Streamlining Coding Assignments and Grading on the Cloud: A Preconfigured JupyterHub Image for Chemistry Education

*Lechen Dong, Fang Liu**

Department of Chemistry, Emory University, 1515 Dickey Dr, Atlanta, GA 30322

Keywords

Upper-Division Undergraduate, Graduate Education, Physical Chemistry, Computer-Based Learning, Assessment, Theoretical Chemistry

Abstract: Integrating coding skills into chemistry education is crucial for preparing students to meet the demands of modern research. However, the technical challenges associated with installing computational tools often discourage chemistry educators from incorporating programming exercises into their courses. To tackle these challenges, we developed a preconfigured image on Jetstream 2, a cloud computing environment using OpenStack infrastructure. This image, shared with the community, allows chemistry instructors to effortlessly deploy a JupyterHub platform for their classrooms, facilitating the teaching of programming skills. Integrated with the automatic grading package nbgrader, the JupyterHub website created from this image enables seamless assignment creation, distribution, and automatic grading on a cloud-based platform. Students can access assignments directly through their web browsers without the need

to install software or configure their local machines. This tool empowers educators to equip future scientists with essential coding skills, enabling them to tackle interdisciplinary challenges and drive chemical discoveries forward.

Introduction

Data-driven research and machine learning techniques are revolutionizing chemical discovery by enabling breakthroughs in areas like materials designs,[1, 2] reaction optimizations,[3, 4] and property predictions.[5, 6] As a result, programming skills have become essential for chemistry students and researchers nowadays.[7-10] However, the integration of coding skills into undergraduate or graduate chemistry curricula is significantly hindered by the need for specialized software and hardware dependencies. These technical requirements often present logistical challenges, such as the need for extensive configuration, compatibility issues, and demanding grading due to variations in computing environments. As a result, many instructors either provide students with precomputed data to plot in widely available software,[11, 12] bypassing the programming component entirely, or rely on institution-specific licensed software packages, making the exercise hard to be adopted by other schools.[12-14]

To address this challenge, we developed a solution that enables chemistry instructors to integrate coding assignments into their curriculum seamlessly. Our approach leverages nbgrader[15] and JupyterHub[16] on the Jetstream 2 cloud platform.[17, 18] nbgrader streamlines assignments' creation, distribution, and grading. JupyterHub provides students with a browser-based coding environment that requires no local setup. With Jetstream 2's cloud-based infrastructure freely available to the community, these tools provide a reliable platform for instructors to incorporate programming assignments into their courses. Our workflow is designed to offer a streamlined approach that

minimizes the steps required for setting up a customized JupyterHub website for a classroom. Educators can foster a more accessible and effective learning environment to prepare the next-generation scientists by utilizing our solution.

Software Description

nbgrader and JupyterHub are powerful tools that have transformed the management of computational assignments. JupyterHub provides a multi-user platform where students can access Jupyter Notebooks,[19] an interactive computing environment, directly through a web browser. This setup eliminates the need for local software installations and ensures a consistent coding environment for all students within the same class. Meanwhile, nbgrader enables educators to automatically create, distribute, and grade assignments in the Jupyter Notebook interface. Since the Jupyter system supports over 100 programming languages (referred to as "kernels" in the Jupyter ecosystem), including Python, Java, R, Julia, and Matlab,[20] a JupyterHub integrated with nbgrader can be used to teach various coding skills related to chemical sciences. For example, it can be used to teach machine learning and data analysis with Python, biostatistics with R, quantum chemistry and molecular simulation code development with C++ or Fortran, and scientific plotting with Matlab or Python. Additionally, JupyterHub's uniform computing environment and nbgrader's automated grading system ensure fair, consistent, and transparent assessment practices across the class.

However, deploying a JupyterHub instance with the graphical interface (GUI) of ngbrader has become increasingly challenging for instructors. Since Formgrader, the GUI of nbgrader, is no longer compatible with the Littlest JupyterHub (TLJH) following the latter's recent update to version 4.2.6., instructors must either downgrade to an earlier version of JupyterHub or explore

alternative workarounds. Additionally, setting up JupyterHub on a cloud server involves multiple intricate steps crucial to a successful deployment. These challenges can be overwhelming for chemistry educators who may lack expertise in cloud computing and are hesitant to invest additional time in system configuration.

