

UC Riverside

UC Riverside Previously Published Works

Title

GPU Implementation of a Gas-Phase Chemistry Solver in the CMAQ Chemical Transport Model

Permalink

<https://escholarship.org/uc/item/973054zh>

Journal

ACS ES&T Air, 2(2)

ISSN

2837-1402

Authors

Quevedo, Duncan

Do, Khanh

Delic, George

et al.

Publication Date

2025-02-14

DOI

10.1021/acsestair.4c00181

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NonCommercial-NoDerivatives License, available at

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Peer reviewed

GPU Implementation of a Gas-Phase Chemistry Solver in the CMAQ Chemical Transport Model

Published as part of ACS ES&T Air special issue "John H. Seinfeld Festschrift".

Duncan Quevedo, Khanh Do, George Delic, José Rodríguez-Borbón, Bryan M. Wong, and Cesunica E. Ivey*



Cite This: ACS EST Air 2025, 2, 226–235



Read Online

ACCESS |

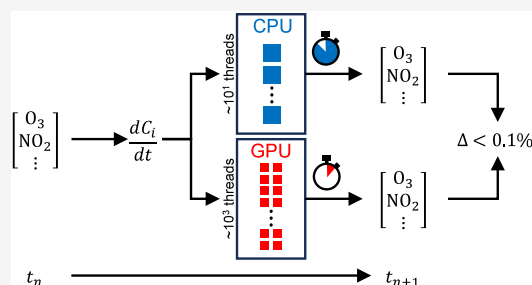
Metrics & More

Article Recommendations

Supporting Information

ABSTRACT: The Community Multiscale Air Quality (CMAQ) model simulates atmospheric phenomena, including advection, diffusion, gas-phase chemistry, aerosol physics and chemistry, and cloud processes. Gas-phase chemistry is often a major computational bottleneck due to its representation as large systems of coupled nonlinear stiff differential equations. We leverage the parallel computational performance of graphics processing unit (GPU) hardware to accelerate the numerical integration of these systems in CMAQ's CHEM module. Our implementation, dubbed CMAQ-CUDA, in reference to its use in the Compute Unified Device Architecture (CUDA) general purpose GPU (GPGPU) computing solution, migrates CMAQ's Rosenbrock solver from Fortran to CUDA Fortran. CMAQ-CUDA accelerates the Rosenbrock solver such that simulations using the chemical mechanisms RACM2, CB6R5, and SAPRC07 require only 51%, 50%, or 35% as much time, respectively, as CMAQv5.4 to complete a chemistry time step. Our results demonstrate that CMAQ is amenable to GPU acceleration and highlight a novel Rosenbrock solver implementation for reducing the computational burden imposed by the CHEM module.

KEYWORDS: GPGPU, CUDA, Fortran, stiff differential equation, Rosenbrock solver



INTRODUCTION

Chemical transport models (CTMs) simulate numerous varied and complex atmospheric processes. These include advection, diffusion, cloud processes, aerosol processes, gas-phase chemistry, and deposition. These processes are modulated by input data including meteorology and emissions. The governing equation describing the simulation of these processes is the species continuity equation, a partial differential equation (PDE) of the form

$$\frac{\partial C_i}{\partial t} + \nabla \cdot (\mathbf{u} C_i) = (\nabla \cdot \mathbf{K} \nabla) C_i + R_i + E_i - S_i \quad (1)$$

where C_i is the concentration of species i , t is time, $\mathbf{u}$ is the wind velocity vector, $\mathbf{K}$ is the eddy diffusivity tensor, R_i and E_i are the net chemical rate of change and emissions of species i , respectively, and S_i is the loss of species i due to deposition.^{1,2}

CTMs tend to employ an operator-splitting strategy in their solution schemes.^{1,2} Operator splitting assumes that each operator in eq 1 can be solved independently with little loss of solution accuracy on sufficiently small time scales. For instance, eq 1 may be split into a family of simpler equations:

$$\frac{\partial C_i}{\partial t} = \begin{cases} -\nabla \cdot (\mathbf{u} C_i) \\ (\nabla \cdot \mathbf{K} \nabla) C_i \\ R_i \\ E_i \\ -S_i \end{cases} \quad (2)$$

Each case of eq 2 is solved separately by its own code module within a CTM's software package.³ Additional processes, including aerosol and cloud processes, are also treated in their own modules. Therefore, each module can be timed separately to determine how long it takes to execute its solution. The chemistry operator is determined by the chemical mechanism chosen to represent atmospheric gas-phase chemistry and takes the form of a net reaction rate. This

Received: July 26, 2024

Revised: December 23, 2024

Accepted: December 24, 2024

Published: January 6, 2025



reaction rate is determined by the solution of a system of coupled ordinary differential equations (ODEs) that needs to be solved for every grid cell in a domain.

Although species lumping is typical to reduce the number of species represented in a chemical mechanism, any mechanism needs to include many species to be representative of atmospheric gas-phase chemistry.⁴ This can result in very large systems of coupled ODEs. The Statewide Air Pollution Research Center (SAPRC, now APRC) mechanism SAPRC07, for instance, contains 569 gas-phase reactions.⁵ Moreover, these reactions are nonlinear and occur over a variety of time scales, producing stiff systems. These characteristics result in longer runtimes for the code modules that solve these systems. In fact, the gas-phase chemistry module often represents a major computational bottleneck in a CTM.^{6,7} Indeed, gas-phase chemistry is often solved using schemes like Euler backward iterative (EBI) solvers for their efficiency, at the cost of solution accuracy. More accurate methods, while slower, can improve the quantification of quantities of interest like sensitivity parameters. Additionally, there is growing interest in leveraging machine learning emulators to bypass traditional numerical integration of these systems.^{7,8} However, there is room for improvement in these emulators' solution approximations in terms of numerical stability and accuracy. Furthermore, emulation has not been standardized for regulatory decision making and emulation does not necessarily capture the physical dynamics that govern first-principles models like CTMs.

One may view emulators as a software-based approach to accelerating gas-phase chemistry modules. Another approach is hardware-based. CTM simulations are traditionally executed on high-performance computing (HPC) clusters with computational processing distributed in parallel across multiple central processing units (CPUs), which are the standard hardware units for computing. An alternative class of hardware capable of carrying out the floating-point arithmetic necessary for the numerical solution of gas-phase chemistry systems is the graphics processing unit (GPU).

With the advent of general purpose GPU (GPGPU) computing, we can leverage the massively parallel nature of GPUs to compute floating-point operations on large data arrays in parallel. Although GPU technology has been driven by advancements in real-time 3D computer graphics as required for interactivity in video games, the nature of graphics rendering, devoting individual GPU threads to rendering individual pixels, drives GPU thread counts higher and higher as pixel counts and graphics complexity increases.⁹ This leads to the ability to compute floating-point operations on large data structures in parallel. Consequently, GPUs present a means of handling larger matrices to reduce the length of serialized loops required to cover study domains.

