Journal of Pure and Applied Mathematics: Advances and Applications
Volume 21, Number 1, 2019, Pages 1-41
Available at http://scientificadvances.co.in
DOI: http://dx.doi.org/10.18642/jpamaa_7100122086

CPU-GPU ACCELERATION OF THE AIR POLLUTION FORECAST SYSTEM WITH AN EFFICIENT PARALLEL CHEMICAL SOLVER

FAN FENG[1,2,3,4,6], XUEBIN CHI[1,2,4], ZIFA WANG[3], JINRONG JIANG[1,2], LIN WU[3], JIE LI[3], YUZHU WANG[5], GUOFENG ZHOU[6], XIPENG LI[6] and SIMON SEE[6]

[1]Computer Network Information Center
Chinese Academy of Sciences
Beijing
P. R. China
e-mail: feng_fan_100@hotmail.com

[2]Center of Scientific Computing Applications and Research
Chinese Academy of Sciences
Beijing
P. R. China

[3]State Key Laboratory of Atmospheric Boundary Layer Physics and Atmospheric Chemistry (LAPC)
Institute of Atmospheric Physics
Chinese Academy of Sciences
Beijing
P. R. China

2010 Mathematics Subject Classification: 65L04, 65Y05, 65Z05.
Keywords and phrases: chemistry-transport model, CPU-GPU acceleration, gas-phase chemistry module, parallel algorithm, modified-backward-Euler.
Received August 6, 2019

[4]University of Chinese Academy of Sciences
Beijing
P. R. China

[5]School of Information Engineering
China University of Geosciences
Beijing
P. R. China

[6]NVIDIA Corporation
Beijing
P. R. China

Abstract

Chemistry-Transport Model (CTM) plays an important role in the air pollution prevention and control. Their general applications such as forecast and prevention of heavy pollution demand highly efficient CTM simulations. The gas-phase chemistry module is always the most computationally intensive module of a CTM. The main reason is that solving the stiff chemical ordinary differential equations (ODEs) of the gas-phase chemistry module consumes most of the computation time. Here we use the Nested Air Quality Prediction Modelling System (NAQPMS) as the CTM and CBM-Z mechanism as its gas-phase chemistry module. CBM-Z adopts the popular Livermore Solver for Ordinary Differential Equations (LSODE) to solve the chemical ODEs. However, LSODE does not adapt to the parallel acceleration due to its complicated matrix iteration and complex code. In our previous work, we have designed an efficient chemical solver Modified-Backward-Euler (MBE) to improve the simulation speed and precision. In this paper, we review MBE algorithm, show its intrinsic parallelism and port the CBM-Z module with MBE solver on the CPU-GPU architecture to accelerate the NAQPMS further.

1. Introduction

Chemistry-Transport Model (CTM) plays an important role in the air pollution prevention and control. Compared with other modules of the CTM, the gas-phase chemistry module is the bottleneck, which restricts the simulation speed of the whole system. The gas-phase chemistry module fulfils two tasks: constructing the chemical ODEs and using a solver to solve it. The chemical ODEs has numerical difficulties such as

stiffness, nonlinearity and close-to-zero exact solutions. In the past decades, various chemical solvers have been used. However, these solvers are not suitable for parallel acceleration.

We adopt the Nested Air Quality Prediction Modelling System (NAQPMS) [1, 12, 15, 16] as the CTM. NAQPMS is a three-dimensional regional Eulerian CTM that has been widely applied in operational air pollution prevention and control in China. It includes modules such as emission, diffusion, advection, convection, wet deposition, dry deposition, gas-phase chemistry, aqueous-phase chemistry, aerosol chemistry, etc. The gas-phase chemistry module CBM-Z [17] is the most computationally intensive module in NAQPMS. Livermore Solver for Ordinary Differential Equations (LSODE) [4] is used as the original solver, which consumes most of the computation time [2, 3].

CPU-GPU architecture is widely used in the acceleration of scientific calculation [6, 8, 9, 10, 11, 13]. We try to accelerate LSODE on the CPU-GPU architecture. However, the experiments show that the parallel acceleration of LSODE is completely inefficient. The main reasons for this inefficiency are the followings. LSODE solver is based on Gear's Methods, which concerns solving equations by Modified-Newton-Iteration (MNI). For a fixed spatial point, the number of iterations highly depends on the specific values of the parameters (temperature, pressure, location, point of time, etc.). Therefore, the computation time of solving the ODEs for each spatial point is quite different. Moreover, the code of LSODE solver is complicated with many "if-then" branches for different mathematical cases and only points with the same calculation flow can be computed concurrently. All of these make the GPU acceleration of LSODE inefficient and sometimes even slower than the CPU computation.

In our previous work, we have designed a new efficient solver Modified-Backward-Euler (MBE) for the chemical ODEs. MBE is much faster and more precise than the traditional solvers such as LSODE [2,

3]. MBE is based on the "P-L" structure of the chemical equations. It is a very simple efficient explicit algorithm without complicated iteration and all the points have the same calculation flow. The computational complexity of MBE is independent of the values of the parameters, which makes the computation time for each spatial point almost the same. Therefore, MBE has intrinsic parallelism and suits CPU-GPU architecture very well. Therefore, we desire to accelerate CBM-Z module with MBE on the CPU-GPU architecture to speed up the whole system further.

In this paper, we firstly make a general comparison of MBE and LSODE solver. Then we port the gas-phase chemistry module CBM-Z with MBE on the CPU-GPU architecture and embed it into NAQPMS system. In the end, we report the performance of this new version of NAQPMS.

2. Comparison of the MBE and LSODE Solver

We suppose that there are m species, then the ODEs of the gas-phase chemistry module at each spatial point can be described by the following form:

$$\frac{dC_i}{dt} = P_i(C_j, t) - L_i(C_j, t)C_i, \tag{1}$$

$$C_i(0) = C_i^0, \tag{2}$$

where $i, j = 1, 2, \ldots, m$; $P_i \geq 0$, $L_i \geq 0$; C_i represents the concentration of species i; P_i and L_iC_i are the chemical production and loss rates of the species, respectively; C_i^0 is the initial value. P_i and L_i are functions of meteorological parameters, geographical parameters, emission rate and so on. A chemical mechanism such as CBM-Z provides methods of how to use the given parameters to construct P_i and L_i in the equations. Appendix B shows a simple example of the chemical ODEs, which presents the general structures of P_i and L_i.

LSODE solver

LSODE is a popular solver for general ODEs and is used as the original solver of CBM-Z module (Appendix A). However, it is the efficiency bottleneck of NAQPMS and the computation of LSODE may fail because of the small oscillation of Gear's methods and unsuccessful iteration procedure. For detail, please read Appendix A of [2], Appendix D of [3], Appendices C and D of this paper.

Moreover, different values of P_i, L_i, C_i may lead to different code branches of LSODE and affect the number of iterations. That means different spatial points have quite different calculation procedure and running time. Thus, LSODE is not suitable for the parallel acceleration on the GPUs.

MBE solver

To remove the efficiency bottleneck of NAQPMS, an efficient chemical solver is desired. In our previous work, we have designed a simple robust nonnegativity-preserved efficient chemical solver MBE for (1) and (2).

Our MBE solver is

$$C_i^{n+1} = \frac{C_i^n + \Delta t P_i^n}{1 + \Delta t L_i^n} \geq 0, \tag{3}$$

where Δt is the time stepsize and n means the n-th step.

MBE method can be used for any ODEs with the "P-L" structure like (1), where $P_i \geq 0$, $L_i \geq 0$. MBE is much faster and more precise than the traditional solvers such as LSODE. There is no iteration in (3), therefore it is fast and never fails. In fact, the numerical solutions of MBE are also nonnegativity-preserved and converge to the exact solutions as $\Delta t \to 0$ [3].

Moreover, (3) and Table 4 of [2] show that the running time of MBE is independent of the values of P_i, L_i, C_i. That means different spatial points have the same calculation procedure and running time. Thus, MBE is suitable for the parallel acceleration on the GPUs.

Now we compare the two solvers roughly (Table 1).

Table 1. Comparison of MBE and LSODE

	MBE	LSODE
Type of ODEs	"P-L" structure, $P_i \geq 0$, $L_i \geq 0$	General ODEs
Complexity	Simple, without iteration	Complicated, with iteration
Robustness	Never fails	May fail
Speed	Fast	Slow
Nonnegativity-preserved	Yes	No
Converge to the exact solutions as $\Delta t \to 0$	Yes	No
Suitable for GPU acceleration	Yes	No

3. Porting CBM-Z Module with MBE Solver on CPU-GPU Architecture

Compared with other modules of the NAQPMS, the gas-phase chemistry module CBM-Z with the original solver LSODE is the bottleneck, which restricts the simulation speed of the whole system badly. We try to accelerate LSODE on CPU-GPU architecture. However, like we mentioned in Section 2, LSODE is not suitable for the parallel acceleration, which is inefficient (Table 3).

On the contrary, MBE suits the parallel computing on GPUs very well.

In this section, we will port the whole CBM-Z module with MBE on CPU-GPU architecture (Figure 1) in order to achieve a better performance.

For the CPU-GPU architecture, a function that executes on the GPUs is called a kernel and there are mainly three steps for the acceleration:

(1) Allocate memory on both CPU and GPU. Copy data from CPU to GPU.

(2) Start the kernel from CPU. Execute the kernel on the GPU (i.e., do the parallel computing).

(3) Copy the results from GPU to CPU. Deallocate the memory.

The data copied from CPU to GPU include longitude, latitude, height, previous emission rate, temperature, pressure, species concentration and so on. The results copied from GPU to CPU are the present species concentration.

The calculation of each spatial point is independent of others. CPU-GPU architecture has the hierarchy of threads, blocks and grids (Appendix E). The thread is the basic programming unit and one thread is for one spatial point.

In the original CPU NAQPMS system, the module CBM-Z is called by each 3D spatial point serially. In the CPU-GPU architecture NAQPMS, all the spatial points call the module CBM-Z concurrently. This is the key procedure of the acceleration. The parallel computing on the GPUs consists of two parts: one is constructing the P_i^n and L_i^n of the chemical ODEs; the other is using MBE solver to solve these ODEs.

Compared with the CPU version CBM-Z module, the GPU version module abandons the species selection (Appendix A) and stepsize adaptation (Subsection 3.3 of [2]). It calculates all the species at each time step and uses constant step size such as $\Delta t = 5\,\mathrm{sec}$.