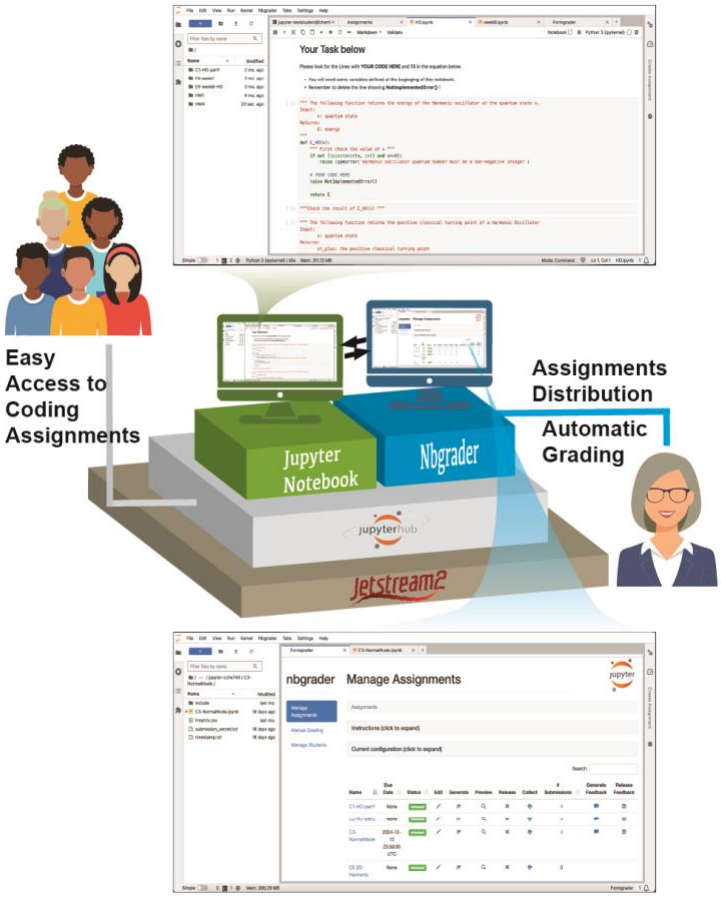

Figure 1. nbgrader integrated JupyterHub allows professors to distribute coding assignments with automatic grading. Students can easily access their homework and grading reports through a web browser.

Some engineering educators have previously reported deploying JupyterHub with nbgrader on their university-owned, cloud-based Kubernetes nodes.[21] Their deployment followed the "Zero to JupyterHub with Kubernetes"[22, 23] and required either an existing Kubernetes container orchestration platform or setting up Kubernetes on a cloud computing system, which can be

technically challenging for educators unfamiliar with Kubernetes. In contrast, we sought an even simpler solution that provides shareable resources, enabling other chemistry educators to deploy a JupyterHub instance integrated with nbgrader effortlessly.

Preconfigured JupyterHub Image

We chose to utilize a widely used technique in software development that leverages a preconfigured image to address these challenges. In software development, an image refers to a preconfigured snapshot of an environment that includes all necessary software, dependencies, and settings to perform specific tasks.[24] It serves as a blueprint for creating reproducible systems, ensuring that applications and workflows run consistently across various platforms. We aim to provide instructors with a preconfigured image that eliminates compatibility issues and simplifies the setup process required for deploying a JupyterHub with nbgrader. To achieve this goal, we created an image from a fully operational JupyterHub instance integrated with nbgrader (Figure 1), which was initially deployed in December 2023 for a class before the compatibility issue between Formgrader and TLJH emerged. The original JupyterHub instance and the newly created image both reside in Jetstream 2,[18] a user-friendly cloud computing environment based on the OpenStack[25] infrastructure. Jetstream 2 provides researchers and educators with zero-cost, always-on infrastructure through the ACCESS[17] ecosystem, supported by the National Science Foundation. Jetstream 2 provides efficient instance management through its web interface,[26] allowing users to seamlessly create, share, and deploy community images. Our preconfigured image, shared with the JetStream 2 community, enables instructors to set up a JupyterHub with nbgrader in just a few clicks. While additional steps -- such as renaming the course, managing student access, and configuring permissions -- are required to tailor the JupyterHub for individual

classes, the customization process is straightforward and guided by the instructions provided in the following sections.

