of the models under consideration. The only observed differences occur very close to the source (Fig. 3), although this can probably be attributed to the relatively small number of particles at this distance rather than to the applied speed-up procedure.

Table 3
Statistical comparison for speeded-up models (only by factorizing expression for a_i) and appropriate original model T87 or R96, respectively. Simulations have been performed for all 9 tracer experiments from Copenhagen experiment (see text). Number of data points, $n = 23$

	Model	NMSE	FB	COR
CIC	T87	0.002	0.001	0.994
	R96	0.007	0.009	0.989
ARCMAX	T87	0.024	0.018	0.989
	R96	0.080	0.014	0.978
SIGMA Y	T87	0.001	−0.004	0.993
	R96	0.001	0.012	0.994

The use of a variable time step allows the chosen Δt^{def} to be increased by at least a factor of 100, e.g. from 0.01 to 1 s. Hence, the required number of simulated time steps for a fixed simulation time will be reduced by the same factor. However, the CPU-time cannot be decreased by the same amount because there is some time spent on checking the time-step criteria (6) and therefore a shorter time step than Δt^{def} will be attributed to certain particles at a given instant. Table 2 shows that the time spent only on checking the time-step criteria (6) can be reduced by up to 90% when performing a graphical analysis to determine the relevant criteria, as shown in Fig. 1. Overall, a simulation with the model T87 using a Δt^{def} of 1.0 s and the appropriate time-step criterion requires approximately 3% of the CPU-time needed for the same simulation with $\Delta t^{def} = 0.01$ s. The introduction of a variable time step and the speeded-up time-step criterion does not affect the output of the models. However, a substantial difference can be observed when using Eq. (3) instead of Eq. (2) for the

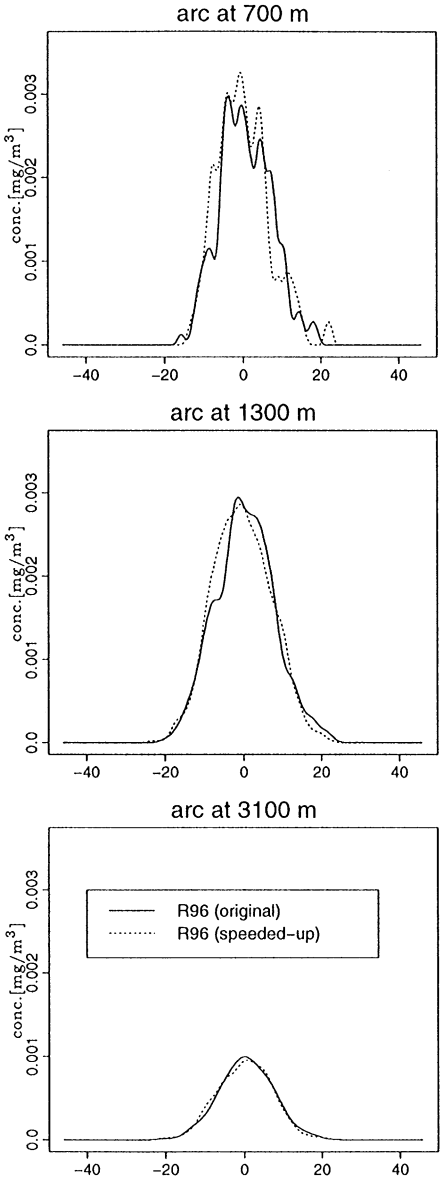

Fig. 3. Lateral concentration distribution (mg/m^3) at various distances downwind from source for one of Copenhagen tracer experiments (April 30, 1979) for model R96. Some differences can be observed close to source (arc at 700 m).

calculation of b_{ij}. This is due to the fact that Eq. (3) is only valid for homogeneous conditions with $T_{L,w}$ being constant (Thomson, 1987), although the simulations presented in this section are performed for vertically inhomogeneous conditions.