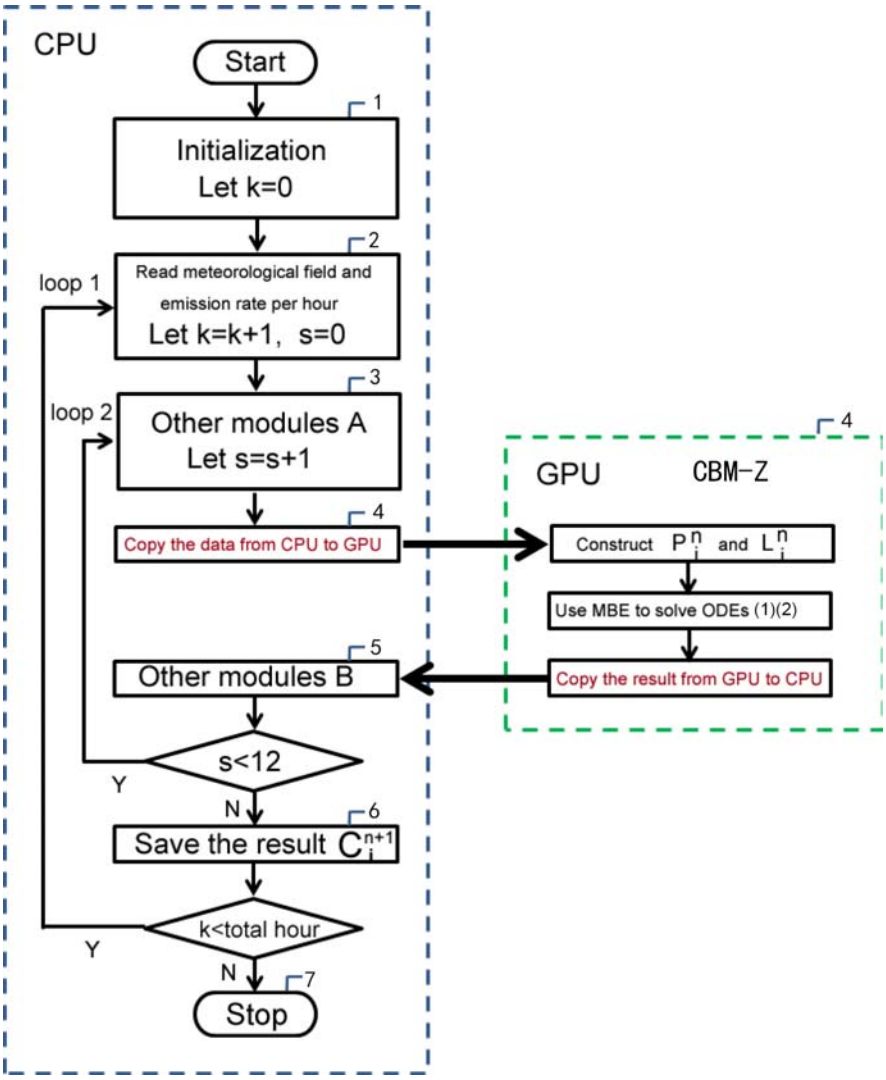


Figure 1. The flow chart of the CPU-GPU architecture NAQPMS system.

Table 2. The 7 steps of Figure 1

---

1. Initialize.

do it1 = 1,TotalHour

2. For each hour, read meteorological field and emission rate.

do it2 = 1,12

3. For each 5 minutes, run the modules before gas chemistry CBM-Z (such as advection, diffusion, dry deposition, etc).

4. For each 5 minutes, run the module CBM-Z on GPUs.

4.1. The initial concentrations and parameters are copied from CPUs to GPUs;

4.2. P_i^n and L_i^n in the chemical equations are constructed on GPUs;

4.3. The chemical equations are solved on GPUs with MBE solver;

4.4. The concentrations of the species are copied from GPUs to CPUs.

5. For each 5 minutes, run the modules after gas chemistry CBM-Z (such as aqueous-phase chemistry, aerosol chemistry, wet deposition, etc).

end do

6. For each hour, save the concentrations of the species to the file.

end do

7. Stop until the end of the simulation time.

---

4. Performance Test

In Subsection 4.1, we show the CPU-GPU acceleration performance of the single CBM-Z module. For the reasons mentioned in Section 2, parallel computing of LSODE is even slower than the original CPU computing. In contrast, our MBE has intrinsic parallelism.

In Subsection 4.2, we embed the GPU version CBM-Z module with MBE solver in the NAQPMS. We will see that CPU-GPU architecture is much faster than CPU.

Therefore, our efficient chemical solver MBE suits the parallel computing very well and plays an important role in the CPU-GPU acceleration of the NAQPMS.

4.1. Acceleration performance of the single CBM-Z module

In the tests (Table 3), we use 1 CPU core (Xeon E5 2690) for the CPU experiments and 1 CPU core (Xeon E5 2690) + 1 GPU (K40, K80 or P100) for the CPU-GPU architecture experiments. For LSODE, we only give K40 result to show that the GPU acceleration of LSODE is inefficient.

Table 3. CBM-Z module speedup with LSODE and MBE (1 CPU core + 1 GPU) vs. 1 CPU core

	K40	K80 (use only 1 GPU)	P100
CBM-Z Speedup(with LSODE)	0.28 < 1	–	–
CBM-Z Speedup(with MBE)	60.9	52.7	116.2

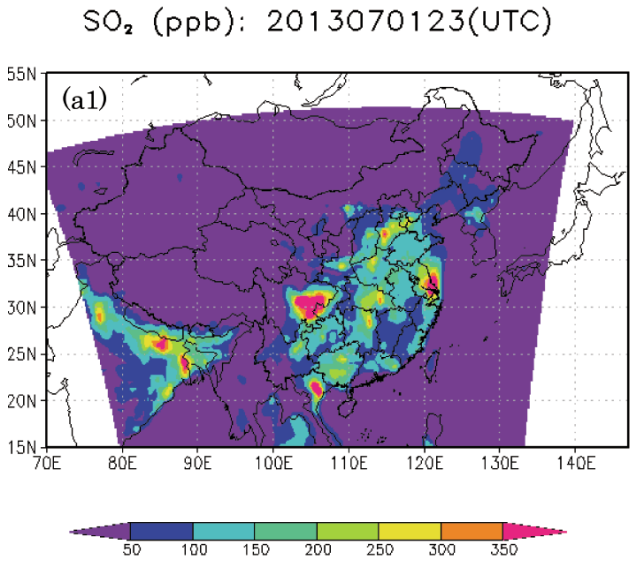
One K80 card has 2 GPUs, here we use only 1 GPU.

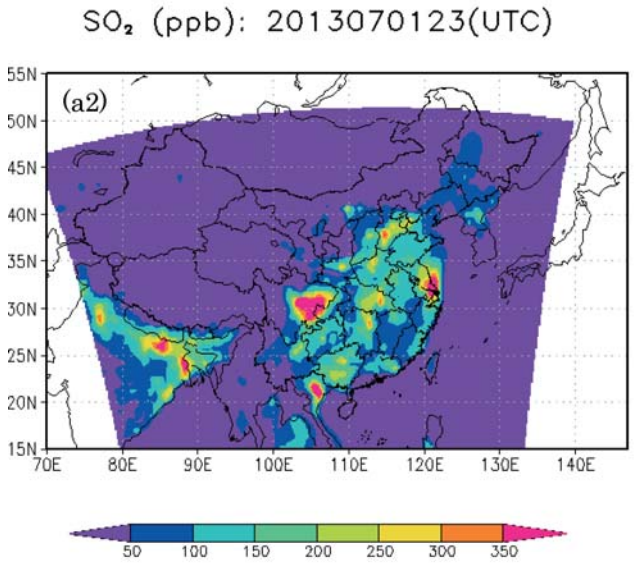
4.2. Embedding GPU version CBM-Z module with MBE solver in the NAQPMS

Here we give an example of one day NAQPMS simulation from UTC 00:00, July, 1st, 2013 to UTC 23:00, July, 1st, 2013 (Figure 2). The horizontal resolution of NAQPMS domain randomly selected is 45km × 45km with 122×132 grids. Vertically, there are 20 layers. The step size of MBE is 5 seconds.

In Table 4, we have two tests. For one test, we use 4 CPUs (Xeon E5 2690) for the CPU experiment and 4 CPUs (Xeon E5 2690) + 4 GPUs (K80 × 2) for the CPU-GPU architecture experiment. For the other test, we use 8 CPUs (Xeon E5 2690) for the CPU experiment and 8 CPUs (Xeon E5 2690) + 8 GPUs (K80 × 4) for the CPU-GPU architecture experiment.

From the example we can see that CPU-GPU architecture is much faster than CPU.





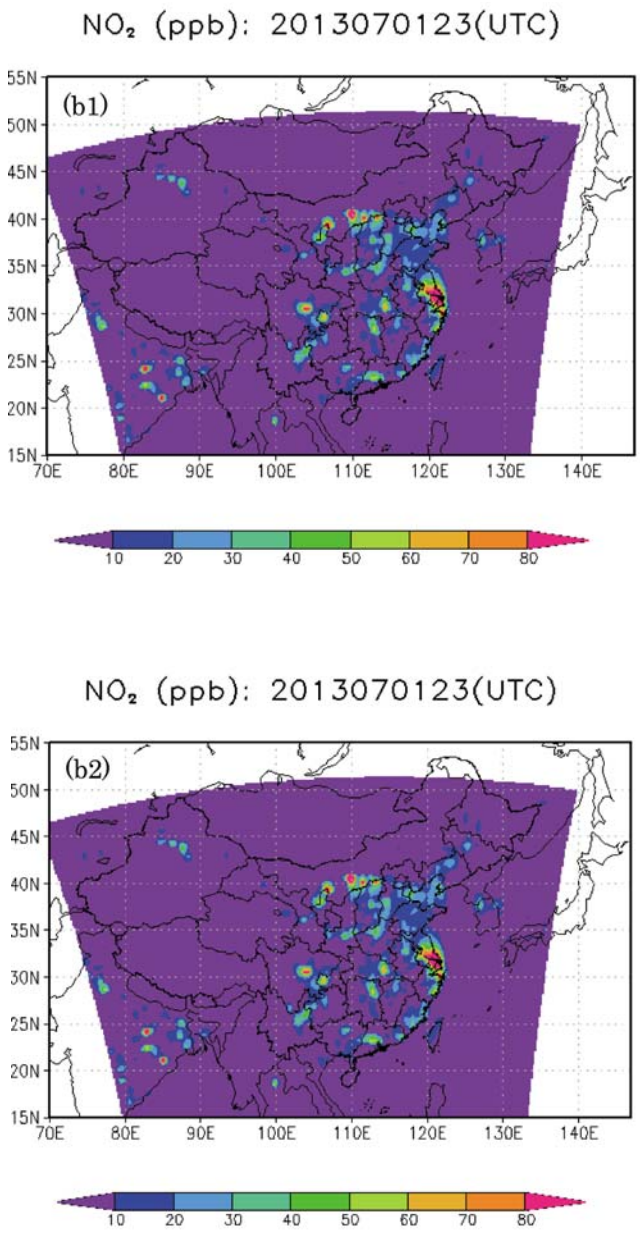


Figure 2. Concentration of SO_2 and NO_2 at UTC 23:00, July, 1st, 2013; (a1) SO_2 with CPU; (a2) SO_2 with CPU-GPU architecture; (b1) NO_2 with CPU; and (b2) NO_2 with CPU-GPU architecture.