Setting Up a JupyterHUB Instance Using the Image

To set up an instance using our image, the course instructor should first create an ACCESS[17] account following the instructions [27] and submit an application for a project on Jetstream 2.[18] It is worth noting that the ACCESS project application requires the users to choose from four project types depending on the number of computing credits requested. Based on our experience, the smallest "Explore" project is enough for hosting the JupyterHub for a small class of less than 20 students. The demand for computing credits increases with the class size because more CPUs are needed to host the JpyterHub to ensure a smooth computing experience for concurrent users of the platform.

Once a Jetstream 2 allocation is available, the instructor can log into the portal and locate the "TLJH-nbgrader-image" in the community images section on the allocation page. The educator can create a cloud computing instance using the image. Clicking the 'Create Instance' button associated with our image will allow the instructor to customize the instance configuration by selecting a preconfigured option under the 'Flavor' menu (SI Figure S1). Since our image requires a minimum root disk size of 60GB, we recommend choosing a configuration with at least 60GB of root disk space. Once the configuration is specified, a fully functional deployment of The Littlest JupyterHub (TLJH) with nbgrader preinstalled will be set up upon confirming the creation.

Secure communication for the newly created JupyterHub instance is established by enabling Hypertext Transfer Protocol Secure (HTTPS) using Let's Encrypt[28]. This can be achieved by configuring the newly created instance via the web shell. The instructor can access the web shell

by selecting the 'Web Shell' button under the 'Interactions' section (SI Figure S2). Running the commands provided in the script shown in Box 1 will successfully set up an encrypted communication for the instance.

```
# Initialize HTTPS

sudo tljh-config set https.enabled true

# Configure HTTPS

sudo tljh-config set https.letsencrypt.email <you@example.com>

sudo tljh-config add-item https.letsencrypt.domains <your JupyterHub's public IP>

# Load the New Configuration

sudo tljh-config reload proxy
```

Box 1. Scripts to be typed into the web shell to establish secure communication for the newly created instance. Successful configuration requires registering with a valid email address and the instance's hostname, which can be found under the 'Credential' section on the allocation page. Please note that JupyterHub's public IP address is automatically generated by the Jetstream 2 platform and can be found on the Jetstream 2 project's page under Credentials → Hostname (an example screen snapshot is available in SI Figure S2).

Since the instance is created using a preconfigured image, an administrator account must be added before accessing the JupyterHub interface. Thus, the instructor needs to continue the setup process in the web shell. After following the commands in Box 2, the JupyterHub interface can be accessed through the instance's public IP address shown in the 'Credentials' section on the Jetstream 2 allocation page (SI Figure S2). Entering the IP address via a web browser will lead to the login page for JupyterHub (SI Figure S4). During the initial login, authentication can be performed with any chosen password.

```
# Add Administration Account

sudo tljh-config add-item users.admin <your-username>

# Load the New Configuration

sudo tljh-config reload proxy
```

Box 2. Scripts to add an administration account to the newly established JupyterHub instance. After completing this step, the web shell can be closed.

The nbgrader environment must be configured within the JupyterHub instance to enable communication between instructors and students (SI Figure S5). To achieve this, the instructor must log into the JupyterHub and open the terminal window in the main menu. Following the commands shown in Box 3 will complete the initial setup. The instructor needs to create a configuration file under their home directory (`~/.jupyter/nbgrader_config.py`), and the content of the file is provided in Box 4. This step can be finished by any Linux text editor, such as VI (detailed instructions available in SI Figure S6).[29] Once the file is successfully created, the instructor should run the commands in Box 5 in the terminal. This last step ensures the correct nbgrader configuration settings are carried out every time JupyterHub launches.

```
# Quick Setup of nbgrader (Replace <your_course_name> With Desired Name)
nbgrader quickstart <your_course_name>
# Setup File Exchange Between Administrator and Students
rm -rf /tmp/exchange
mkdir /tmp/exchange
chmod ugo+rw /tmp/exchange
```