The size of the pool of random derivates $d\xi_j$ has an effect on the statistical output of the models. A systematic analysis of the pool size showed that a ratio of at least 1/1000 (size of pool divided by required number of random derivates for original model) proved

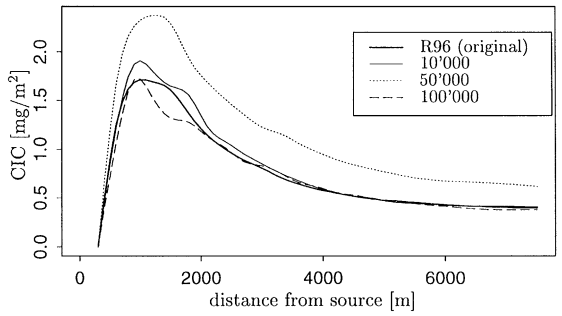

Fig. 4. CIC (mg/m^2) as function of distance from source for one of Copenhagen tracer experiments (September 16, 1978). Solid line represents simulation for original model R96 with 10,000 time steps and 5000 particles, and hence, 1.5×10^8 random numbers are needed for original model. Three lines labeled 10,000, 50,000 and 100,000 refer to speeded-up models with respective size of pool for random numbers.

to produce acceptable results. For example, in a simulation with 5000 particles and 10,000 time steps, 1.5×10^8 random numbers are needed for the original model and, hence, the size of the pool has to be at least 1.5×10^5. With this size, accurate results are obtained (Fig. 4). Over all, the CPU-time spent on dealing with the random derivates $d\xi_j$ can be reduced by a factor of 3–4 due to loosing time on counting/storing the indices and randomly picking new initial indices.

4.2. The long-range model

Four model experiments were carried out to study the effect of two speed-up measures on the model results:

(A) time steps determined by Eq. (7), random numbers generated for every time step and for each parameter separately,
(B) time steps determined by Eq. (7), sequences of random numbers taken from a pool of 2.0×10^6 numbers, a new sequence of random numbers being initialized every 15 min by randomly picking an index of the array representing the pool.
(C) as (A), but for constant 15 min time steps,
(D) as (B), but for constant 15 min time steps.

For the purpose of obtaining realistic input conditions, and also in order to compare the results to measurements, the simulations are performed with the conditions of the ETEX release (see Section 2). For a statistical comparison of the model results, three parameters were used:

The Figure of Merit in Space FMS $= 100(A_p \cap A_m)/(A_p \cup A_m)$, where A_p and A_m are the predicted and measured subsets of concentrations above a significant

Table 4
CPU-times, normalized to that of experiment (D), for four model experiments, and fraction (%) of CPU-time used[a]

Experiment	CPU	MF	CC	RN	INT	TS	PARA	LANG	REST
(A)	107.5	<1	<1	45	2	2	20	6	24
(B)	54.3	<1	<1	<1	2	4	38	12	44
(C)	1.2	21	8	23	32	0	<1	2	14
(D)	1.0	26	10	2	29	0	1	2	20

[a] MF = preparation of meteorological fields, CC = concentration calculation, RN = random number generation, INT = interpolation, TS = time step criteria, PARA = parameterization of turbulent statistics, LANG = Langevin equations, REST = various tasks (e.g. particle reflection, loops, if statements, etc.).

level (0.1 ng/m^3), the Pearson correlation coefficient, r, between the individual grid cell values and the fractional bias, FB.

A comparison of CPU-time spent on different 'tasks' during the four simulations is given in Table 4. Due to the different modes and typical applications the short-term models of Section 4.1 and the long-term model have somewhat different 'tasks', in which the CPU-time consumption is split up. However, the 'calculation of acceleration a_i' from Fig. 2 roughly corresponds to the sum of the entries MF + PARA + LANG in Table 4. In experiment (A) the generation of random numbers consumed almost half of the CPU-time, whereas this made a negligible contribution in experiment (B). Here, the parameterization of the turbulent statistics and the solution of the Langevin equation determined the CPU-time. Also important were the time-step criteria, the reflection algorithm, and code segments that could not be attributed to any specific task, whereas the CPU-times needed for the preparation of the wind fields and the concentration calculations were negligible. Visual inspection of the tracer concentration fields showed no obvious differences between the tracer patters produced by model runs (A) and (B), except for a small bias of model run B towards lower values (Table 5).