Table 4. NAQPMS speedup with MBE

	(4 CPUs + 4 GPUs) vs. 4 CPUs	(8 CPUs + 8 GPUs) vs. 8 CPUs
NAQPMS Speedup	3.52	4.33

One K80 has 2 GPUs.

5. Conclusion

In this paper, firstly we compared our chemical solver MBE with the original solver LSODE. Then we ported the module CBM-Z with MBE solver on the CPU-GPU architecture to accelerate NAQPMS. At last, we gave the numerical performance of the GPU acceleration.

MBE solver is simple, robust, efficient and is suitable for parallel computing. CPU-GPU architecture is a good choice for the acceleration.

Besides the gas-phase chemistry module CBM-Z, the advection and diffusion modules are also time consuming. In the future, we will also port these two modules on CPU-GPU architecture to speed up NAQPMS further.

Acknowledgement

The research is funded by the National Key Research and Development Program of China (No.2016YFB0200800), the Knowledge Innovation Program of the Chinese Academy of Sciences (No.XXH13506-302,XXH13506-402), Strategic Priority Research Program (No.XDC01040000), National Research Program for Key Issues in Air Pollution Control (No.DQGG0106), the National Natural Science Foundation of China (No.41620104008) and NVIDIA Corporation.

References

[1] H. S. Chen, Z. F. Wang, Q. Z. Wu, J. B. Wu, P. Z. Yan, X. Tang and Z. Wang, Application of air quality multi-model forecast system in Guangzhou: Model description and evaluation of PM10 forecast performance, Climatic and Environmental Research 18(4) (2013), 427-435 (in Chinese).

DOI: https://doi.org/10.3878/j.issn.1006-9585.2012.11207

[2] F. Feng, Z. F. Wang, J. Li and G. R. Carmichael, A nonnegativity preserved efficient algorithm for atmospheric chemical kinetic equations, Applied Mathematics and Computation 271 (2015), 519-531.

DOI: https://doi.org/10.1016/j.amc.2015.09.033

[3] F. Feng, X. B. Chi, Z. F. Wang, J. Li, J. R. Jiang and W. Y. Yang, A nonnegativity preserved efficient chemical solver applied to the air pollution forecast, Applied Mathematics and Computation 314 (2017), 44-57.

DOI: https://doi.org/10.1016/j.amc.2017.06.008

[4] A. C. Hindmarsh, LSODE and LSODI, two new initial value ordinary differential equation solvers, ACM-SIGNUM Newsletter 15(4) (1980), 10-11.

DOI: https://doi.org/10.1145/1218052.1218054

[5] L. W. Horowitz, S. Walters, D. L. Mauzerall, L. K. Emmons, P. J. Rasch, C. Granier, X. Tie, J. F. Lamarque, M. G. Schultz, G. S. Tyndall, J. J. Orlando and G. P. Brasseur, A global simulation of tropospheric ozone and related tracers: Description and evaluation of MOZART: Version 2, Journal of Geophysical Research: Atmospheres 108(D24) (2003), 4784.

DOI: https://doi.org/10.1029/2002JD002853

[6] Y. L. Hou, X. N. Cheng and T. S. Ikenaga, Real-time 3D ball tracking with CPU-GPU acceleration using particle filter with multi-command queues and stepped parallelism iteration, 2nd International Conference on Multimedia and Image Processing (ICMIP) (2017), 235-239.

DOI: https://doi.org/10.1109/ICMIP.2017.59

[7] M. R. Houyoux and J. M. Vukovich, Updates to the Sparse Matrix Operator Kernel Emissions (SMOKE) Modeling System and Integration with Models-3, presented at The Emission Inventory: Regional Strategies for the Future Conference, Air and Waste Management Association, Raleigh, N. C. (1999), 1461.

[8] M. C. Lau and R. Srinivasan, A hybrid CPU-Graphics Processing Unit (GPU) approach for computationally efficient simulation-optimization, Computers & Chemical Engineering 87 (2016), 49-62.

DOI: https://doi.org/10.1016/j.compchemeng.2016.01.001

[9] P. L. Li, Y. Luo, N. Zhang and Yu Cao, HeteroSpark: A heterogeneous CPU/GPU spark platform for machine learning algorithms, IEEE International Conference on Networking, Architecture and Storage (NAS) (2015), 347-350.

DOI: https://doi.org/10.1109/NAS.2015.7255222

[10] P. L. Li and Y. Luo, P4GPU: Acceleration of programmable data plane using a CPU-GPU heterogeneous architecture, 17th IEEE International Conference on High Performance Switching and Routing (HPSR) (2016), 168-175.

DOI: https://doi.org/10.1109/HPSR.2016.7525662

[11] A. D. Ma and C. G. Guo, Parallel acceleration of HEVC decoder based on CPU+ GPU heterogeneous platform, 7th International Conference on Information Science and Technology (ICIST) (2017), 323-330.

DOI: https://doi.org/10.1109/ICIST.2017.7926778

[12] X. Tang, J. Zhu, Z. F. Wang, M. Wang, A. Gbaguidi, J. Li, M. Shao, G. Q. Tang and D. S. Ji, Inversion of CO emissions over Beijing and its surrounding areas with ensemble Kalman filter, Atmospheric Environment 81 (2013) 676-686.

DOI: https://doi.org/10.1016/j.atmosenv.2013.08.051

[13] O. Valery, P. F. Liu and J. J. Wu, A collaborative CPU-GPU approach for principal component analysis on mobile heterogeneous platforms, Journal of Parallel and Distributed Computing 120 (2018), 44-61.

DOI: https://doi.org/10.1016/j.jpdc.2018.05.006

[14] Y. Z. Wang, J. R. Jiang, H. Zhang, X. Dong, L. Z. Wang, R. Ranjan and A. Y. Zomaya, A scalable parallel algorithm for atmospheric general circulation models on a multi-core cluster, Future Generation Computer Systems 72 (2017), 1-10.

DOI: https://doi.org/10.1016/j.future.2017.02.008

[15] Z. F. Wang, F. Y. Xie, X. Q. Wang, J. L. An and J. Zhu, Development and application of nested air quality prediction modeling system, Chinese Journal of Atmospheric Sciences 30 (2006), 778-790 (in Chinese).

[16] Q. Z. Wu, Z. F. Wang, A.Gbaguidi, C. Gao, L. N. Li and W. Wang, A numerical study of contributions to air pollution in Beijing during CARE Beijing-2006, Atmospheric Chemistry and Physics 11(12) (2011), 5997-6011.

DOI: https://doi.org/10.5194/acp-11-5997-2011

[17] R. A. Zaveri and L. K. Peters, A new lumped structure photochemical mechanism for large-scale applications, Journal of Geophysical Research: Atmospheres 104(D23) (1999), 30387-30415.

DOI: https://doi.org/10.1029/1999JD900876

[18] Q. Zhang, D. G. Streets, G. R. Carmichael, K. B. He, H. Huo, A. Kannari, Z. Klimont, I. S. Park, S. Reddy, J. S. Fu, D. Chen, L. Duan, Y. Lei, L. T. Wang and Z. L. Yao, Asian emissions in 2006 for the NASA INTEX-B mission, Atmospheric Chemistry and Physics 9(14) (2009), 5131-5153.

DOI: https://doi.org/10.5194/acp-9-5131-2009

■

Appendix A: Introduction to NAQPMS and CBM-Z Module

NAQPMS

The NAQPMS forecast system (Figure A.1) is driven by meteorological data from the Weather Research and Forecasting (WRF) modelling system (Version 3.2) and the emissions data from the Sparse Matrix Operator Kerner Emissions (SMOKE) modelling system [7]. WRF is used to generate the meteorological field for SMOKE and NAQPMS. SMOKE is applied to deal with the emission inventory and provide three-dimensional gridded emission data for NAQPMS. The anthropogenic emission inventory is obtained from Multi-resolution Emission Inventory for China (MEIC) [18]. The output of the NAQPMS is the concentration of the pollutants.

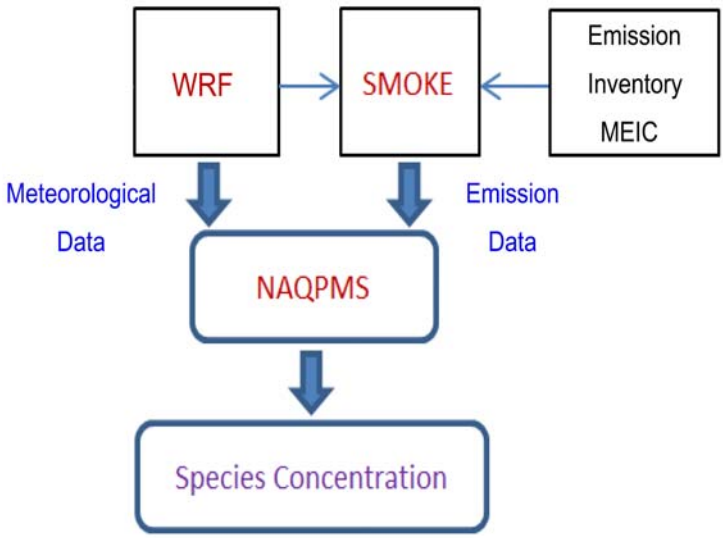


Figure A.1. The framework of the forecast system.

Essentially, NAQPMS is a set of Partial Differential Equations (PDEs) with different terms to describe the physical and chemical processes of the air pollutants. Different terms are computed by different modules. For the computational convenience, NAQPMS adopts the operator splitting method to solve the PDEs. That means NAQPMS runs these modules one by one and the output of one module is the input of the next module.