Box 3. Scripts for the initial setup for the nbgrader in the newly configured JupyterHub instance.

```
# Configuration file for file exchange
c = get_config()
c.CourseDirectory.course_id = "your_course_name"
c.Exchange.root = "/tmp/exchange"
```

Box 4. nbgrader configuration file content. Replace "your_course_name" with the same course name chosen in Box 3.

```
# Copy the nbgrader configuration to global setting

sudo mkdir -p /usr/local/etc/jupyter/
sudo cp .jupyter/nbgrader_config.py
/usr/local/etc/jupyter/nbgrader_config.py
```

Box 5. Scripts for copying the nbgrader configuration file to the global setting.

Lastly, access restrictions in nbgrader need to be implemented to prevent students from accessing grading information. This is achieved by executing the commands in Box 6 into the terminal in the main menu. Then, the instructor should navigate to "File→ Hub Control Panel" to change some settings of the JupyterHub (SI Figure S7. On the Hub Control Panel page, the instructor should click "Admin" on the toolbar to bring up the list of users of the JupyterHub and delete the user "mockinstructor" (SI Figure S7). Then, the instructor should restart the JupyterHub by navigating back to the "Home" page of the Hub Control Panel, clicking "Stop My sever" and then "Start My server" (SI Figure S8).

Once these steps are completed, a fully functional JupyterHub with nbgrader is established. The instructor can validate this by clicking "nbgrader→ formgrader" on the toolbar of JupyterHub (SI Figure S9). This action should take the instructor to Formgrader, the graphical interface of nbgrader. The Formgrader page is expected to display a list of existing assignments (Figure 2). For a new JupyterHub site, this list will be empty

```
# Applying restrictions such that student cannot access grading information

sudo jupyter server extension disable nbgrader.server_extensions.formgrader

sudo jupyter labextension disable @jupyter/nbgrader:formgrader

jupyter server extension enable --user --py
nbgrader.server_extensions.formgrader

jupyter labextension enable --level=user @jupyter/nbgrader:formgrader
```

Box 6. Scripts for applying restrictions to the student accounts to prevent them from accessing the grading information.

Implication in Classroom

In practice, this platform empowers instructors to design interactive coding exercises that enhance student's understanding of chemistry concepts while practicing their coding skills. For example, here, we demonstrate how to create a coding assignment in a quantum chemistry class focusing on training the students to write a function to calculate the energy of a harmonic oscillator at a specific quantum state (Jupyter notebook included in SI).

On the Formgrader page of JupyterHub (Figure 2), the instructor can create a new assignment by clicking "Add new assignment" at the bottom of the assignment list (Figure 2). This will result in the creation of a new folder "/[yourCourseID]/source/[AssignmentName]", where the instructor can create the instructor-version of Jupyter notebooks. For example, the instructor can upload our example instructor-version Jupyter notebook, "HO.ipynb", provided in the SI. The instructor version includes some regular markdown cells for instructions, some nbgrader-specific coding cells to be filled by the students, and some nbgrader-specific testing cells that automatically run tests on the functions coded by students to assign grades (SI Figure S10). It is worth noting that the instructor-version of the Jupyter notebook should contain the completed code that can pass all the embedded tests, even for the parts to be filled by the students, which are labeled between the comment lines, "### BEGIN SOLUTION" and "### END SOLUTION" (see Figure 3). Detailed instructions about creating Jupyter Notebook assignments are available in the nbgrader user manual.