GPUs outperform CPUs in floating-point operation throughput.¹⁰ Therefore, GPUs represent a potential hardware solution for accelerating gas-phase chemistry modules in CTMs. Moreover, since GPU acceleration is hardware-based, the original CTM algorithms can be retained, albeit with certain modifications to facilitate parallelism. This circumvents the numerical stability and accuracy issues that can arise in emulation. Already, stable GPU implementations of CTM algorithms have been achieved: the horizontal advection scheme in CAMx and the gas-phase chemistry schemes in CAM4-Chem and EMAC.^{11–13}

The hardware acceleration of the horizontal advection module in CAMx achieves impressive performance, with the GPU implementation executing the module up to 28 times as fast as the standard CPU implementation.¹¹ However, the authors report that this module only accounts for around 14% of total simulation time, with chemistry accounting for over 55% of walltime, and do not report the overall acceleration when accounting for all modules. Therefore, we can expect that accelerating chemistry would have a greater impact on total simulation walltime.

Accelerating the chemistry module in CAM4-Chem enabled simulations to execute about 4 times as fast as on standard hardware.¹² However, the simulations in the study are limited to box models with only chemistry active, as opposed to a full CTM simulation.

Hardware accelerated chemistry in a full EMAC simulation was shown to speed application execution by a factor of 1.75.¹³ Accelerated model output for select species mostly stays within 5% of the value of the standard model where discrepancies result from architectural differences in how hardware handles floating point operations. This illustrates the potential performance gains achievable by accelerating gas-phase chemistry modules in CTMs.

In this work, we migrate a gas-phase chemistry solver in the United States Environmental Protection Agency (US EPA)'s Community Multiscale Air Quality (CMAQ) CTM to GPU hardware to accelerate the solution of the reaction system and relieve the computational burden it imposes. We refer throughout this article to the migrated heterogeneous CPU-GPU code as "CMAQ-CUDA" in reference to its use of the Compute Unified Device Architecture (CUDA) general purpose GPU (GPGPU) computing solution. We say heterogeneous to refer to the fact that science processes other than the gas-phase chemistry module, CHEM, are executed on CPUs whereas CHEM is executed on GPUs.

MATERIALS AND METHODS

CMAQ is widely used for regulatory compliance studies at the regional scale.^{14,15} CMAQ ingests meteorology fields and emissions inventories as input and uses these data to inform science processes. CMAQ science processes include horizontal advection, horizontal diffusion, vertical advection, vertical diffusion, gas-phase chemistry, aerosol processes, cloud processes, and dry deposition.³ The advection and diffusion modules solve advection or diffusion equations, respectively. Gas-phase chemistry solves the ODE systems that represent chemical mechanisms. Aerosol processes represent particulate physics and chemistry. Cloud processes handle scavenging, wet deposition, and aqueous chemistry. Dry deposition represents gravitational settling and adsorption onto surfaces. These modules are executed in series in accordance with operator splitting and with their own internal time-stepping schemes. When a certain time step, known as the synchronization time step, is reached, execution of one module concludes, and the next module is invoked to update its science process to the same synchronization time step. This time stepping scheme is diagrammed in Figure 1.

We develop CMAQ-CUDA for CMAQ version 5.4. We focus on the gas-phase chemistry module as it represents one of the major computational bottlenecks for the CMAQ CTM (CCTM) executable, with computation burdens illustrated in Figure 2.

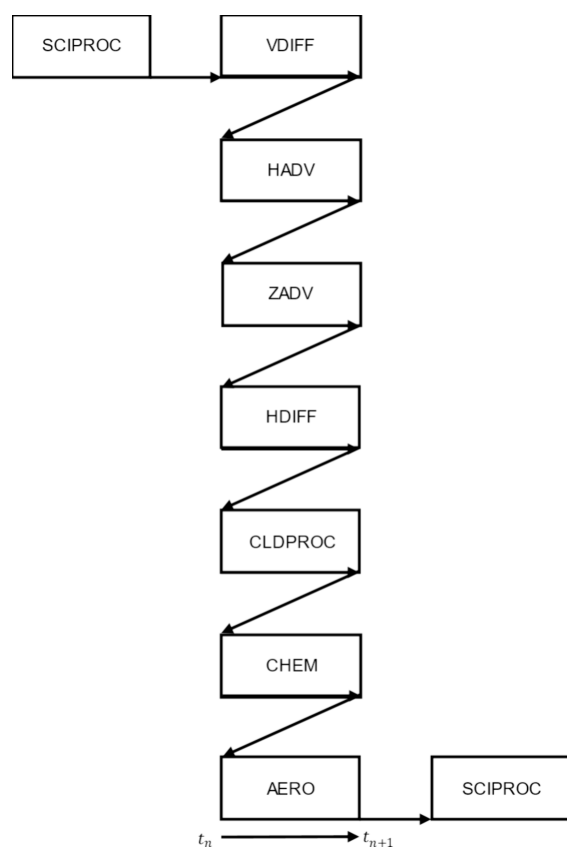


Figure 1. Schematic of CMAQ science process time stepping structure. SCIPROC is the Fortran routine that calls the science process routines. VDIFF is vertical diffusion. HADV is horizontal advection. ZADV is vertical advection. HDIFF is horizontal diffusion. CLDPROC denotes cloud processes. CHEM is gas-phase chemistry. AERO denotes aerosol processes. Times t_n and t_{n+1} are the n th and subsequent synchronization time steps, respectively.

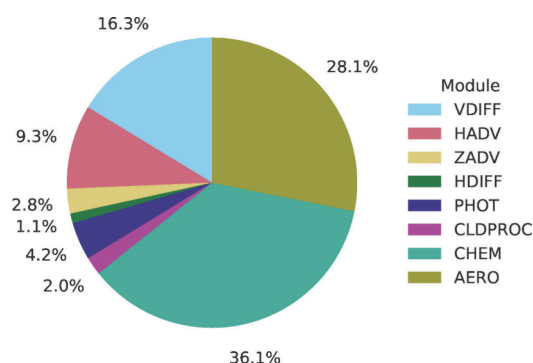


Figure 2. Pie chart of module timing burdens per time step for an 8-core simulation using the ROS3 solver and CB6R5 chemical mechanism with aero7 SOA treatment and standard cloud chemistry for the CMAQ 2018 12NE3 benchmark scenario.

CMAQv5.4 has several different gas-phase mechanisms available and three ODE solver options to numerically integrate the gas-phase ODE system. The mechanisms are versions of the Carbon Bond (CB) mechanism, Regional Atmospheric Chemistry Mechanism (RACM), SAPRC07 mechanism, Community Regional Atmospheric Chemistry Multiphase Mechanism (CRACMM), and the 2D-VBS mechanism.^{4,16–19} The solvers are the Euler backward iterative (EBI), third-order Rosenbrock (ROS3), and sparse matrix

vectorized Gear (SMVGEAR) solvers.^{3,6,20,21} These solvers are listed in descending order of speed and ascending order of accuracy.²¹ The EBI solver is actually a family of solvers because its code implementation must be tailored to the chosen mechanism. In contrast, the ROS3 and SMVGEAR solvers are mechanism agnostic: any given mechanism can be solved by these codes. Since ROS3 is the faster of the two mechanism-agnostic solvers, we focus our efforts on migrating ROS3 to GPUs.