In order to diminish the error of the operator splitting method, NAQPMS has arranged the order of the modules reasonably: the slower process is before the faster process. The order of the operators is the following:

$$C(t_0 + \Delta t) = L_{wdep} \cdot L_{chem} \cdot L_{ddep} \cdot L_{dif} \cdot L_{conv} \cdot L_{adv} \cdot C(t_0),$$

where L_{adv} is the advection, L_{conv} is the convection, L_{dif} is the diffusion, L_{ddep} is the dry deposition, L_{chem} is the chemistry, L_{wdep} is the wet deposition, $C(t_0)$ is the concentration at time t_0, $C(t_0 + \Delta t)$ is the concentration at the next time step.

CBM-Z module

In NAQPMS, the gas-phase chemistry module CBM-Z [17] mainly fulfils two tasks: to construct the chemical ODEs and to solve the ODEs by using a solver. CBM-Z is a key module of NAQPMS and is also the most time-consuming module. The high computation cost is caused by its original solver LSODE, which concerns complicated matrix iteration (Appendix D).

In CBM-Z module, there are 67 the species (Appendix C of [3]), which means for each spatial point, a set of ODEs with 67 variables needs to be solved. However, for calculation efficiency consideration, CBM-Z module with the original solver LSODE only calculates the selected species at each time step. If the concentration of the species is lower than the given threshold value, it may be ignored at this step. Therefore, the chemical equations to be solved at each time step is always less than 67, that is only an approximation of the 67 ODEs. With the efficient solver MBE, we do not need to select species, but all the 67 ODEs can be solved at each time step.

Appendix B: A Simple Example of the Chemical ODEs

In this section, we give a simple example to show the concrete form of the chemical ODEs (1), (2). Here is a set of chemical reaction (*hv* indicates light condition):

$$\mathrm{NO_2} + hv \xrightarrow{k_1} \mathrm{NO} + \mathrm{O}, \tag{B1}$$

$$\mathrm{O} + \mathrm{O_2} \xrightarrow{k_2} \mathrm{O_3}, \tag{B2}$$

$$\mathrm{O_3} + \mathrm{NO} \xrightarrow{k_3} \mathrm{NO_2} + \mathrm{O_2}, \tag{B3}$$

where the reaction rate of (*Bj*) is k_j, $j = 1, 2, 3$. k_j is the function of longitude (*lon*), latitude (*lat*), altitude above mean sea-level (*alt*), relative humidity (*rh*), temperature (*temp*) and pressure (*pres*). Different spatial points have different constant position parameters: *lon*, *lat* and *alt*. *rh*(*lon, lat, alt, t*), *temp*(*lon, lat, alt, t*), *pres*(*lon, lat, alt, t*) are functions of position and time *t*.

The emission rate of species *i* is $E_i(lon, lat, t)$, $i = 1, 2, 3, 4$.

For a fixed point, k_j and E_i are functions of time *t*:

$$k_j(t) = k_j(lon, lat, alt, rh, temp, pres),$$

$$E_i(t) = E_i(lon, lat, t).$$

For each spatial point, there is a set of $k_j(t)$ *and* $E_i(t)$. *Therefore different spatial points have different sets of chemical ODEs*.

Here we show an example of one point.

Let $C_1(t)$ be the concentration of $\mathrm{NO_2}$, $C_2(t)$ be the concentration of NO, $C_3(t)$ be the concentration of O, $C_4(t)$ be the concentration of $\mathrm{O_3}$, constant $\hat{C}_5$ be the concentration of $\mathrm{O_2}$. Suppose that the emission rates of $\mathrm{NO_2}$ and NO are $E_1(t)$ and $E_2(t)$, respectively.

Then the chemical ODEs is:

$$\frac{dC_1(t)}{dt} = k_3(t)C_2(t)C_4(t) - k_1(t)C_1(t) + E_1(t), \tag{B4}$$

$$\frac{dC_2(t)}{dt} = k_1(t)C_1(t) - k_3(t)C_4(t)C_2(t) + E_2(t), \tag{B5}$$

$$\frac{dC_3(t)}{dt} = k_1(t)C_1(t) - k_2(t)\hat{C}_5C_3(t), \tag{B6}$$

$$\frac{dC_4(t)}{dt} = k_2(t)\hat{C}_5C_3(t) - k_3(t)C_2(t)C_4(t). \tag{B7}$$

In this example, P_i and L_i are:

$$P_1 = k_3(t)C_2(t)C_4(t) + E_1(t), \qquad L_1 = k_1(t),$$

$$P_2 = k_1(t)C_1(t) + E_2(t), \qquad L_2 = k_3(t)C_4(t),$$

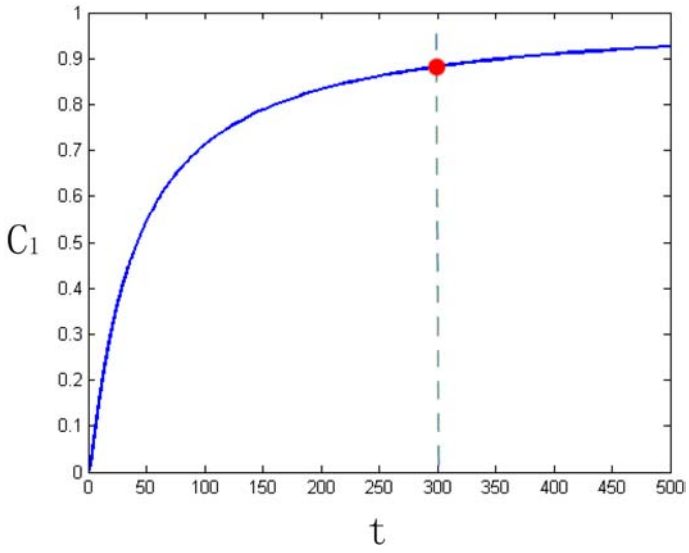
$$P_3 = k_1(t)C_1(t), \qquad L_3 = k_2(t)\hat{C}_5,$$

$$P_4 = k_2(t)\hat{C}_5C_3(t), \qquad L_4 = k_3(t)C_2(t).$$

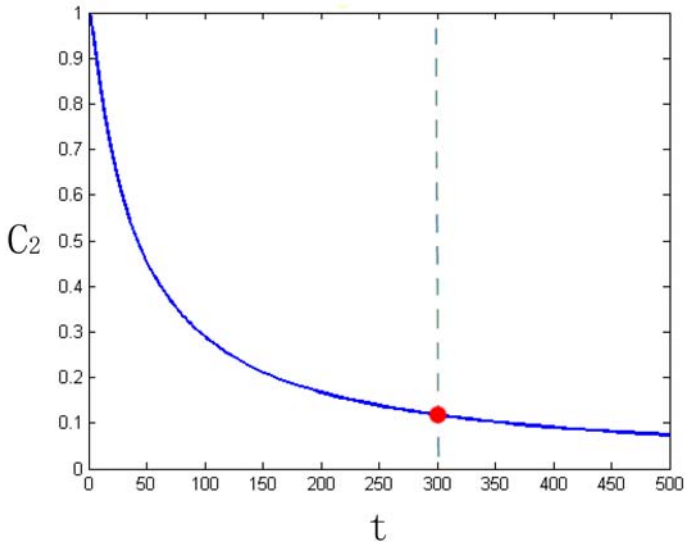
Like we introduced, NAQPMS uses the operator splitting method and calls the CBM-Z module every 5 minutes of simulation time. For the convenience of calculation, in each 5 minutes, $k_j(t)$ and $E_i(t)$ are constructed by the CBM-Z module as constants. Obviously, in the case of constant k_j and E_i, the chemical ODEs is an autonomous system. Therefore, the solution approaches the steady state and never passes through it (Section 2 of [2]).

Figure B.1 presents an example of (B4)-(B7) with constant k_j and E_i. Supposing the unit of "t" is second, then the values of "t = 300" are the output of the CBM-Z module in this 5 minutes of simulation time.

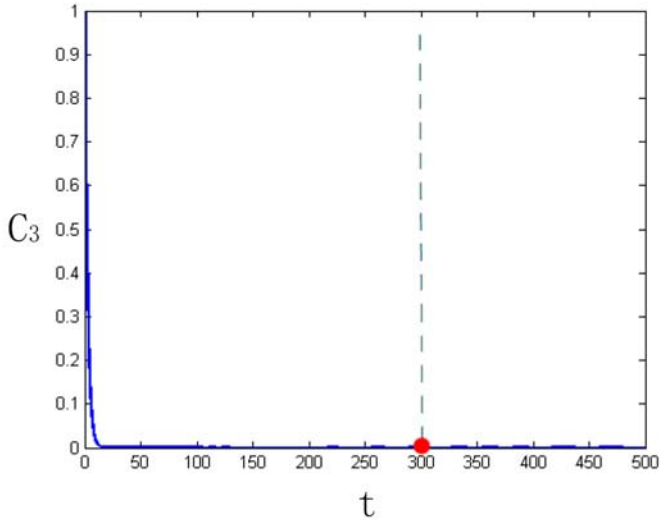
In the computation, the time dependent functions $k_j(t)$ and $E_i(t)$ almost equal to piecewise constant functions. If we ignore the effect of other modules, Figure B.2 shows the computational process of the CBM-Z module for one species. Supposing the unit of "t" is second, then the values of "t = 300, 600, 900" are the output of the CBM-Z module in this 15 minutes of simulation time.


C1
t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400
450
500

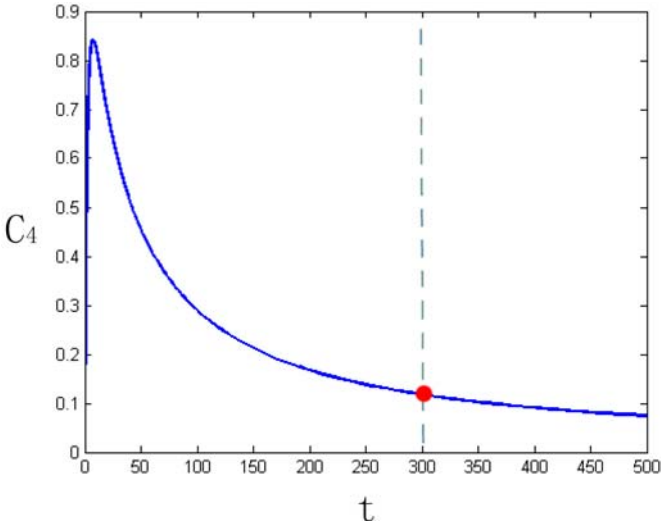
(a)


C2
t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400
450
500

(b)



(c)



(d)

Figure B.1. An example of (B4)-(B7) with constant k_j and E_i. $k_1 = 0.01/1e12$, $k_2 = 4.2e6/1e12$, $k_3 = 2.52e10/1e12$, $E_1 = 0$, $E_2 = 0$, $\hat{C}_5 = 1e5$. The initial values are $C_1(0) = 0$, $C_2(0) = 1$, $C_3(0) = 1$, $C_4(0) = 0$. (a) C_1; (b) C_2; (c) C_3; and (d) C_4.

