Supplementary material: The adjoint TM5 model

The adjoint transport model TM5 accounts for the fact that TM5 allows two-way nested zooming. Fig. S1 shows a schematic representation of one time step of the forward TM5 model (Krol et al., 2005) in the case of a global region 1 (the parent), and a zoom region 2 (a child).

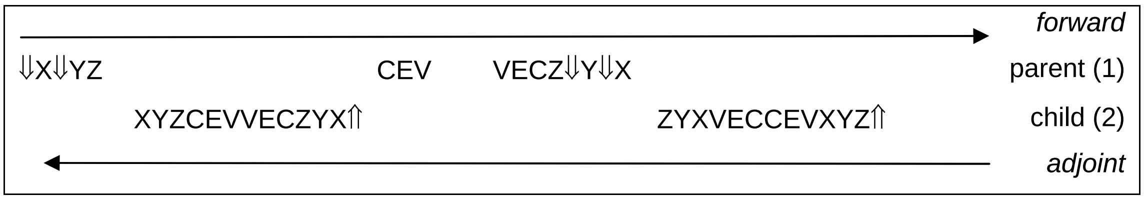

Figure S1. Schematic representation of the building blocks of the forward and adjoint TM5 model.

As described in more detail in Krol et al. (2005), X, Y, Z, C, E, and V represent the subroutines in TM5 for x,y,z-advection, chemistry, emission, and vertical transport (convection and diffusion), respectively. The symbol ⇓ indicates that boundary conditions of the parent region are written to the child regions, whereas the symbol ⇑ indicates that results from the child are used to overwrite that part of the parent that overlaps with the nested higher resolution grid (see Fig. S1). The modular structure of the forward model is used to construct subroutines that are the adjoint of the forward routines. In this way, the correct coding of the various routines could be checked by dedicated testing routines.

Variables in the adjoint model are either active or inactive (Giering and Kaminski, 1998). Active variables represent those variables that are used in the calculation of the tangent linear derivatives, e.g. the adjoint tracer masses, denoted adrm (where rm are the ‘forward’ tracer masses), and the adjoint emissions, denoted adE (where E are the ‘forward’ emissions). Inactive variables must be identical in the forward and adjoint integrations. Examples are temperature, humidity, and winds. In an offline model like TM5 these inactive variables are stored in files and the adjoint model simply reads the same files as the forward model.

As an example, the manual coding of the adjoint TM5 is illustrated by two routines that perform the communication between the parent and child regions (⇓,⇑ in Fig. S1). Table S1 contains a schematic representation of the forward and adjoint source code for the routine that writes boundary conditions (tracer masses) from the parent region (rmp) to the child region (rmc). The situation is illustrated in Fig. S2. The corresponding forward algorithm is outlined in detail in Krol et al. (2005).

Table S1. Examples (in 1-D) of the translation of forward model code to adjoint model code. One child region with a refinement factor *n* is considered.

forward code (⇓)	adjoint ($\Downarrow^T$)
rmc(0) = rmp(i) rmc(1:*n*) = rmp(i+1)/*n*	rmp(i) = rmp(i) + rmc(0) rmp(i+1) = rmp(i+1) + 1/*n* ∑ rmc(1:*n*) rmc(0:*n*) = 0
forward code (⇑)	adjoint ($\Uparrow^T$)
rmp(i+1) = ∑ rmc(1:*n*) rmp(i+2) = ∑ rmc(*n*+1:2*n*) …	rmc(1:*n*) = rmc(1:*n*) + rmp(i+1) rmp(i+1) = 0 rmc(n+1:2*n*) = rmc(*n*+1:2*n*) + rmp(i+2) rmp(i+2) = 0 …

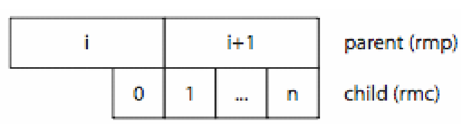

Figure S2. Schematic representation of the alignment between the parent and the child regions. The refinement factor amounts to *n*. The child cell 0 is used to store the boundary condition that is written by the parent.

The adjoint code is most easily constructed by writing the forward code in matrix form. This forward matrix represents the change in the active variables rmp and rmc caused by the program lines in Table S1. The adjoint code follows directly from the transposed matrices (Giering and Kaminski, 1998). For instance, according to the adjoint flow diagram in Table S1, the parent tracer masses are added to the child ($\Uparrow^T$) and subsequently put to zero. It is hard to attach a physical meaning to the adjoint operations, because the adjoint code tracks perturbations backward in time, which is not equivalent to the transport of tracer masses.