Rosenbrock solvers like ROS3 are multistage semi-implicit schemes.⁶ An s -stage Rosenbrock scheme for the solution of the autonomous equation $\dot{y} = f(y)$ has the form

$$y_{n+1} = y_n + \sum_{i=1}^s b_i k_i \quad (3a)$$

$$k_i = hf \left(y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j \right) + hJ \sum_{j=1}^i \gamma_j k_j \quad (3b)$$

where y_n is the numerical solution at time n , f prescribes the form of the ODE, h is the time step, J is the Jacobian, and the constants s , b_i , α_{ij} , and γ are chosen based on considerations of numerical stability and order of accuracy.⁶ For ROS3, $s = 3$.^{21,22}

The implementation of ROS3 within the CCTM executable is written to leverage parallelization across multiple CPU cores. CPU parallelization is accomplished with the Message-Passing Interface (MPI), which is a software standard that facilitates parallel computing by mapping processes that are generated to execute instruction threads to logical processors and enabling interprocess communication.²³ The simulation domain is divided across MPI processes and each process handles its own subdomain. To accelerate the evaluation of the CHEM module on each subdomain, the matrices describing the scheme are split into smaller block matrices, and each subdomain loops through these blocks. The block size is chosen at compile time. Figure 3 diagrams the domain-process-GPU mapping for our simulation configuration.

In the standard CMAQv5.4 case, each process splits its subdomain into 928 blocks of 50 grid cells, whereas the CMAQ-CUDA case splits each subdomain into just 3 blocks of 17,408 grid cells each, with the caveat that in both cases the final block may be smaller when the block size does not evenly divide the subdomain.

The results we present in this study come from simulations set up to mirror the EPA's 2018 12NE3 case study used to benchmark CMAQv5.4.²⁴ The study area encompasses the northeastern United States with a grid that starts at 84.41W, 35.16N with 100 12-km-wide columns eastward, 105 12-km-wide rows northward, and 35 vertical layers for a total of 367500 grid cells. The benchmark simulates hourly concentrations for July first and second, 2018 and uses the CB6R5 chemical mechanism with aero7 SOA treatment and standard cloud chemistry. We use the same time period, time resolution, and chemical mechanism as well as with the additional mechanisms SAPRC07 and RACM2 with aero6 SOA treatment and standard cloud chemistry to conduct our tests.

Our simulations' compute environment has Intel Xeon Skylake 6130 CPUs rated at 2.1 GHz with NVIDIA RTX 2080 Ti GPUs. Models are compiled with an OpenMPI 5.0.2 wrapper around the Intel OneAPI 2023.1.0 compiler for CPU

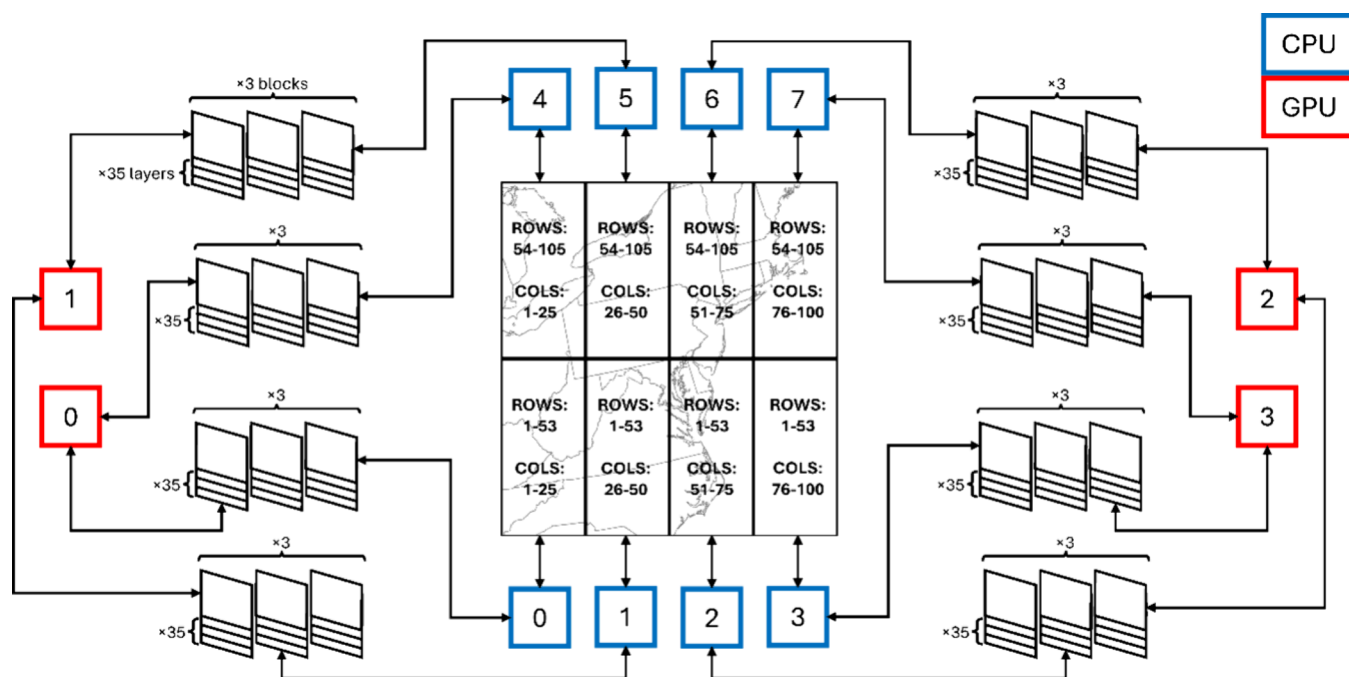


Figure 3. Example schematic of domain subdivision and mapping to MPI processes. Each MPI processes is given its own CPU core and shares GPUs as instructed within our CMAQ-CUDA runtime configuration.