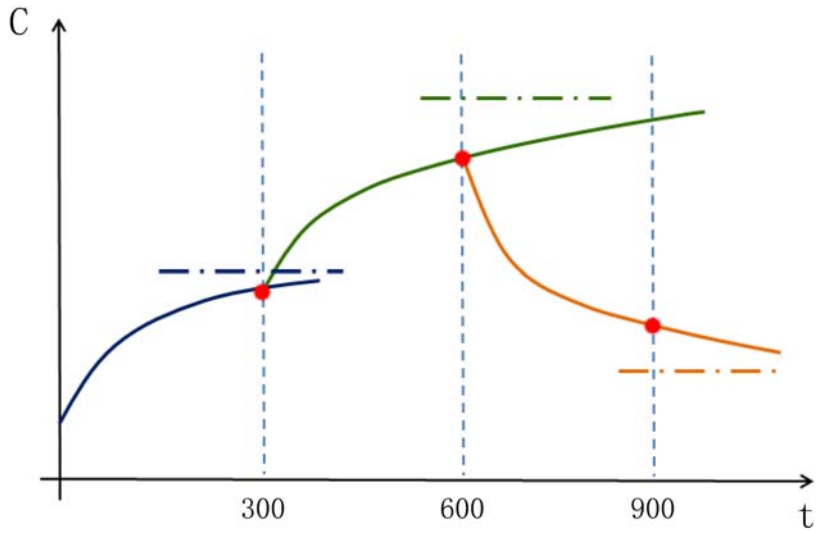


Figure B.2. Sketch of the computational process of the CBM-Z module (Ignore the effect of other modules).

Appendix C: High Order Gear's Methods Do Not Necessarily Give Better Results than Low Order Methods

The original solver of CBM-Z is LSODE, which is based on high order Gear's methods (Appendix D of [3]) and Modified Newton Iteration. Gear's methods are from order 1 to 6. In LSODE, the Gear's order is from 1 to 5. The order of MBE is 1.

In this section, we give some simple examples to show the numerical results of MBE and Gear's methods order 1 to 6. We will see that with the same step size, high order methods do not necessarily give better results than low order methods. The computational error is related to numerical methods, step size, parameter values, initial values and so on.

For initial-value problem (C1), (C2), (C3), (C4),

$$\frac{dy_1(t)}{dt} = -y_1(t), \tag{C1}$$

$$\frac{dy_2(t)}{dt} = ky_1(t) - y_2(t), \tag{C2}$$

$$y_1(0) = 100, \tag{C3}$$

$$y_2(0) = 0, \tag{C4}$$

the exact solutions are

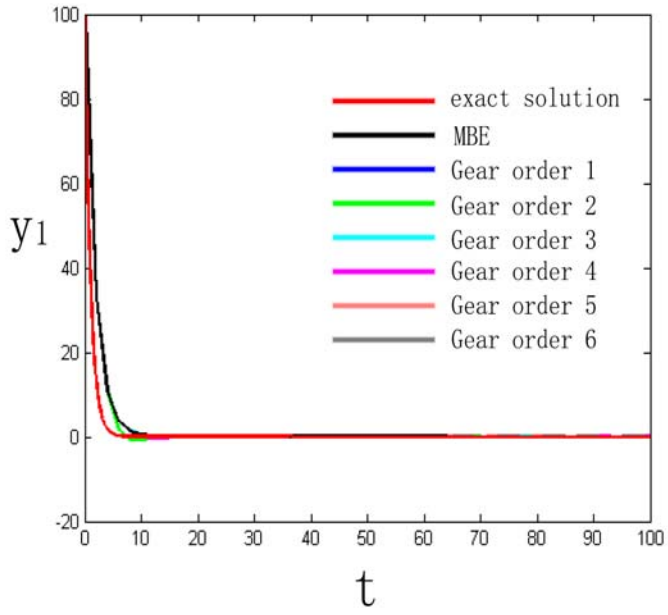
$$y_1(t) = 100e^{-t}, \tag{C5}$$

$$y_2(t) = 100kte^{-t}. \tag{C6}$$

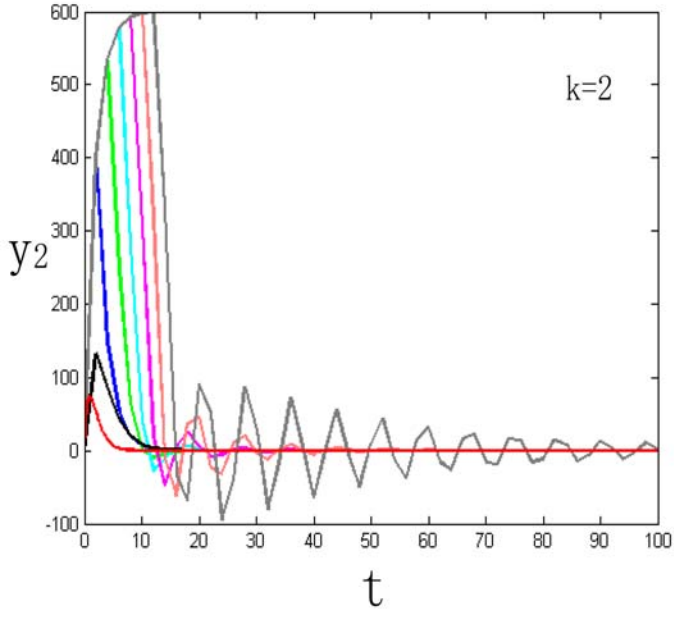
Figure C.1-C.4 present the numerical solutions to initial-value problem (C1), (C2), (C3), (C4) with $k = 2$ or $k = 2000$, when $\Delta t = 2, 1, 0.1, 0.01$, respectively. For order q Gear's method, the q initial values are given by using explicit Euler method. We can see from the figures that with the same step size, high order methods do not necessarily give better results; different parameter values lead to different computational error; if the step size is not small enough, high order Gear's methods oscillate obviously. Table C.1 summarizes the minimum values in Figure C.2, indicating the noticeable oscillation of high order Gear's methods.

To avoid negative numerical results, CBM-Z module forces all the negative results to be zero (i.e., artificial zero) at each time step. Figure C.5 is the artificial zero results of Figure C.2 with $k = 2$. We can see that artificial zero strategy only removes the negative results, which can not decrease the oscillation of the positive part.

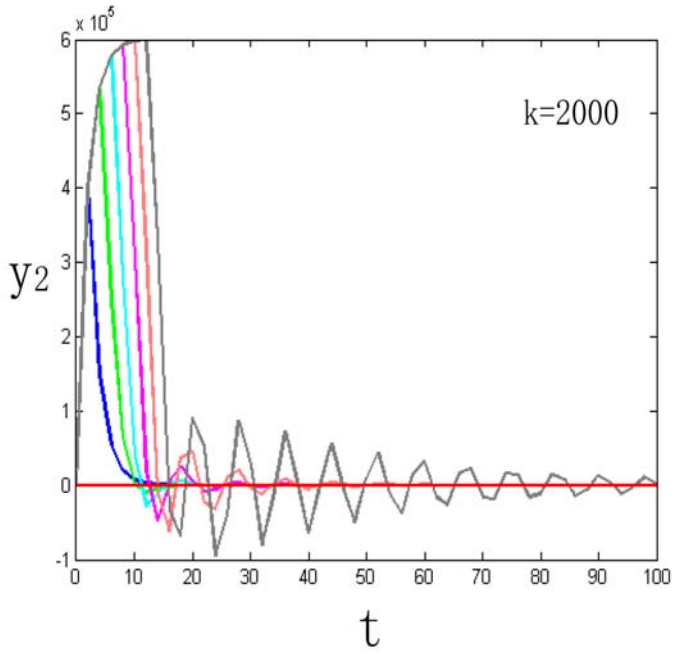
Compared with Figure C.1 with $k = 2$, Figure C.6 shows the results of different initial values. Different initial values lead to different computational error.


exact solution
MBE
Gear order 1
Gear order 2
Gear order 3
Gear order 4
Gear order 5
Gear order 6
y1
t

(a)

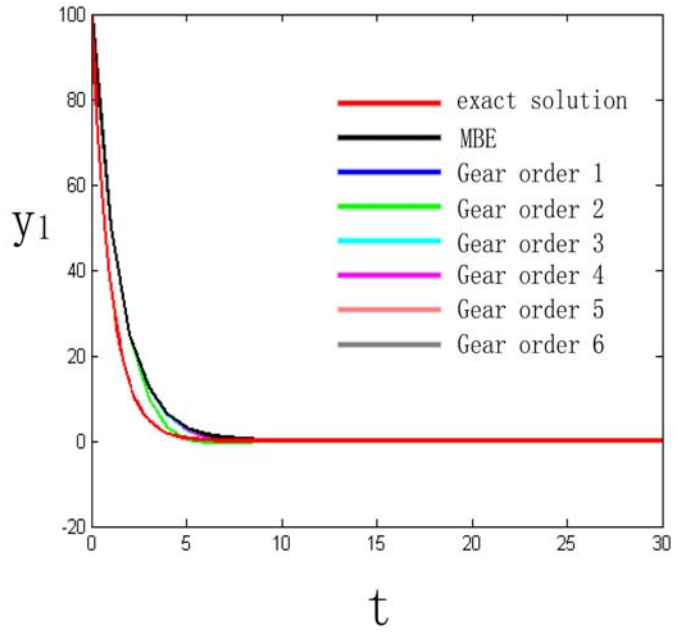

k=2
y2
t

(b)