A significant reduction of the CPU-time (by a factor 100 and 50 compared to experiments (A) and (B)) was achieved by using constant time steps of 15 min (experiment (C)). The CPU-time loading for these experiments was completely different from that of the previous experiments (Table 4). Much time was spent for the preparation of the meteorological fields, the concentration calculation and the interpolation, whereas the parameterization of the turbulent statistics was negligible. Drawing from a pool of random numbers (experiment D) instead of generating them each time step, reduced the CPU-time by a further 20%, without significantly affecting the model results (Table 5).

Using constant 15-min time steps introduced a negative FB of 16% as compared to model run A. The bias was largest during the first 12 h (during the release), when vertical dispersion was unrealistically high in experiments (C) and (D). Furthermore, the density correction with these long time steps was not successful, and thus surface concentrations were lower for experiments (C) and (D) throughout the whole modeling period. However, a visual comparison of the model results after ~48 h (Fig. 5) shows that the differences after the initial phase were rather small. In view of the large CPU-time savings, these differences seem to be acceptable for many practical applications. In a statistical comparison with the tracer measurements (not shown), model run C even performed slightly better than model run A for many parameters.

Table 5
Global statistical comparison of different model runs with long-range model

Comparison	FMS (%)	r	FB
(A) and (B)	96.9	0.999	0.016
(A) and (C)	89.2	0.966	0.165
(C) and (D)	96.8	0.999	0.003

5. Discussion and conclusions

As an overall observation we find that in the 'original models' the computation of the accelerations (i.e., the function a_i in Eq. (1)) takes, in general, the largest part of the computing time for a Lagrangian stochastic particle model. As the second most expensive task, we can identify the generation of (high quality) random numbers, and time-step criteria rank as the third task. The computation of the a_i thereby uses about 25% of the total CPU-time for the simplest of the models investigated here (S98: Gaussian turbulence, no correlation between the velocity components). For a more sophisticated model (Gaussian inhomogeneous turbulence, T87) this is already about 50%, whereas the most sophisticated model investigated (R96), which takes into