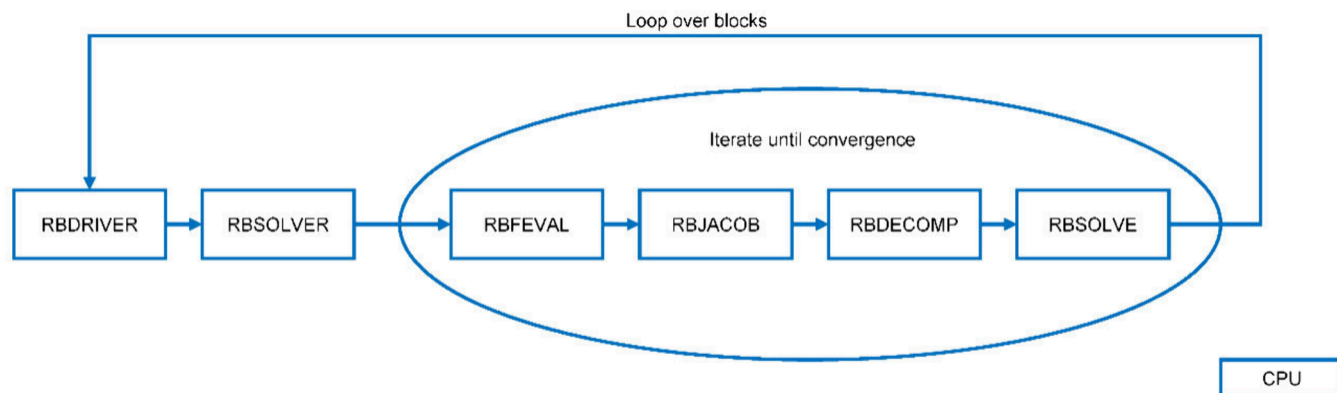


Figure 4. Schematic of CMAQ ROS3 code structure. Arrows represent data transfer. All routines are on the CPU.

code and the NVHPC SDK version 23.9 for GPU code. Our runtime configuration uses 8 CPU cores and 4 graphics cards.

Three popular GPGPU solutions are OpenCL, OpenACC, and NVIDIA CUDA. The ROS3 code, like the rest of CMAQ, is written in the Fortran programming language. We, therefore, leverage CUDA Fortran in our code migration from CPU to GPU as OpenCL is based on C/C++. This avoids the need for interfacing with C from Fortran and enables a native Fortran implementation of CMAQ-CUDA. Although OpenACC supports Fortran, CUDA Fortran offers greater control over optimizations.

CMAQ's ROS3 implementation is split across four subroutines: RBFEVAL, RBJACOB, RBDECOMP, and RBSOLVE. RBFEVAL computes the derivative with respect to time, RBJACOB computes the Jacobian matrix, RBDECOMP computes the LU decomposition of the system to facilitate its solution, and RBSOLVE computes the numerical solution to the linear system of equations based on the LU decomposition. Another subroutine, RBSOLVER, calls the preceding subroutines in the order listed, iterates until solution

convergence, and outputs the concentration field. A further subroutine, RBDriver, calls RBSOLVER in a loop over blocks and fills the MPI process subdomain's concentration field with RBSOLVER's arrays until the whole subdomain's concentration field is updated. Each MPI process does this to update the entire simulation domain's concentration field. This structure is diagrammed in Figure 4.

A typical Fortran nested DO loop in these routines has the following structure for species index J , grid cell index I , and array A :

```
DO J = 1 TO NUMBER OF SPECIES
  DO I = 1 TO NUMBER OF CELLS
    UPDATE A( I, J )
  END DO
END DO
```

These loops step through grid cells to update the spatial dimension of array A and do this for every species in the mechanism.

Because of operator splitting, transport processes (i.e., advection and diffusion) are inactive during execution of the

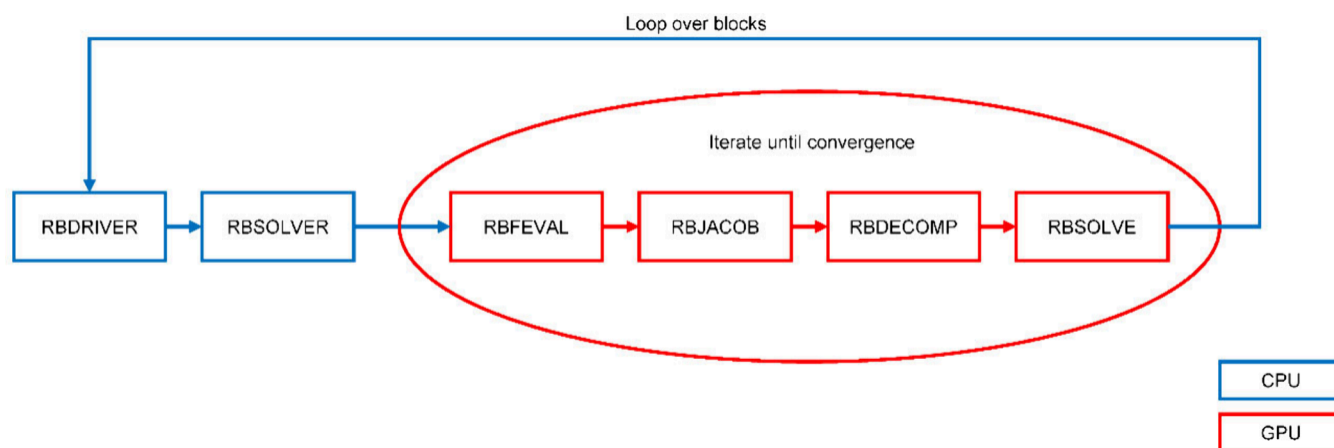


Figure 5. Schematic of CMAQ-CUDA ROS3 code structure. Arrows represent data transfer.

CHEM module, so each grid cell in the simulation domain may be treated independently during CHEM execution, and every cell in a block can be updated in parallel. We, therefore, target the block size dimension of the ROS3 arrays for GPU parallelization: each grid cell is handled by individual GPU threads in parallel.

```

DO J = 1 TO NUMBER OF SPECIES
  DO I = THREAD INDEX TO NUMBER OF CELLS, STEP BY NUMBER OF THREADS
    UPDATE A( I, J )
  END DO
END DO
END DO
  
```

The index I now represents a GPU thread index, with each thread having its own unique index and executing its own copy of the GPU device routine, or kernel. The inner loop becomes a grid-stride loop, where the loop stride is the size of the grid of threads on the GPU. This enables the code to loop through portions of the block if the number of threads initialized on the GPU is insufficient to cover the block at once, although this would slow the code down. Ideally, we initialize enough threads to handle the entire block in parallel. In this case, the grid-stride loop executes just once. This leads to a new ROS3 structure, which we diagram in Figure 5.

With the convergence loop entirely on the device, data transfer only occurs once at the start and once at the end of each synchronization time step. Data transfer between GPU subroutines does not change the data's location in memory, as the convergence loop subroutines are contained in one Fortran module, so arrays are simply updated in place. This greatly relieves the bandwidth bottleneck exposed by data transfer and results in a performant CMAQ-CUDA implementation.

Timing is calculated as the time the model spends in the CHEM science process module, which accounts for the time that the primary MPI process spends on computation in that module and, for CMAQ-CUDA, data transfer overhead. Discrepancies between CMAQ and CMAQ-CUDA concentration fields are reported as mean bias or normalized mean error. With CMAQ output denoted y_i and CMAQ-CUDA output denoted $\hat{y}_i$, we define the following metrics:

$$\text{mean bias(MB)} = \frac{\sum_{i=1}^N (\hat{y}_i - y_i)}{N} \quad (4a)$$

$$\text{normalized mean error(NME)} = \frac{\sum_{i=1}^N |\hat{y}_i - y_i|}{\sum_{i=1}^N y_i} \quad (4b)$$

where N is the number of grid cells and/or time steps and i is the index.