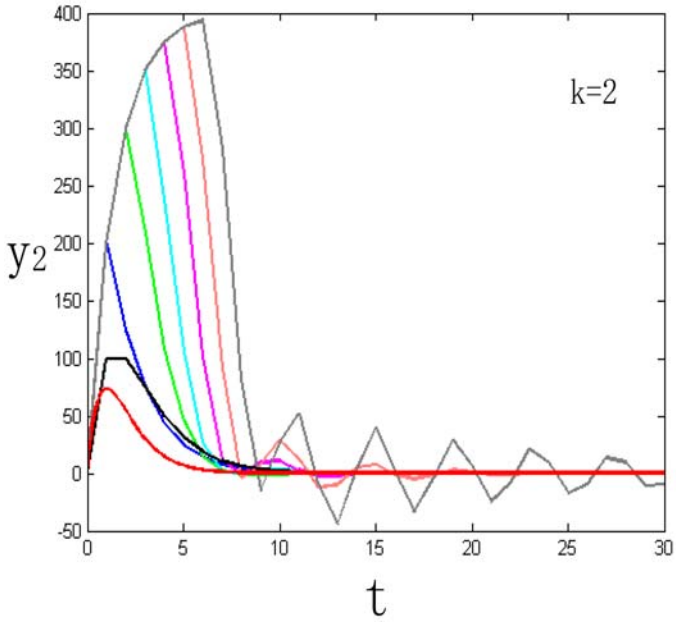


(c)

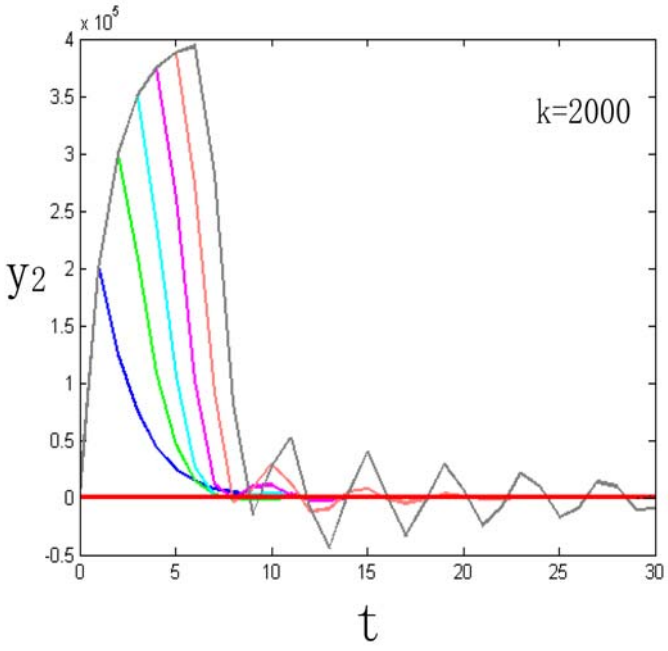
Figure C.1. Solutions to initial-value problem (C1), (C2), (C3), (C4), when $\Delta t = 2$. (a) $y_1(t)$; (b) $y_2(t)$, when $k = 2$; and (c) $y_2(t)$, when $k = 2000$.


exact solution
MBE
Gear order 1
Gear order 2
Gear order 3
Gear order 4
Gear order 5
Gear order 6
y1
t

(a)

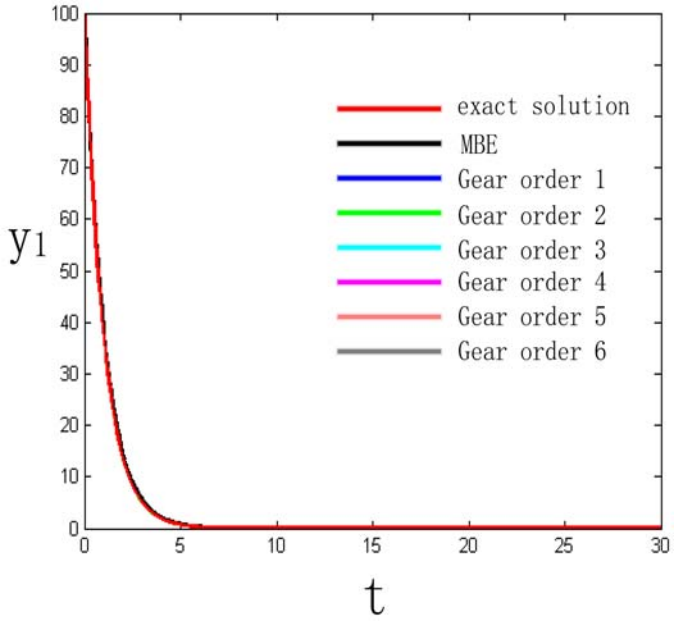

k=2
y2
t

(b)

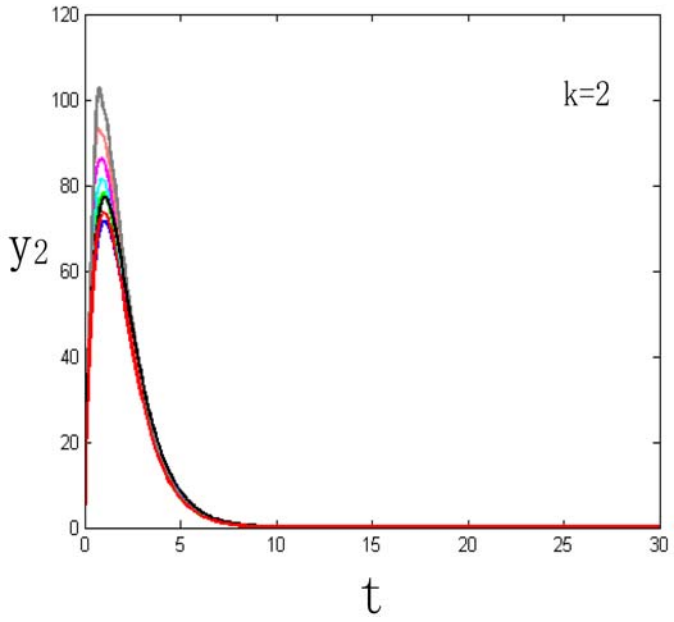


(c)

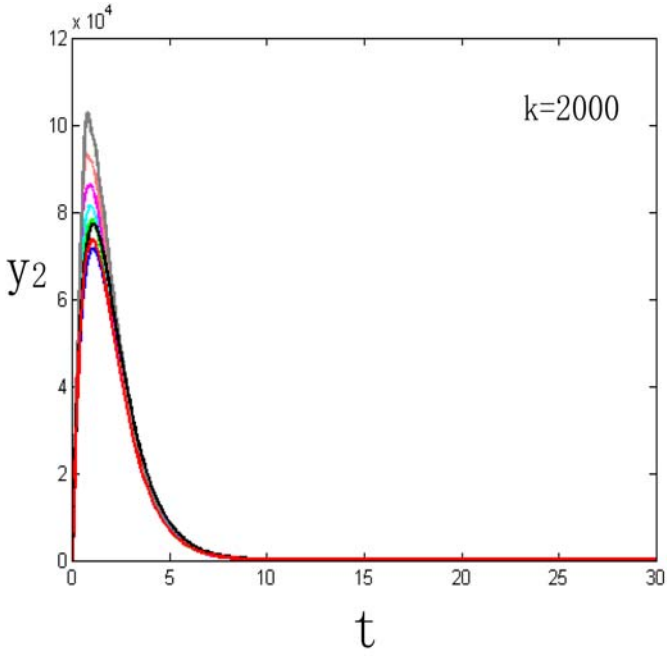
Figure C.2. Solutions to initial-value problem (C1), (C2), (C3), (C4), when $\Delta t = 1$. (a) $y_1(t)$; (b) $y_2(t)$, when $k = 2$; and (c) $y_2(t)$, when $k = 2000$.


exact solution
MBE
Gear order 1
Gear order 2
Gear order 3
Gear order 4
Gear order 5
Gear order 6
y1
t

(a)

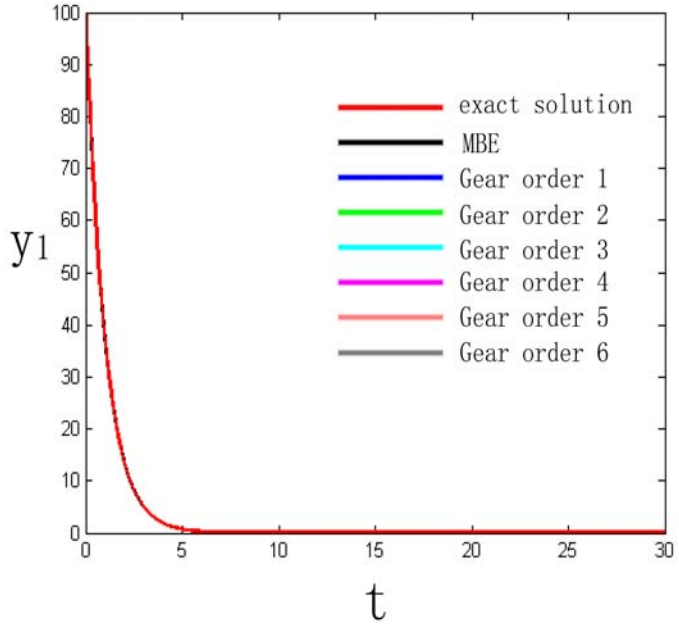

k=2
y2
t

(b)

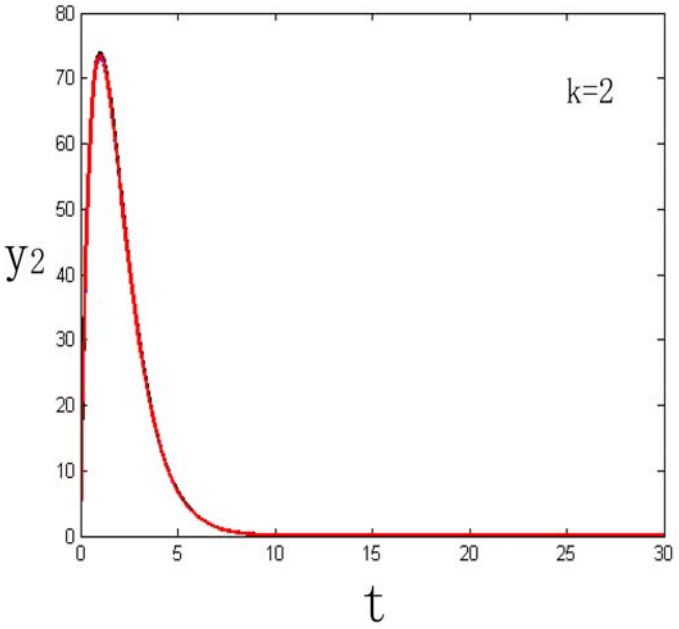


(c)