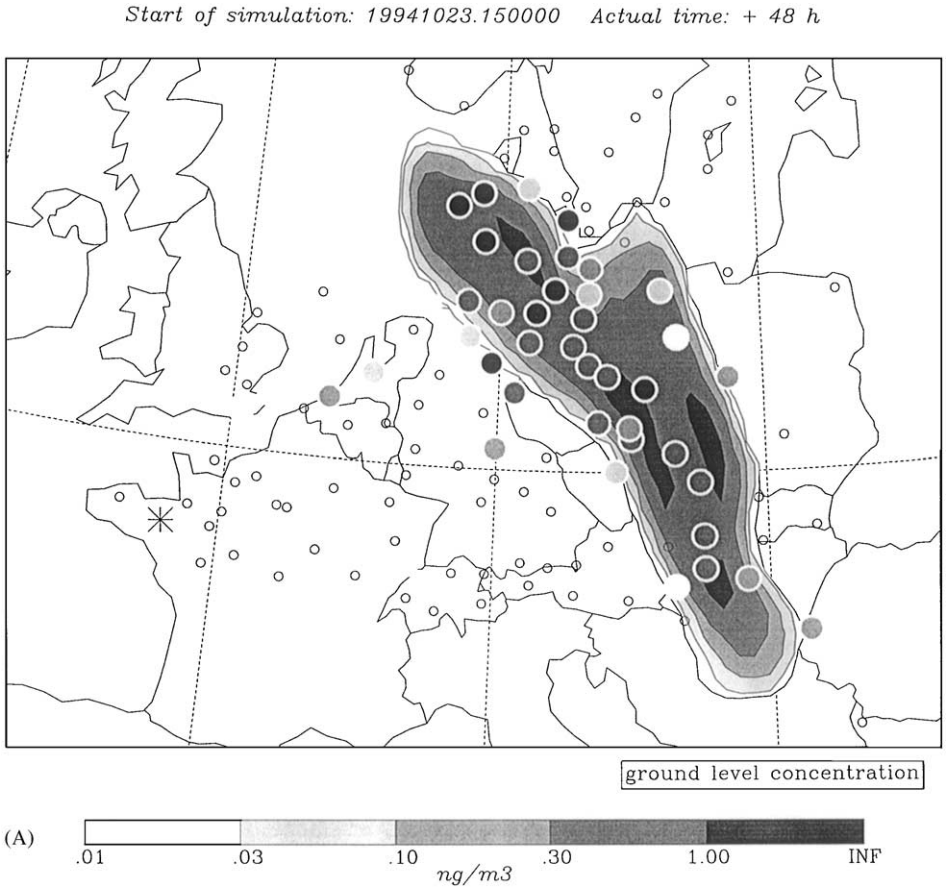

Fig. 5. Comparison of model results for numerical experiments A (A) and C (B) 48 h after start of release with tracer measurements. Empty circles denote measurements of zero concentration, whereas filled circles are coded with shading representing model results. Release location is marked with asterisk.

account non-Gaussian inhomogeneous turbulence, spends more than 80% on this task. Therefore, the proposed method of factorizing the functions a_i into contributions which are only dependent on the position in space of the particle and others which are dependent on their actual speed at every time step, proves to be simple and effective in reducing the required CPU-time.

Dispersion models, which use three-dimensional meteorological fields rather than parameterized flow and turbulence profiles as input, will spend a considerable part of their CPU-time on the interpolation of the meteorological data. In our study this is the situation for the long-range transport model (S98), but any Lagrangian particle model can be run in this mode, also in short- or meso-scale applications. For models of this type, factorizing is not an attractive reduction strategy, as outlined in detail in Section 3.1. As an alternative, we suggest the use of a (massively) increased time step (as compared to standard practice). More specifically, the best compromise between CPU-time consumption and accuracy of the results may be reached when using time steps limited by the Lagrangian time scales close to the source, and long time steps, exceeding the Lagrangian time scales, at larger distances from the source.

Once the major time-consuming task is resolved, it is suggested that the number of actual calculations of random numbers be reduced by introducing a 'pool', from which the random number can be picked, according to the needs. It is shown that the size of such a pool may be about three orders of magnitude smaller than the actually required number of random numbers (i.e. number of particles times number of time steps times number of coordinate directions taken into account) without significant effects on the resulting concentration fields. Such a procedure will typically save another 10% of computing time in a dispersion calculation.