RESULTS AND DISCUSSION

The objective of this work is to accelerate CMAQ's gas-phase chemistry module, CHEM, by leveraging GPU hardware. Figure 6 displays the timing of the CHEM module per

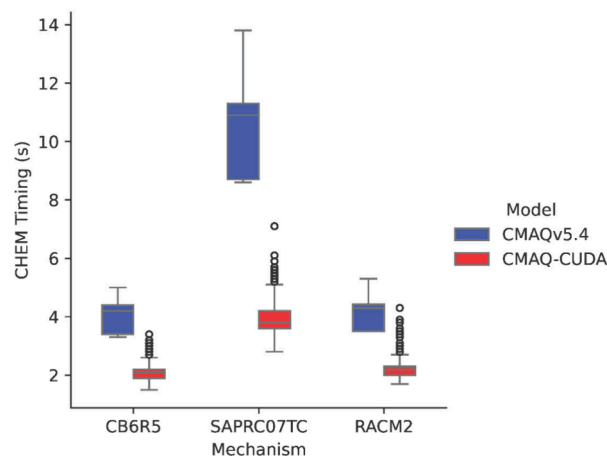


Figure 6. Boxplots of CHEM module timing per synchronization time step for CMAQv5.4 and CMAQ-CUDA across three mechanisms.

synchronization time step of CMAQv5.4 versus CMAQ-CUDA for the chemical mechanisms CB6R5, SAPRC07TC, and RACM2 with aero6 SOA treatment (aero7 for CB6R5) and standard cloud chemistry. These mechanisms contain 349, 752, and 411 reactions, respectively.

We see that, in all cases, CMAQ-CUDA executes CHEM with the ROS3 solver substantially faster than CMAQv5.4. Comparing medians, CMAQ-CUDA executes CB6R5 in 50% as much time as CMAQv5.4, SAPRC07TC in 35% as much time, and RACM2 in 51% as much time. While there is more variability in CMAQ-CUDA timing, the slowest outliers remain near or below the median timing for CMAQv5.4. The variability present in each distribution is due to variation in the number of loops required for the solver to converge, as time steps with greater degrees of nonlinear behavior require more iterations to converge.

Figure 7 presents the total simulation walltime for our three mechanisms. Two features stand out. First, the differential is

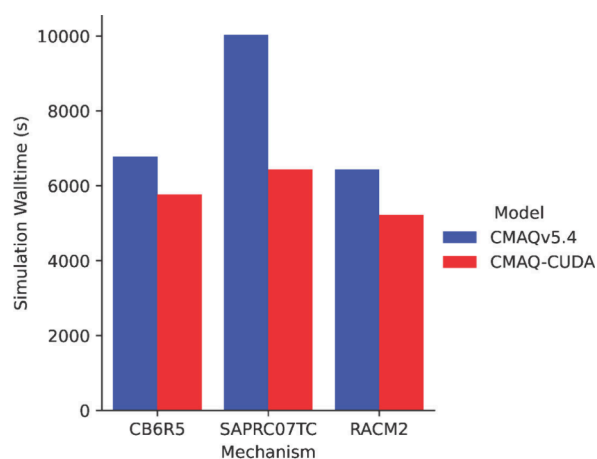


Figure 7. Bar plots of total simulation walltime for CMAQv5.4 and CMAQ-CUDA across three mechanisms.

greatest for the most complex mechanism, indicating that studies simulating larger mechanisms stand to benefit the most from GPU acceleration. Second, the differences between CMAQ-CUDA walltimes across the different mechanisms are smaller than the differences between CMAQv5.4 walltimes across the same mechanisms. This indicates that CMAQ-CUDA scales well in the sense that the rate at which it slows with increasing mechanism complexity is less than the rate at which CMAQv5.4 slows with increasing mechanism complexity. Moreover, for each mechanism, we see notably smaller walltimes even though we have only accelerated one of CMAQ's eight science process modules. This suggests that accelerating other modules in the same way could greatly reduce the walltimes required for CMAQ simulations, saving valuable compute resources.

We explore the impact of varying the block size parameter in Figure 8. Blocking is an efficiency consideration for the CHEM

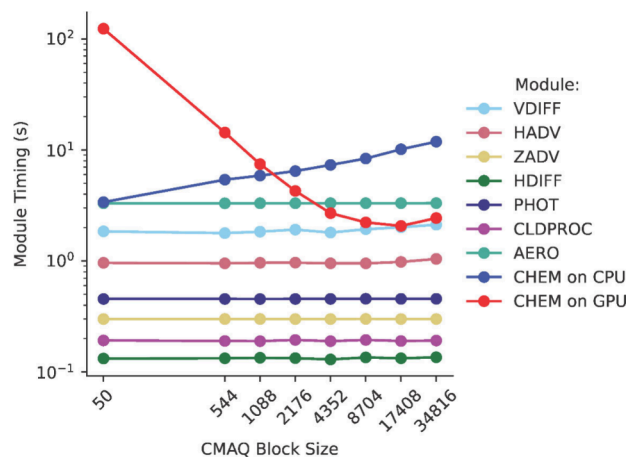


Figure 8. CMAQ science process module timing per synchronization time step as block size varies for CB6R5.

module, so it is unsurprising that we see little to no variation in other modules' timing as we vary block size. CMAQv5.4 is fastest at small block sizes and slows exponentially as we increase the block size. CMAQ-CUDA is also sensitive to block size, but in this case, we see speedup, up to a point, when we increase the block size. The speedup associated with increasing block size is indicative of increasing parallelization.

In the case of GPUs, small block sizes greatly slow execution because this results in more blocks per subdomain. Each block requires kernel calls with their own scheduling overhead in addition to data transfer overhead. Hundreds of small blocks indicate that that overhead gets compounded hundreds of times over and in serial. On the other hand, we see slowdown at very large block sizes as this results in the grid-stride loops executing more than once. The ideal block size indicated by Figure 8 is specific to the NVIDIA RTX 2080 Ti graphics card. It is the number of threads that can reside simultaneously on the GPU when accounting for multiple MPI processes sharing a GPU. Refer to the SI for details.

GPUs are generally designed to excel at single-precision calculations. CMAQ's CHEM module performs double-precision computations. Each double-precision number requires two registers to reside on the GPU for computation. The result is that compiling our kernel that is intensive in double-precision operations reserves 152 registers per thread, regardless of block size. This translates into 37.5% device occupancy, which measures how well a kernel utilizes a GPU's capacity for computation. Higher occupancy generally equates to better performance. To attain greater occupancy for CMAQ-CUDA, we would need to reduce register pressure. One way to achieve this would be to perform single-precision computation, but doing so sacrifices solution accuracy. Indeed, we implemented and tested a single-precision version of CMAQ-CUDA. The solver failed to converge.

While future work should seek to reduce register pressure and accelerate CMAQ-CUDA, even our current low-occupancy implementation attains the significant performance gains documented above, and these are even with the NVIDIA RTX 2080 Ti, a consumer graphics card. Tailoring the block size to the hardware specifications of newer, faster graphics cards may further accelerate CMAQ-CUDA even if occupancy changes little.