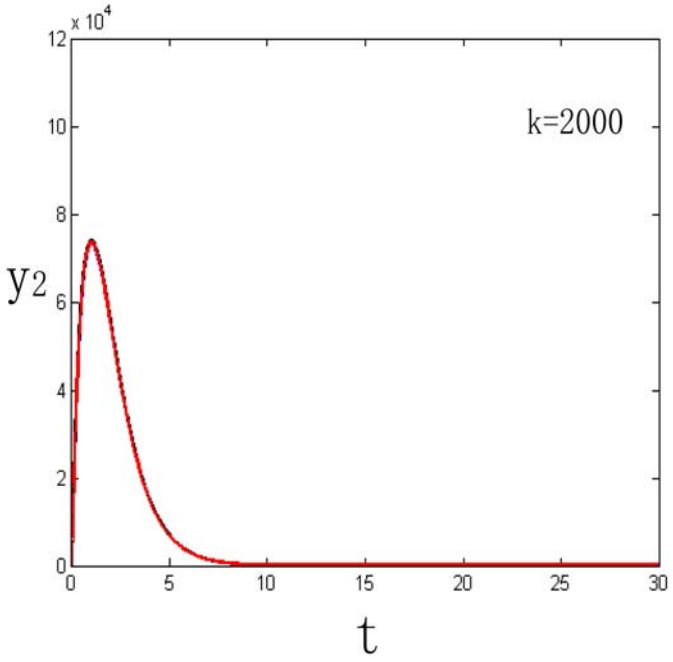
Figure C.3. Solutions to initial-value problem (C1), (C2), (C3), (C4), when $\Delta t = 0.1$. (a) $y_1(t)$; (b) $y_2(t)$, when $k = 2$; and (c) $y_2(t)$, when $k = 2000$.


exact solution
MBE
Gear order 1
Gear order 2
Gear order 3
Gear order 4
Gear order 5
Gear order 6
y1
t

(a)

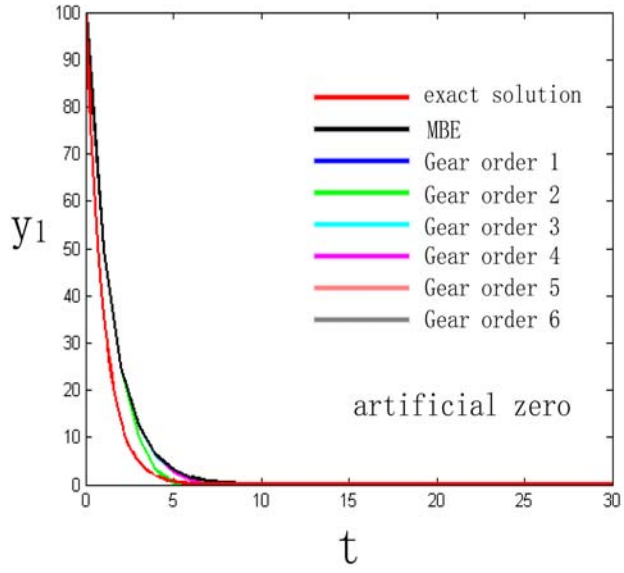

k=2
y2
t

(b)

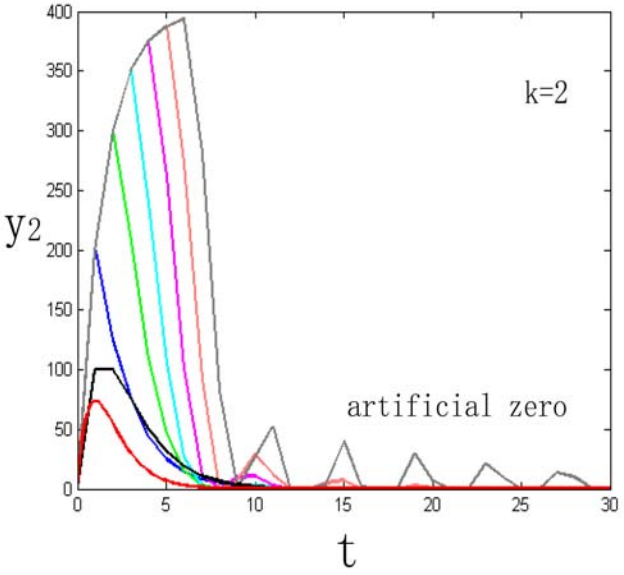


(c)

Figure C.4. Solutions to initial-value problem (C1), (C2), (C3), (C4), when $\Delta t = 0.01$. (a) $y_1(t)$; (b) $y_2(t)$, when $k = 2$; and (c) $y_2(t)$, when $k = 2000$.

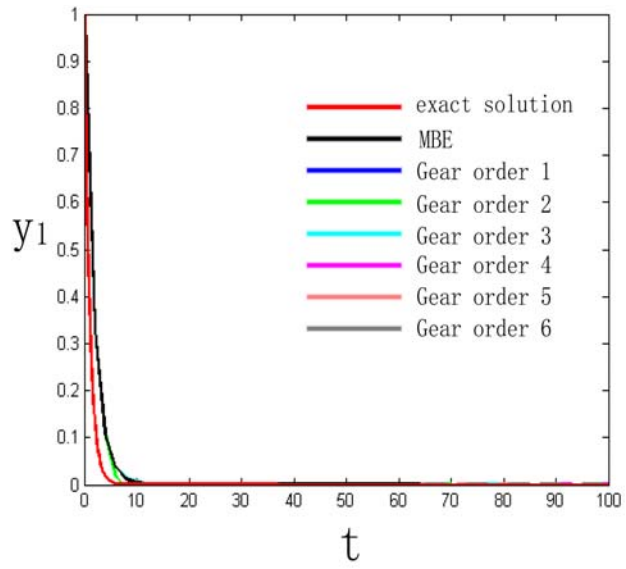


(a)

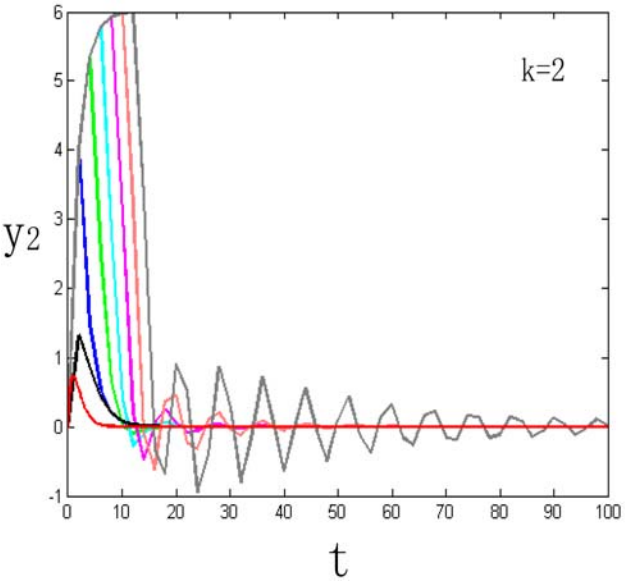


(b)

Figure C.5. Solutions to initial-value problem (C1), (C2), (C3), (C4) with artificial zero, when $\Delta t = 1$. (a) $y_1(t)$; (b) $y_2(t)$, when $k = 2$.



(a)



(b)

Figure C.6. Solutions to initial-value problem (C1), (C2), (C4), and $y_1(0) = 1$, when $\Delta t = 2$. (a) $y_1(t)$; (b) $y_2(t)$, when $k = 2$.

Table C.1. The minimum values of the solutions to initial-value problem (C1), (C2), (C3), (C4) (minimum values in Figure C.2, $\Delta t = 1$, $t \in [0, 30]$, $k = 2$ or $k = 2000$)

	Min y_1	Min $y_2(k = 2)$	Min $y_2(k = 2000)$
Exact solutions	9.3576e − 12	0	0
MBE	9.3132e − 08	0	0
Gear order 1	9.3132e − 08	0	0
Gear order 2	− 0.3040	− 1.5171	− 1.5171e+03
Gear order 3	2.1364e-08	− 0.5284	− 528.3962
Gear order 4	− 0.0242	− 2.3604	− 2.3604e+03
Gear order 5	− 0.0095	− 12.4208	− 1.2421e+04
Gear order 6	− 0.0391	− 44.2688	− 4.4269e+04

In sum, if the time step size is not small enough, high order Gear's methods do not necessarily give better results than low order methods.

However, very small time step size is impractical to LSODE. The reason is that the general equations of Gear's methods are implicit and they need to be solved approximately by complicated iterations of matrix (Appendix D), which means extremely heavy workload.

Moreover, very small time step size of the Gear's equations can not guarantee the accuracy of the iteration results. In other words, even if we do not care about the long computation time and we just want to obtain more accurate numerical results, smaller time step size does not necessarily helpful. The reason is that no matter the value of the time step size, the accuracy of the iteration is another complicated story (Appendix D).

Appendix D: The Sticky Problems of Newton Iteration

The original chemical solver LSODE is based on high order Gear's methods (Appendix D of [3]) and Modified Newton Iteration. In Appendix C, we use very simple examples to present the limitations of high order

Gear's methods themselves. These examples are so simple that their Gear's forms are explicit, no iteration is needed and the numerical results are the exact solutions to

$$C_{n+1} = \sum_{j=0}^{q-1} a_j C_{n-j} + \Delta t b_0 f(C_{n+1}, t_{n+1}). \qquad \text{(Gear's methods) (D1)}$$

In fact, the general Gear's methods have implicit forms usually solved approximately by Newton Iteration or Modified Newton Iteration. That is to use local linearization to give the approximate solution to the equation

$$g(C_{n+1}) = \sum_{j=0}^{q-1} a_j C_{n-j} + \Delta t b_0 f(C_{n+1}, t_{n+1}) - C_{n+1} = 0. \qquad \text{(D2)}$$

Let's suppose that $g(x^*) = 0$ and the initial value x_0 is near x^*, Newton Iteration of this equation is:

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)},\ n = 0, 1, 2, \ldots. \text{ (Newton Iteration for one equation)} \qquad \text{(D3)}$$

when

$$|x_{n+1} - x_n| < \epsilon \text{ or } n > N, \qquad \text{(D4)}$$

then we accept

$$x^* \approx x_{n+1},$$

where ϵ is the given threshold value, N is the given maximum iteration number.