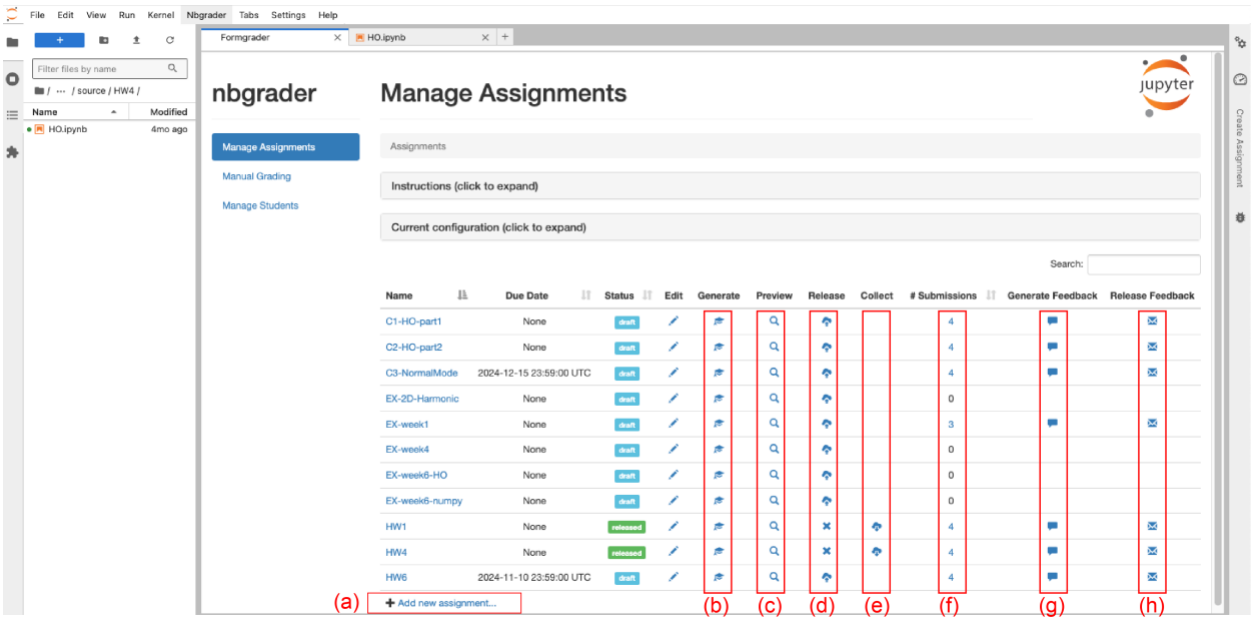

Figure 2. Illustration of Forgrader interface on the JupyterHub for Chem 531 classroom. (a) The button to add a new assignment. (b)The icon to generate the student version of the coding assignment. (c) The icon to preview the student version assignment. (d) The icon to release the assignment. (e) The icon to collect submissions is only available after an assignment is released. (f) The icon to navigate individual submissions. (g) The icon to generate automatic feedback. (h) The icon to release feedback.

Once the Jupyter Notebook for an assignment is created, the instructor can navigate back to Formgrader and create the student version of the Jupyter notebook by clicking the "Generate" icon belonging to that assignment (Figure 2). The generated student-version notebook can be viewed by clicking the "Preview" icon (Figure 2) and is expected to have the same contents as the instructor version, except for the removal of code blocks between the comment lines, "### BEGIN SOLUTION" and "### END SOLUTION" (Figure 3). The instructor can then release the assignment to the students by clicking the "Release" icon (Figure 2). Students can then fetch the assignment by navigating to "nbgrader → Assignment List" (SI Figure S11), start to fill in codes in the student version of the Jupyter notebook, save their changes, and finally submit the assignment from the same Assignment List interface (SI Figure S11). After the assignment is due, the instructor can collect the submissions on the Formgrader interface by clicking the "collect"

icon, which is available only for already-released assignments (Figure 2). To grade the collected submission, the instructor can first click on the submissions to get to a new page showing the list of all submissions, and then click the "Autograde" button for each submission (SI Figure S12). Finally, feedback on the coding assignment can be generated and released to the student by clicking the "Generate feedback" and "Release feedback" buttons in the Formgrader list (Figure 2). This approach enhances students' understanding of quantum mechanical concepts while introducing coding in an engaging manner.