In addition to acceleration, we must also ensure that our GPU implementation of ROS3 maintains agreement with its CPU implementation. We want and expect CMAQ-CUDA's output to replicate the standard CMAQ output given the same model inputs, parameters, and configurations. Indeed, according to CMAQ documentation, we are typically satisfied with concentration field discrepancies within 1% when comparing compilations across different computer systems.²¹

We compare MB in the surface layer for three key species between a full simulation and a simulation executed with CHEM as the only active module in Figure 9. This is to investigate the effect of other modules on the overall simulation agreement between CMAQv5.4 and CMAQ-CUDA.

When CHEM is the only active module, CMAQv5.4 and CMAQ-CUDA are in significantly better agreement. This suggests that transferring concentration fields between modules incurs additional error in CMAQ-CUDA, though we note that for simulations with CHEM (but no other modules) deactivated, errors are identically zero (Figure S4). This suggests that the small errors in the CHEM module are magnified by operations in other modules. We posit that this is due to roundoff error magnification in reactive trace species whose concentrations are so small that such errors are disproportionately large. This could be due to architectural differences in how our CPUs and GPUs handle roundoff. We investigate this hypothesis in Figure 10.

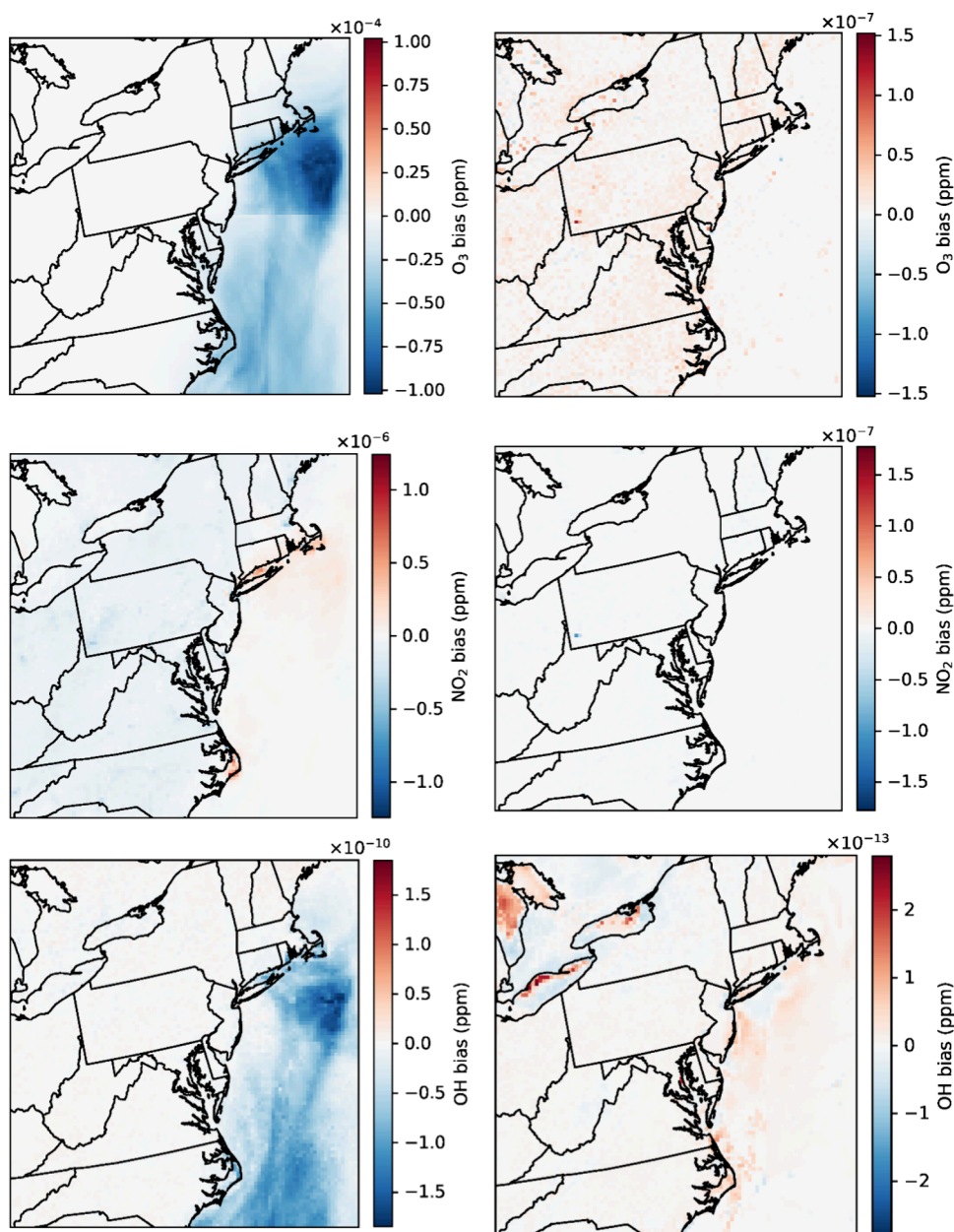


Figure 9. Maps of MB averaged in time for three key species in CB6R5. The left column is the MB between full simulations, the right column is the MB between simulations run with only the CHEM module active.

We see a tendency for NME to increase as median concentrations decrease, supporting our hypothesis. This may also help explain the patterns we see in our full simulations. Certain reactive trace species concentrations may be especially small and prone to roundoff error magnification over water where anthropogenic emissions are minimal.

Figure 11 compares NME in the surface layer for our three mechanisms as well as a benchmark comparison for which the NME is taken between our CMAQv5.4 simulation and the published EPA simulation results. We see that, at the surface, CMAQ-CUDA NME stays within our threshold for most species, and all CMAQ-CUDA NME distributions are concentrated below the benchmark comparison. This indicates that the discrepancy due to hardware differences between our cluster's CPUs and GPUs is smaller than between our architecture and the EPA cluster's architecture.

We expand our error analysis in Figure 12 to examine more vertical layers. There does not appear to be a strong dependence on layer until we reach the 26–30 layer bin, where the distributions begin to shift toward larger NME. Following our earlier hypothesis, it may be the case that errors grow in the upper layers because we exit the planetary boundary layer and certain reactive trace species' concentrations decrease.

In all cases, most species remain within our threshold. In all cases, there are outliers beyond our threshold, and in the top 10 layers, some species in the distributions' tails exceed our threshold. In Figure S5 we see that, in our CHEM-only simulation, all species at all layers, without exception, remain below our threshold. This meets expectations considering our preceding discussion.

We acknowledge the following limitations in our work, in addition to the roundoff error magnification just discussed.