For a set of chemical ODEs,

$$G(C_{n+1}) = \sum_{j=0}^{q-1} a_j C_{n-j} + \Delta t b_0 F(C_{n+1}, t_{n+1}) - C_{n+1} = 0 \qquad \text{(D5)}$$

is a set of equations, where $F = (f_1, f_2, \ldots, f_m)^T$.

Newton Iteration for a set of equations has the following form:

$X_{n+1} = X_n - G'(X_n)^{-1}G(X_n), n = 0, 1, 2, \ldots$ (Newton Iteration for a set of equations), (D6)

where $X = (x_1, x_2, \ldots, x_m)^T$ and $G'(X_n)$ is the Jacobian matrix of $G(X_n)$.

Newton Iteration is popular, but whether the iteration converges to the root we want, how many iteration number it needs and so on are big problems. For simplicity, we give examples of Newton Iteration for one equation to show the problems. Compared with the iteration for one equation, matrix calculation (D6) is obviously more complicated.

Example 1. $g_1(x) = x^3 - kx - 3 = 0, k = 1.$

The 3 roots of $g_1(x) = 0$ are $r_1 = 1.6717$, $r_2 = -0.8358 - 1.0469i$, $r_3 = -0.8358 + 1.0469i$. Obviously, it has only one real root $r_1 = 1.6717$. To obtain r_1 by using Newton Iteration (D3), we firstly write down the derivative $g_1'(x) = 3x^2 - 1$.

Then we use different initial values x_0 to start the iteration (Table D.1). For $x_0 = 0$, x_4 returns to the neighborhood of $x_0 = 0$ and the iteration falls into an infinite loop. To obtain r_1, $x_0 = 1$ uses 5 iteration steps, $x_0 = 0.5$ uses 10 iteration steps, $x_0 = 0.5774$ uses 27 iteration steps, $x_0 = 0.57$ uses 61 iteration steps ($g_1'(\pm\frac{1}{\sqrt{3}}) = 0$ and $\frac{1}{\sqrt{3}} \approx 0.57735$).

Table D.1. The iteration results of Example 1

x_0	0	1	0.5	0.5774	0.57
x_1	– 3.0000	2.5000	– 13	1.9648e+04	–1 33.2168
x_2	– 1.9615	1.9296	– 8.6779	1.3099e+04	– 88.8128
x_3	– 1.1472	1.7079	– 5.7976	8.7326e+03	– 59.2109
x_4	– 0.0066	1.6726	– 3.8737	5.8217e+03	– 39.4774
x_5	– 3.0004	1.6717	– 2.5730	3.8811e+03	– 26.3233
x_6	– 1.9618	1.6717	– 1.6472	2.5874e+03	–17.5558
.....	.....	.....	.....	.....	.....
x_9	– 3.0005	1.6717	1.6732	766.6443	– 5.2259
x_{10}	– 1.9619	1.6717	1.6717	511.0965	– 3.4899
.....	.....	.....	.....	.....	.....
x_{26}	– 1.9619	1.6717	1.6717	1.6763	0.2013
x_{27}	– 1.1475	1.6717	1.6717	1.6717	– 3.4340
.....	.....	.....	.....	.....	.....
x_{60}	– 0.0074	1.6717	1.6717	1.6717	1.6723
x_{61}	– 3.0005	1.6717	1.6717	1.6717	1.6717
.....	.....	.....	.....	.....	.....

Example 2. $g_2(x) = x^3 - kx - 3 = 0$, $k = 3$.

The 3 roots of $g_2(x) = 0$ are $r_1 = 2.1038$, $r_2 = -1.0519 - 0.5652i$, $r_3 = -1.0519 + 0.5652i$. Obviously, it has only one real root $r_1 = 2.1038$. To obtain r_1 by using Newton Iteration (D3), we firstly write down the derivative $g_2'(x) = 3x^2 - 3$.

Then we use different initial values x_0 to start the iteration (Table D.2). For $x_0 = 0$ and $x_0 = 1$, Newton Iteration can not continue. The reason is that $g_2'(x) = 0$ happens during the iterations. To obtain r_1, $x_0 = 1.5$ uses 4 iteration steps, $x_0 = 0.5$ uses 22 iteration steps $(g_2'(\pm 1) = 0)$.

Table D.2. The iteration results of Example 2

x_0	0	1	1.5	0.5
x_1	– 1	Inf	2.6000	– 1.4444
x_2	Inf	NaN	2.2079	– 0.9289
x_3	NaN	NaN	2.1099	– 3.3945
x_4	NaN	NaN	2.1038	– 2.3830
.....	.....	.....	.....	.....
x_{21}	NaN	NaN	2.1038	2.1040
x_{22}	NaN	NaN	2.1038	2.1038
.....	.....	.....	.....	.....

Note: • Inf means $+\infty$.

• NaN means not a number.

Here we summarize some sticky problems of Newton Iteration.

(1) When $g'(x_n) = 0$, Newton Iteration can not continue.

(2) For the same equation, different initial values x_0 lead to quite different number of iterations, some x_0 even lead to iteration non-convergence. It is difficult to choose a proper x_0.

(3) For the same initial value x_0, equations with different parameter values need quite different number of iterations.

(4) Much computational work is required to obtain $g'(x_n)$. For problems in which the analytical $g'(x_n)$ is difficult or impossible to evaluate, $\frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}}$ is used as the substitute, this crude approximation or close to zero $x_k - x_{k-1}$ may give rise to computational error.

(5) If the equation $g(x) = 0$ has more than 1 real root, whether the iteration converges to the desired root is another complicated problem.

(6) Different ϵ and N in (D4) lead to different results.

In the case of Modified Newton Iteration, the problems are similar.

In sum, based on high order Gear's methods and Modified Newton Iteration, the original chemical solver LSODE is complicated, relatively slow, not accurate enough, and not suitable for GPU acceleration.

Appendix E: CPU-GPU Architecture Overview

GPUs (Graphic Processing Units) are parallel processors widely used in the field of high performance computing. Compared with CPUs, GPUs possess more transistors devoting to data processing rather than data caching and flow control (Figure E.1). Nowadays, the CPU-GPU architecture is popular: CPUs are used for irregular data structure and unpredictable access mode; GPUs are used for regular data structure and predictable access mode. CPU-GPU architecture applications run the sequential part of the workload on the CPUs and run the parallel part on the GPUs. The cooperation of CPUs and GPUs optimizes the overall performance of the systems in many applications.

We refer to the CPUs and their memory as the host and refer to the GPUs and their memory as the device. Host and device communicate with each other through PCIe.

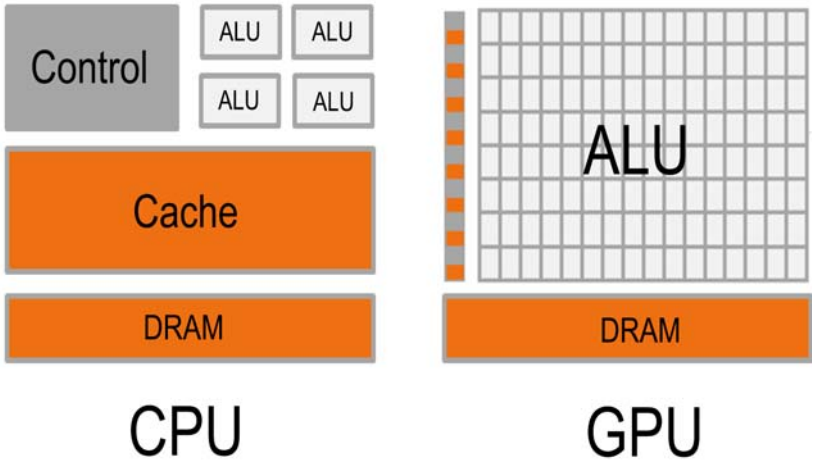


Figure E.1. GPUs possess more ALU.

For CPU-GPU architecture, the thread is the basic programming unit and one thread is responsible for the calculation of one spatial point. Several threads form a block. Blocks executing the same program form a grid.

The entire programme is composed of host code and device code. The host code is the serial part and is executed on the CPUs; the device code (so called the kernel) is the parallel part and is executed on the GPUs (Figure E.2).

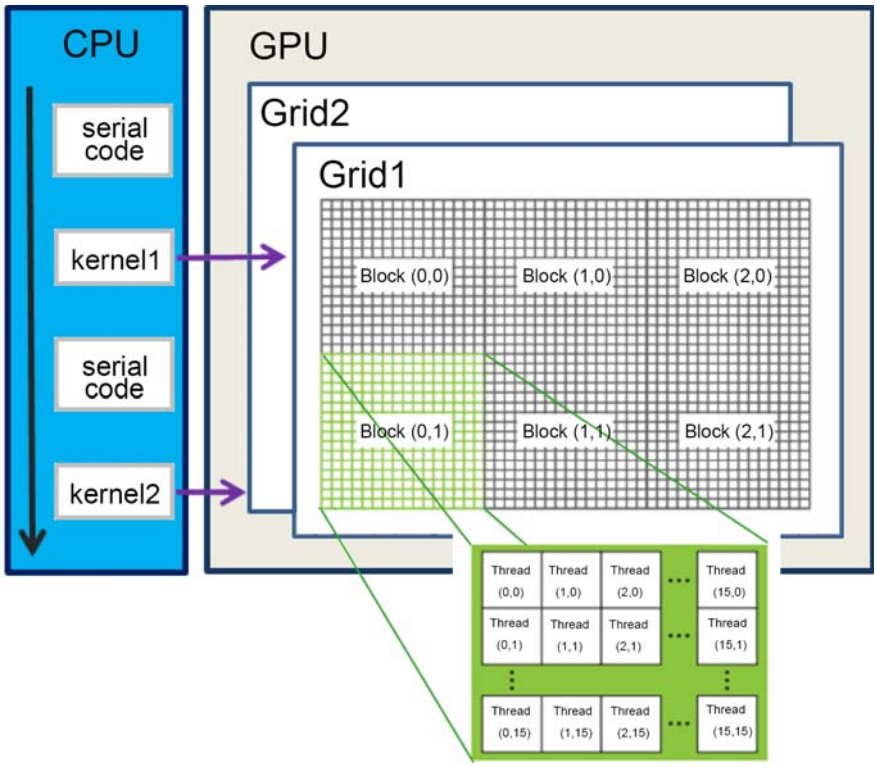


Figure E.2. The hierarchy of threads, blocks and grids of CPU-GPU architecture.