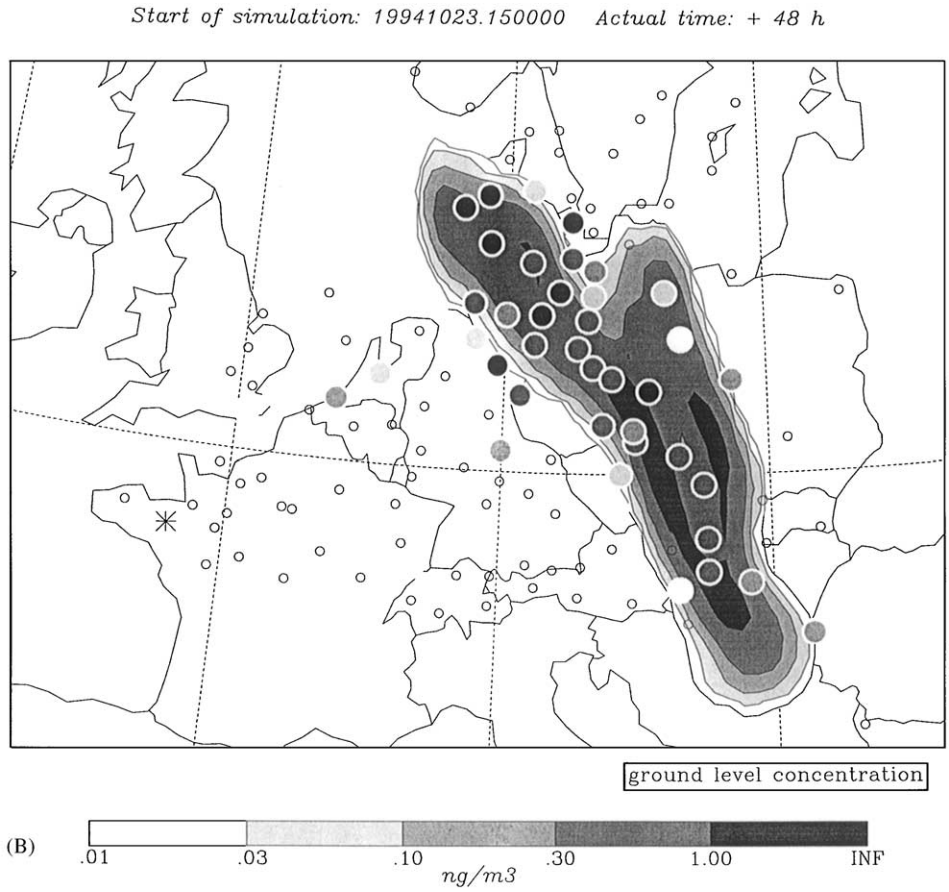

Fig. 5. (*Continued*)

The evaluation of the appropriate time step itself is not very 'expensive' in the original version of the models. However, if the above reduction strategies are followed, this task may easily become the dominant factor (in terms of CPU-time). A method is therefore suggested, which is likely to reduce the required computational time by as much as approximately 90%, again without loss of 'accuracy'.

The over-all reduction which can be achieved with the proposed strategies can be quantified as follows: for the model which uses three-dimensional meteorological fields as input a reduction of almost a factor of 100 is achieved by (massively) increasing the time step. Note that this model is at the same time the simplest of the three considered in that it uses a Gaussian, uncorrelated pdf to describe the turbulent dispersion process. For the models with parameterized meteorological input, the over-all speed-up factor is about 9 in the example of the model for Gaussian, correlated and inhomogeneous turbulence, and about 2 for the most sophisticated model that takes into account non-Gaussian inhomogeneous turbulence. For this latter model, the determination of the accelerations (the functions a_i in Eq. (1)) remains after the speed-up procedures, as suggested here to be by far the most time consuming. However, a further reduction in CPU-time for this model type can probably only be achieved through relaxing some of the constraints (for example, using a vertically homogeneous turbulence field even if the situation under consideration would not allow for such an assumption). This will then result in modified concentration fields.

The reduction potentials in CPU-time as summarized previously for the short-range models apply only to stationary conditions over flat and horizontally homogeneous terrain. Only then, the accelerations can be calculated once for all before the start of the actual dispersion simulation. However, depending on the scale of the spatial inhomogeneity or the time scale of changes in atmospheric conditions, it may still be computationally advantageous to pre-calculate the particle accelerations for certain periods or regions in the modeling domain separately. The calculation of the random

numbers and the control of the time step criterion can be optimized according to the suggested methods, even under non-stationary or inhomogeneous conditions.

Acknowledgements

This work was supported by the Swiss National Science Foundation (grant #21-46849.96). We are thankful for discussions on the subject with Dr. D. Anfossi. Also, we acknowledge the 'Deutscher Wetterdienst' and ECMWF for giving us access to their data (special project 'Validation of trajectory calculations').

References

de Haan, P., 1999. On the use of density kernels for concentration estimates within particle and puff dispersion models. Atmospheric Environment 33, 2007–2021.

de Haan, P., Rotach, M.W., 1998. A novel approach to atmospheric dispersion modelling: the puff-particle model (PPM). Quarterly Journal Royal Meteorological Society 124, 2771–2792.