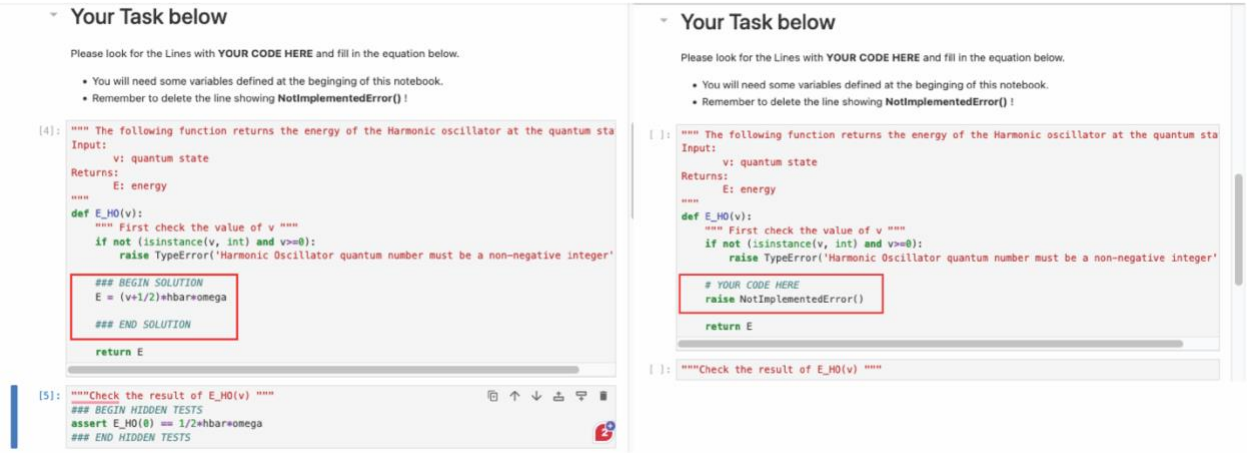

Figure 3. Comparison of the instructor version (left) and student version (right) of the same code block. For each version, we demonstrate one cell for coding exercise and one cell for testing the code filled in the previous cell. For the first cell, the code between the comment lines, "### BEGIN SOLUTION" and "### END SOLUTION", in the instructor version (marked by the red rectangle) is automatically removed when the student version is generated. For the second cell, the test code in the instructor version is automatically hidden in the student version. The full instructor-version and student-version notebooks are included in the SI Data.

Result

A survey was conducted in a small graduate-level quantum chemistry class that utilized the JupyterHub platform integrated with nbgrader over a semester. Although the survey responses were limited due to the small class size, the feedback provides valuable insights into the platform's impact on student learning, coding proficiency, and overall usability.

The platform proved effective in creating a supportive environment for coding-based exercises. Most students reported feeling comfortable engaging with coding assignments, with 50% indicating they felt "Very Comfortable" and 25% "Somewhat Comfortable". This suggests that the platform successfully lowers the barriers to incorporating coding into coursework, even for students with limited prior experience.

Table 1 Students' Perception of Coding Assignments in Chemistry Course

Questions for students to respond		Response by Score (Total N=4)				
On a scale from 1 to 5, how comfortable did you feel when approaching the coding assignments in this course?	Score	5	4	3	2	1
	Count	2	1	1	0	0

In addition to democratizing programming, 75% of respondents felt that using the platform enhanced their learning experience, particularly in understanding and applying coding concepts to scientific problems.

Table 2 Students' Feedback on Utilizing JupyterHub Integrated with nbgrader to Facilitate Course Material

Questions for students to respond		Response by Score (Total N=4)				
On a scale from 1 to 5, do you feel that using JupyterHub enhanced your learning experience?	Score	5	4	3	2	1
	Count	1	2	1	0	0

Notably, all students recommended JupyterHub for future classes. This showcased strong support for its integration into the curriculum. This feedback underscores the platform's value in teaching coding skills alongside scientific concepts and effectively preparing students for future research endeavors.