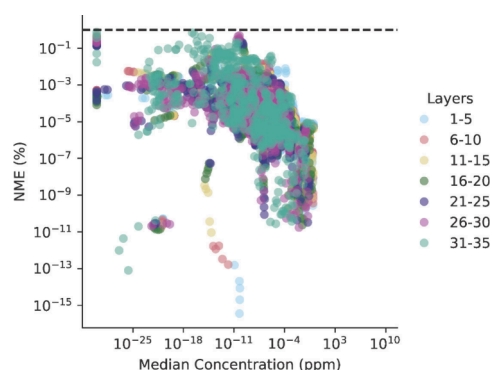


Figure 10. NME against median concentration for all species and all layers in the CB6R5 CHEM-only simulation. Each species appears 35 times, once for each vertical layer. NME is averaged in space (per layer) and time. Our 1% threshold is marked.

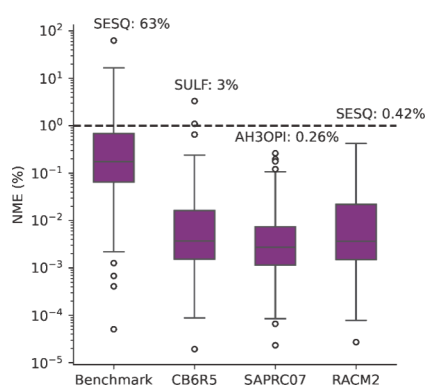


Figure 11. NME for the surface layer across the benchmark and three mechanisms. Each member of the distributions is a different species' NME averaged in space and time. Highest errors are annotated with the corresponding species and its NME value. Our 1% threshold is marked.

Because we migrate CMAQ's ROS3 solver, the integrated source apportionment method (ISAM) tool is unavailable in CMAQ-CUDA. ISAM requires the EBI solver. Further, the direct decoupled method (DDM) sensitivity analysis tool is

not preserved in our current implementation, nor is integrated reaction rate (IRR) analysis.

Despite these limitations, CMAQ-CUDA proves the viability of migrating CMAQ modules to GPU code and highlights the potential performance gains of doing so. Next steps include integrating DDM and IRR into CMAQ-CUDA to enable and accelerate these analysis tools. We also want to migrate additional science process models to CUDA Fortran, starting with the next largest computational burdens: AERO and VDIFF (Figure 2).

CONCLUSIONS

CMAQ-CUDA successfully migrates CMAQ's ROS3 solver to GPU hardware. Our implementation accelerates CMAQ's CHEM module within the modular framework of CMAQ, enabling simulations with all science process modules active. Because we see agreement degrade when CHEM interacts with other modules, we note the importance of expanding our hardware migration to other modules in addition to CHEM.

CMAQ-CUDA highlights the performance gain inherent in GPU hardware acceleration. We have, therefore, reduced the computational burden imposed by CHEM on CMAQ, and in doing so, emphasized the potential for further model acceleration offered by migrating other science process modules to GPU hardware. We have also demonstrated desktop-scale computational improvements in earlier versions of this work, which supports the use of the code in situations where access to HPC resources is limited.²⁵

ASSOCIATED CONTENT

Data Availability Statement

CMAQ-CUDA code is available on GitHub at: <https://github.com/Duncan-Quevedo/CMAQv5.4-CUDAv3.0>. Input data are from the EPA benchmark: <https://dataverse.unc.edu/dataset.xhtml?persistentId=doi:10.15139/S3/BWMI8X>. Output data are available upon request.

Supporting Information

The Supporting Information is available free of charge at <https://pubs.acs.org/doi/10.1021/acsestair.4c00181>.

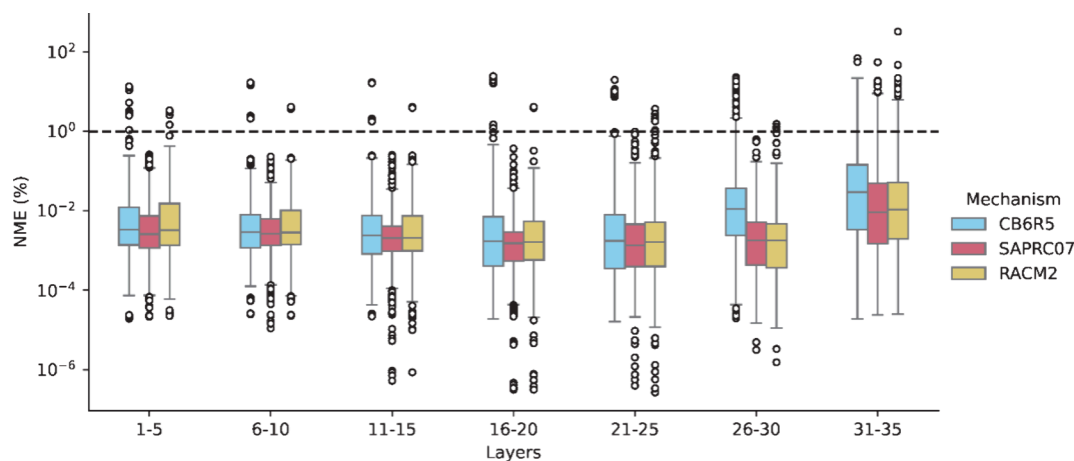


Figure 12. NME for binned vertical layers across three mechanisms. Each member of the distributions is a different species' NME averaged in space and time for the corresponding layer, such that each box plot has 5 members for each species: one for each layer in the corresponding bin. Our 1% threshold is marked.

Code progression, block size determination, MPI timing, and further discussion of CHEM-only simulations along with simulations with CHEM disabled (PDF)

AUTHOR INFORMATION

Corresponding Author

Cesunica E. Ivey – Department of Civil and Environmental Engineering, University of California, Berkeley, California 94720, United States; Center for Environmental Research and Technology, Riverside, California 92507, United States; orcid.org/0000-0002-4740-2627; Email: iveyc@berkeley.edu

Authors

Duncan Quevedo – Department of Civil and Environmental Engineering, University of California, Berkeley, California 94720, United States; orcid.org/0009-0002-7917-6780

Khanh Do – Department of Chemical and Environmental Engineering, University of California, Riverside, California 92557, United States; Center for Environmental Research and Technology, Riverside, California 92507, United States; orcid.org/0000-0002-4587-3337

George Delic – Hiperism Consulting, LLC, Chapel Hill, North Carolina 27514, United States

José Rodríguez-Borbón – Department of Chemistry, Department of Physics & Astronomy, and Material Science & Engineering Program, University of California, Riverside, California 92557, United States

Bryan M. Wong – Department of Chemistry, Department of Physics & Astronomy, and Material Science & Engineering Program, University of California, Riverside, California 92557, United States; orcid.org/0000-0002-3477-8043

Complete contact information is available at:

<https://pubs.acs.org/10.1021/acsestair.4c00181>

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank peer reviewers for their constructive and insightful comments. We thank Prithviraj Yuvaraj for his contribution to the research. This paper was prepared as a result of work sponsored and paid for, in whole or in part, by the Computational and Data-Enabled Science and Engineering from the National Science Foundation Computational Data Science and Engineering Program and the Division of Chemical, Bioengineering, Environmental and Transport Systems (CBET 2053560). The opinions, findings, conclusions, and recommendations are those of the authors and do not necessarily represent the views of the Computational Data Science and Engineering Program. This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley (supported by the UC Berkeley Chancellor, Vice Chancellor for Research, and Chief Information Officer).