ECMWF, 1995. Users guide to ECMWF products 2.1. Meteorological Bulletin M3.2. European Center for Medium Range Weather Forcasts, Reading, UK.

Ermak, D.L., Nasstrom, J.S., 2000. A Lagrangian stochastic diffusion method for inhomogeneous turbulence. Atmospheric Environment 34, 1059–1068.

Gryning, S.E., Lyck, E., 1984. Atmospheric dispersion from elevated sources in an urban area: comparisons between tracer experiments and model calculations. Journal Climatology Applied Meteorology 23, 651–660.

Hanna, S.R., 1982. Applications in air pollution modeling. In: Nieuwstadt, F.T.M., van Dop, H. (Eds.), Atmospheric Turbulence and Air Pollution Modelling. D. Reidel Publishing Company, Dordrecht, Holland, pp. 275–310.

Hurley, P., Physick, W., 1993. A skewed homogeneous Lagrangian particle model for convective conditions. Atmospheric Environment 27A, 619–624.

Luhar, A.L., Britter, R.E., 1989. A random walk model for dispersion in inhomogeneous turbulence in a convective boundary layer. Atmospheric Environment 23, 1911–1924.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes, 2nd edn. The Art of Scientific Computing. Cambridge University Press, 818pp.

Rotach, M.W., Gryning, S.E., Tassone, C., 1996. A two-dimensional stochastic Lagrangian dispersion model for daytime conditions. Quarterly Journal Royal Meteorological Society 122, 367–389.

Rotach, M.W., Schwere, S., 2000. A method to speed up a Lagrangian stochastic particle model. In: Gryning, S.-E., Batchvarova, E. (Eds.), Air Pollution Modeling and its Application XIII. Kluwer Academic/Plenum Publishers, New York, pp. 509–517.

Ryall, D.B., Maryon, R.H., 1998. Validation of the UK Met. Office's NAME model against the ETEX dataset. Atmospheric Environment 24, 4265–4276.

Sawford, B.L., Guest, F.M., 1988. Uniqueness and universality of Lagrangian stochastic models of turbulent dispersion. Preprints, 8th Symposium on Turbulence and Diffusion, San Diego, CA, April 25–29, 1988. American Meteorological Society, Boston, MA, pp. 96–99.

Stohl, A., Hittenberger, M., Wotawa, G., 1998. Validation of the Lagrangian particle dispersion model FLEXPART against large scale tracer experiments. Atmospheric Environment 24, 4245–4264.

Stohl, A., Thomson, D.J., 1999. A density correction for Lagrangian particle dispersion models. Boundary-Layer Meteorology 90, 155–167.

Stohl, A., Wotawa, G., Seibert, P., Kromp-Kolb, H., 1995. Interpolation errors in wind fields as a function of spatial and temporal resolution and their impact on different types of kinematic trajectories. Journal Applied Meteorology 34, 2149–2165.

The FLEXPART Particle Dispersion Model. Version 3.0, Users Guide. http://www.forst.uni-muenchen.de/LST/METEOR/stohl/flexpart.html.

Thomson, D.J., 1987. Criteria for the selection of stochastic models of particle trajectories in the turbulent atmosphere. Journal Fluid Mechanics 180, 529–556.

Uliasz, M., 1994. Lagrangian particle dispersion modeling in mesoscale applications. In: Zannetti, P. (Ed.), Environmental Modeling, Vol. II. Computational Mechanics Publications, Southampton, UK, pp. 71–101.

van Dop, H., 1992. Buoyant plume rise in a Lagrangian framework. Atmospheric Environment 26A, 1335–1346.

Wilson, J.D., Legg, B.J., Thomson, D.J., 1983. Calculation of particle trajectories in the presence of a gradient in turbulent-velocity scale. Boundary-Layer Meteorology 27, 163–169.

Wilson, J.D., Sawford, B.L., 1996. Review of Lagrangian stochastic models for trajectories in the turbulent atmosphere. Boundary-Layer Meteorology 78, 191–210.