Table 3 Students' Outlook on Applying Similar Technology to Future Courses

Questions for students to respond	Response (Total N=4)	
Would you recommend JupyterHub for future classes?	Yes	No
	4	0

Discussion and Conclusion

Integrating coding assignments into the chemistry curriculum is not new. However, achieving it in a seamless and consistent manner remains a challenge in chemistry education. Our approach simplifies the technical hurdles in incorporating coding exercises into the curriculum through a preconfigured image for JupyterHub and nbgrader. Additionally, this tool removes many of the frustrations commonly associated with coding assignments for students.

Combining accessibility with functionality, this preconfigured JupyterHub image represents a valuable resource for integrating computational tools into chemistry education. It addresses the logistical challenges faced by instructors while simultaneously fostering a positive learning experience for students. Such tools will play a critical role in preparing students to meet the evolving demands of modern chemical research.

Supporting Information

Step-by-step directions on creating a secure instance of JupyterHub on the Jetstream 2 platform; Instructions on accessing the JupyterHub through the website; Setup guide for nbgrader in the JupyterHub; Directions for removing mockinstructor account and restarting the server to complete the JupyterHub setup; Formgrader interface introduction; Demonstrations of the various cell types in an instructor's Jupyter notebook in nbgrader; Visual tutorial on accessing assignments and setting up automatic grading in nbgrader. (PDF)

An example course assignment on harmonic oscillator created for the nbgrader-JupyterHub platform, with the instructor-version Jupyter notebook, student-version Jupyter notebook, and the automatically generated feedback web page. (ZIP)

Corresponding Author

*Email: fang.liu@emory.edu

Acknowledgement

Financial support for this project comes from Cottrell Scholar Award #CS-CSA-2024-099 sponsored by Research Corporation for Science Advancement. This work used JetStream2 at Indiana University through allocation CHE230134 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

Reference:

(1) Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. Machine learning for molecular and materials science. Nature 2018, 559 (7715), 547-555. DOI: 10.1038/s41586-018-0337-2.

(2) Griesemer, S. D.; Xia, Y.; Wolverton, C. Accelerating the prediction of stable materials with machine learning. Nature Computational Science 2023, 3 (11), 934-945. DOI: 10.1038/s43588-023-00536-w.

(3) Taylor, C. J.; Pomberger, A.; Felton, K. C.; Grainger, R.; Barecka, M.; Chamberlain, T. W.; Bourne, R. A.; Johnson, C. N.; Lapkin, A. A. A Brief Introduction to Chemical Reaction Optimization. Chemical Reviews 2023, 123 (6), 3089-3126. DOI: 10.1021/acs.chemrev.2c00798.

(4) Zhou, Z.; Li, X.; Zare, R. N. Optimizing Chemical Reactions with Deep Reinforcement Learning. ACS Central Science 2017, 3 (12), 1337-1344. DOI: 10.1021/acscentsci.7b00492.

(5) Goodall, R. E. A.; Lee, A. A. Predicting materials properties without crystal structure: deep representation learning from stoichiometry. Nature Communications 2020, 11 (1), 6280. DOI: 10.1038/s41467-020-19964-7.

(6) Heid, E.; Greenman, K. P.; Chung, Y.; Li, S.-C.; Graff, D. E.; Vermeire, F. H.; Wu, H.; Green, W. H.; McGill, C. J. Chemprop: A Machine Learning Package for Chemical Property Prediction. Journal of Chemical Information and Modeling 2024, 64 (1), 9-17. DOI: 10.1021/acs.jcim.3c01250.

(7) Charles, J. W. Perspectives: Teaching chemists to code. C&EN Global Enterprise 2017, 95 (35), 30-31. DOI: 10.1021/cen-09535-scitech2 (acccessed 2023).

(8) Cropper, C. Why should chemistry students learn to code? In RSC Education, Royal Society of Chemistry: 2017.

(9) Bazargan, G. Up to code. In Chemistry World, Royal Society of Chemistry: 2021.

(10) Ringer McDonald, A. Teaching Programming across the Chemistry Curriculum: A Revolution or a Revival? In Teaching Programming across the Chemistry Curriculum, ACS Symposium Series, Vol. 1387; American Chemical Society, 2021; pp 1-11.

