

Coupled Transport-Chemistry Computations
in 4D-var Data Assimilation
for
Air Pollution Models

by

Dacian Daescu¹ and Gregory Carmichael²

¹AMCS ²CGRER

THE UNIVERSITY OF IOWA

Model problem:

$$\frac{d}{dt}c_i = -\nabla(\mathbf{u}c_i) + \text{div}(K\nabla c_i) + f_i(c) + E_i \quad (1)$$
$$i = 1 \dots S$$

where

$c(t, x) \in \mathbf{R}^S$ - concentrations vector

$\mathbf{u}(t, x)$ - wind field

$K(t, x)$ - diffusion tensor

$E_i(t, x)$ - source/sink terms

$f_i(t, x, c)$ - chemical reactions

$x \in \Omega \subset \mathbf{R}^3$, bounded

$t \in [t_0, T]$

After space discretization on (N_x, N_y, N_z) :

$$\begin{cases} \frac{dC}{dt} = F_A(c) + F_D(c) + F_R(c) \\ c(t_0) = c_0 \end{cases} \quad (2)$$

Some potential parameters:

- *Initial state*
- Boundary values
- Emissions and depositions rates
- Chemical reactions rates

4D-var with respect to c_0 :

Consider $c = c(t, x, c_0)$

Given

- An estimate c_b of c_0
- Measurements $M_k = M(t_k), k = 1, m$ scattered over $[t_0, T]$

Find c_0 that minimizes:

$$\mathcal{F}(c_0) = \frac{1}{2}(c_0 - c_b)^T \mathbf{B}^{-1}(c_0 - c_b) + \frac{1}{2} \sum_{k=1}^m (c_k - M_k)^T \mathbf{R}_k^{-1}(c_k - M_k) \quad (3)$$

Success of the assimilation depends on:

- Availability of data
- Spatial-temporal distribution of measurements
- Accuracy of the background estimate
- Assimilation window
- Errors in measurements
- Errors in model representation

The gradient of the cost function:

$$\nabla_{c_0} \mathcal{F}(c_0) = \mathbf{B}^{-1}(c_0 - c_b) + \sum_{k=1}^m \left(\frac{\partial c_k}{\partial c_0} \right)^T \mathbf{R}_k^{-1}(c_k - M_k)$$

Algorithm:

1. Initialize $\nabla_{c_0} \mathcal{F} = 0$

2. *for* $k = m, 1, -1$ *do*

$$\nabla_{c_0} \mathcal{F} = \left(\frac{\partial c_k}{\partial c_{k-1}} \right)^T \left[\mathbf{R}_k^{-1}(c_k - M_k) + \nabla_{c_0} \mathcal{F} \right]$$

3. $\nabla_{c_0} \mathcal{F} = \mathbf{B}^{-1}(c_0 - c_b) + \nabla_{c_0} \mathcal{F}$

Intermediate steps:

$$c_{k-1} \rightarrow c_k^1 \rightarrow \dots \rightarrow c_k^s \rightarrow c_k$$

Operator Splitting (*Strang*) $h = t_1 - t_0$

$$c_1 = \overline{F}_A^{\frac{1}{2}, \frac{h}{2}} \overline{F}_D^{\frac{1}{2}, \frac{h}{2}} \overline{F}_R^{0, h} \overline{F}_D^{0, \frac{h}{2}} \overline{F}_A^{0, \frac{h}{2}} c_0$$

\overline{F}_A - explicit, may be nonlinear

\overline{F}_D - (semi)-implicit, linear

\overline{F}_R - highly nonlinear, stiff method

In general at least 2-level checkpointing required.

$$\frac{d}{dt}c = F(c) = F_A(c) + F_D(c) + F_R(c)$$

Second order 2-stage Rosenbrock method

$$\begin{aligned} c_1 &= c_0 + m_1 k_1 + m_2 k_2 \\ (A - \frac{1}{\gamma h} I) k_1 &= F(c_0) \\ (A - \frac{1}{\gamma h} I) k_2 &= F(c_0 + \alpha k_1) + \frac{\beta}{h} k_1 \end{aligned}$$

with $A \approx J_0 \equiv F'(c_0)$

L-stable for $A \equiv J_0$ and

$$\begin{aligned} \alpha &= -\frac{1}{\gamma}; \quad \beta = \frac{2}{\gamma}; \quad m_1 = -\frac{3}{2\gamma}; \quad m_2 = -\frac{1}{2\gamma} \\ \gamma &= 1 \pm \frac{1}{\sqrt{2}} \end{aligned}$$

[Verwer, Blom, Sandu]:

$$\begin{aligned} A &= J_D + J_R - \gamma h J_D J_R \\ (A - \frac{1}{\gamma h} I) &= (I - \gamma h J_D)(J_R - \frac{1}{\gamma h} I) \end{aligned}$$

Adjoint computations:

$$\left(\frac{\partial c_1}{\partial c_0}\right)^T u = u + m_1 \left(\frac{\partial k_1}{\partial c_0}\right)^T u + m_2 \left(\frac{\partial k_2}{\partial c_0}\right)^T u$$

Problems:

- compute $J^T u$
- solve $\left(A - \frac{1}{\gamma h} I\right)^T v = u$
- compute $\left(\frac{\partial(Ak)}{\partial c_0}\right)^T u$

$$J^T u = J_A^T u + J_D^T u + J_R^T u$$

$$\left(A - \frac{1}{\gamma h} I \right)^T = \left(J_R - \frac{1}{\gamma h} I \right)^T (I - \gamma h J_D)^T$$

$$\left(\frac{\partial(Ak)}{\partial c_0} \right)^T u = \left[\frac{\partial(J_R k)}{\partial c_0} \right]^T (I - \gamma h J_D)^T u$$

Compute $\omega = (I - \gamma h J_D)^T u$, then

$$\left[\frac{\partial(J_R k)}{\partial c_0} \right]^T \omega = \left[\frac{\partial(J_R^T \omega)}{\partial c_0} \right] k$$

KPP kinetic preprocessor generates:

- JAC - compute sparse J_R
- DECOMP - sparse $J_R = LU$ decomposition
- SOLVE - forwd/backwd $LUx = b$
- JVECT - sparse $J_R * u$
- JTVECT - sparse $J_R^T * u$
- TSOLVE - forwd/backwd $(LU)^T x = b$