REFERENCES

- (1) Seinfeld, J. H.; Pandis, S. N. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change*, 3rd ed.; John Wiley & Sons: Hoboken, NJ, 2016.
- (2) Jacobson, M. Z. *Fundamentals of Atmospheric Modeling*, 2nd ed.; Cambridge Univ. Press: Cambridge, 2005.
- (3) Byun, D.; Schere, K. L. Review of the Governing Equations, Computational Algorithms, and Other Components of the Models-3 Community Multiscale Air Quality (CMAQ) Modeling System. *Applied Mechanics Reviews* **2006**, *59* (2), 51–77.
- (4) Carter, W. P. L. Development of the SAPRC-07 Chemical Mechanism. *Atmos. Environ.* **2010**, *44* (40), 5324–5335.
- (5) Carter, W. *Documentation of the SAPRC-22 Mechanism*; University of California, Riverside College of Engineering Center for Environmental Research and Technology, 2023. <https://intra.engr.ucr.edu/~carter/SAPRC/22/S22doc.pdf> <https://intra.engr.ucr.edu/~carter/SAPRC/22/S22doc.pdf> (accessed 2024–05–10).
- (6) Sandu, A.; Verwer, J. G.; Blom, J. G.; Spee, E. J.; Carmichael, G. R.; Potra, F. A. Benchmarking Stiff Ode Solvers for Atmospheric Chemistry Problems II: Rosenbrock Solvers. *Atmos. Environ.* **1997**, *31* (20), 3459–3472.
- (7) Kelp, M. M.; Jacob, D. J.; Kutz, J. N.; Marshall, J. D.; Tessum, C. W. Toward Stable, General Machine-Learned Models of the Atmospheric Chemical System. *JGR Atmospheres* **2020**, *125* (23), No. e2020JD032759.
- (8) Kelp, M. M.; Jacob, D. J.; Lin, H.; Sulprizio, M. P. An Online-Learned Neural Network Chemical Solver for Stable Long-Term Global Simulations of Atmospheric Chemistry. *J. Adv. Model Earth Syst* **2022**, *14* (6), No. e2021MS002926.
- (9) Nickolls, J.; Dally, W. J. The GPU Computing Era. *IEEE Micro* **2010**, *30* (2), 56–69.
- (10) CUDA C++ Programming Guide. <https://docs.nvidia.com/cuda/cuda-c-programming-guide/> (accessed 2024–05–10).
- (11) Cao, K.; Wu, Q.; Wang, L.; Wang, N.; Cheng, H.; Tang, X.; Li, D.; Wang, L. GPU-HADVPPM V1.0: A High-Efficiency Parallel GPU Design of the Piecewise Parabolic Method (PPM) for Horizontal Advection in an Air Quality Model (CAMx V6.10). *Geosci. Model Dev.* **2023**, *16* (15), 4367–4383.
- (12) Sun, J.; Fu, J. S.; Drake, J. B.; Zhu, Q.; Haidar, A.; Gates, M.; Tomov, S.; Dongarra, J. Computational Benefit of GPU Optimization for the Atmospheric Chemistry Modeling. *J. Adv. Model Earth Syst* **2018**, *10* (8), 1952–1969.
- (13) Alvanos, M.; Christoudias, T. GPU-Accelerated Atmospheric Chemical Kinetics in the ECHAM/MESSy (EMAC) Earth System Model (Version 2.52). *Geosci. Model Dev.* **2017**, *10* (10), 3679–3693.
- (14) Kim, Y.; Fu, J. S.; Miller, T. L. Improving Ozone Modeling in Complex Terrain at a Fine Grid Resolution – Part II: Influence of Schemes in MMS on Daily Maximum 8-h Ozone Concentrations and RRFs (Relative Reduction Factors) for SIPs in the Non-Attainment Areas. *Atmos. Environ.* **2010**, *44* (17), 2116–2124.
- (15) Simon, H.; Baker, K. R.; Phillips, S. Compilation and Interpretation of Photochemical Model Performance Statistics Published between 2006 and 2012. *Atmos. Environ.* **2012**, *61*, 124–139.
- (16) Sarwar, G.; Lueken, D.; Yarwood, G.; Whitten, G. Z.; Carter, W. P. L. Impact of an Updated Carbon Bond Mechanism on Predictions from the CMAQ Modeling System: Preliminary Assessment. *Journal of Applied Meteorology and Climatology* **2008**, *47* (1), 3–14.
- (17) Goliff, W. S.; Stockwell, W. R.; Lawson, C. V. The Regional Atmospheric Chemistry Mechanism, Version 2. *Atmos. Environ.* **2013**, *68*, 174–185.
- (18) Pye, H. O. T.; Place, B. K.; Murphy, B. N.; Seltzer, K. M.; D'Ambro, E. L.; Allen, C.; Piletic, I. R.; Farrell, S.; Schwantes, R. H.; Coggon, M. M.; Saunders, E.; Xu, L.; Sarwar, G.; Hutzell, W. T.; Foley, K. M.; Pouliot, G.; Bash, J.; Stockwell, W. R. Linking Gas, Particulate, and Toxic Endpoints to Air Emissions in the Community Regional Atmospheric Chemistry Multiphase Mechanism (CRACMM). *Atmos. Chem. Phys.* **2023**, *23* (9), 5043–5099.
- (19) Yarwood, G. CMAS Center: Updates to the Carbon Bond Mechanism for Version 6 (CB6).. https://cmascenter.org/conference/2010/abstracts/emery_updates_carbon_2010.pdf (accessed 2024–10–13).

- (20) Z. Jacobson, M.; Turco, R. P. SMVGEAR: A Sparse-Matrix, Vectorized Gear Code for Atmospheric Models. *Atmos. Environ.* **1994**, *28* (2), 273–284.
- (21) CMAQy5.2 Operational Guidance Document, 2017. https://github.com/USEPA/CMAQ/blob/5.2.1/DOCS/User_Manual/PDF/CMAQ_OGD_Full.06302017.pdf (accessed 2024–05–09).
- (22) Damian, V.; Sandu, A.; Damian, M.; Potra, F.; Carmichael, G. R. The Kinetic Preprocessor KPP—a Software Environment for Solving Chemical Kinetics. *Comput. Chem. Eng.* **2002**, *26* (11), 1567–1579.
- (23) Walker, D.; Dongarra, J. MPI: A Standard Message Passing Interface. *Supercomputer* **1996**, *12*, 56–68.
- (24) Adams, E. UNC Dataverse: CMAQ Model Version 5.4 Benchmark Input Data. <https://dataverse.unc.edu/citation?persistentId=doi:10.15139/S3/BWMI8X> (accessed 2024–10–13).
- (25) Do, K.; Delic, G.; Rodríguez-Borbón, J.; Quevedo, D.; Wong, B.; Ivey, C. Graphics Processing Unit Assisted Computation for a Gas-Phase Chemical Solver in a Regulatory Chemical Transport Model. *Earth ArXiv* **2023**, na.