(11) Martini, S. R.; Hartzell, C. J. Integrating Computational Chemistry into a Course in Classical Thermodynamics. Journal of Chemical Education 2015, 92 (7), 1201-1203. DOI: 10.1021/ed500924u.

(12) Esselman, B. J.; Ellison, A. J.; Hill, N. J. Using Computational Chemistry to Rationalize the Diastereoselectivity of the Borohydride Reduction of Benzoin. Journal of Chemical Education 2022, 99 (11), 3757-3764. DOI: 10.1021/acs.jchemed.2c00828.

(13) Metz, I. K.; Bennett, J. W.; Mason, S. E. Examining the Aufbau Principle and Ionization Energies: A Computational Chemistry Exercise for the Introductory Level. Journal of Chemical Education 2021, 98 (12), 4017-4025. DOI: 10.1021/acs.jchemed.1c00700.

(14) Snyder, H. D.; Kucukkal, T. G. Computational Chemistry Activities with Avogadro and ORCA. Journal of Chemical Education 2021, 98 (4), 1335-1341. DOI: 10.1021/acs.jchemed.0c00959.

(15) Jupyter, P.; , D. B.; , D. B.; , A. B.; ; Bussonnier, M.; , J. F.; , B. G.; , T. L.; Griffiths; et al. nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook. 2019, 2 (16), 32. DOI: 10.21105/jose.00032.

(16) JupyterHub Team. JupyterHub: A multi-user server for Jupyter notebooks. 2023. https://jupyterhub.readthedocs.io/en/stable/ (accessed 2/22/2025).

(17) Boerner, T. J.; Deems, S.; Furlani, T. R.; Knuth, S. L.; Towns, J. ACCESS: Advancing Innovation: NSF's Advanced Cyberinfrastructure Coordination Ecosystem: Services \& Support. In Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good, 2023.

(18) Jetstream2: Accelerating cloud computing via Jetstream. In PEARC 2021 - Practice and Experience in Advanced Research Computing 2021 ,, 2021.

(19) Thomas Kluyver, B. R.-K., Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, Jupyter Development Team. Jupyter Notebooks – a publishing format for reproducible computational workflows; 2016. DOI: 10.3233/978-1-61499-649-1-87.

(20) Barba, L.; Barker, L.; Blank, D.; Brown, J.; Downey, A.; George, T.; Heagy, L.; Mandli, K.; Moore, J.; Lippert, D.; et al. Teaching and Learning with Jupyter; 2019.

(21) Smet, R. D.; Thielemans, S.; Lemeire, J.; Braeken, A.; Steenhaut, K. Educational software-as-a-service based on JupyterHub and nbgrader running on Kubernetes. In 2022 IEEE 9th International Conference on e-Learning in Industrial Electronics (ICELIE), 17-20 Oct. 2022, 2022; pp 1-6. DOI: 10.1109/ICELIE55228.2022.9969419.

(22) Wu, A.; Mar, D.; Gong, J.; Veerman, P.; Lovett, R.; Lau, S.; Panda, Y. Zero to JupyterHub with Kubernetes. 2016. https://z2jh.jupyter.org/en/stable/ (accessed 2/22/2025).

(23) zero-to-jupyterhub-k8s; 2016. https://github.com/jupyterhub/zero-to-jupyterhub-k8s/tree/main (accessed 2/22/2025).

(24) Computer Imaging. 2019. https://www.intel.com/content/www/us/en/business/enterprise-computers/resources/computer-imaging.html?utm_source=chatgpt.com (accessed 2/22/2025).

(25) Sefraoui, O.; Aissaoui, M.; Eleuldj, M. OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications 2012, 55 (3).

(26) Instance Management Actions - Jetstream2 Documentation. 2024. https://docs.jetstream-cloud.org/general/instancemgt/#image (accessed 2/22/2025).

(27) ACCESS User Registration | Operations. https://operations.access-ci.org/identity/new-user (accessed 2/22/2025).

(28) Internet Security Research Group (ISRG). Let's Encrypt. https://letsencrypt.org/ (accessed 01/31/2025).

(29) The Open, G. The Open Group Base Specifications Issue 7. IEEE Std 1003.1-2017 2018.