Appendix E

Visual Studio Code

Visual Studio Code (VSCode) is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It has a rich ecosystem of extensions, including remote development and intellisens for C and fortran. Although you are in no way required to use it (other text editers like gedit, vim or notepad work perfectly fine), it may help you speed up your workflow significantly. VSCode can be downloaded from

```
https://code.visualstudio.com/
```

Note: When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily open a folder in WSL using the `code` command (see Figure E.1).

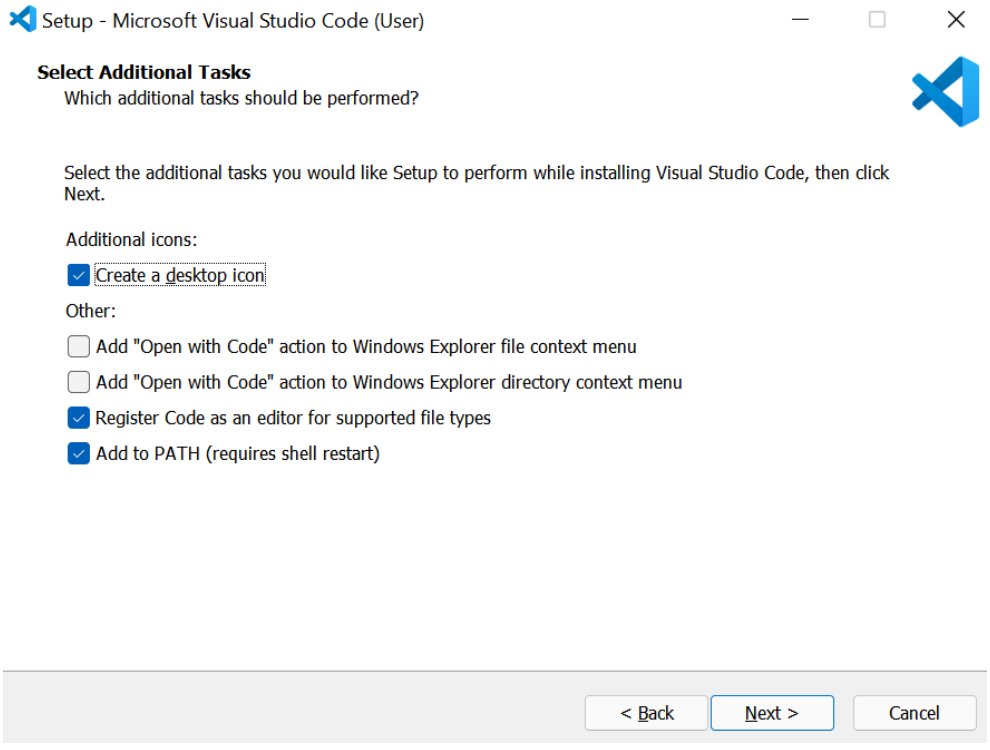

Figure E.1: Adding to path during installation.

VS Code is first and foremost an editor, and relies on command-line tools to do much of the development workflow. There is basic support for html, css, javascript and typescript out of the box. Support for other languages is supported via extensions. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. VS Code does not include a C/C++ or Fortran compiler or debugger. You will need to install these tools (see previous appendices). Extensions can be installed

- via the browser marketplace at

  `https://marketplace.visualstudio.com/`

  (it will open VSCode)

- through the Extensions tab on the left side of the screen in the 'activity bar' (see Figure E.2)

- or via the shortcut `Ctrl-Shift-X`.

We recommend to install extensions like `C/C++ IntelliSense, debugging, and code browsing` (IntelliSense for C or Fortran) and `Code Runner` (allows execution of single files), but feel free to look around for other functionality you might like.

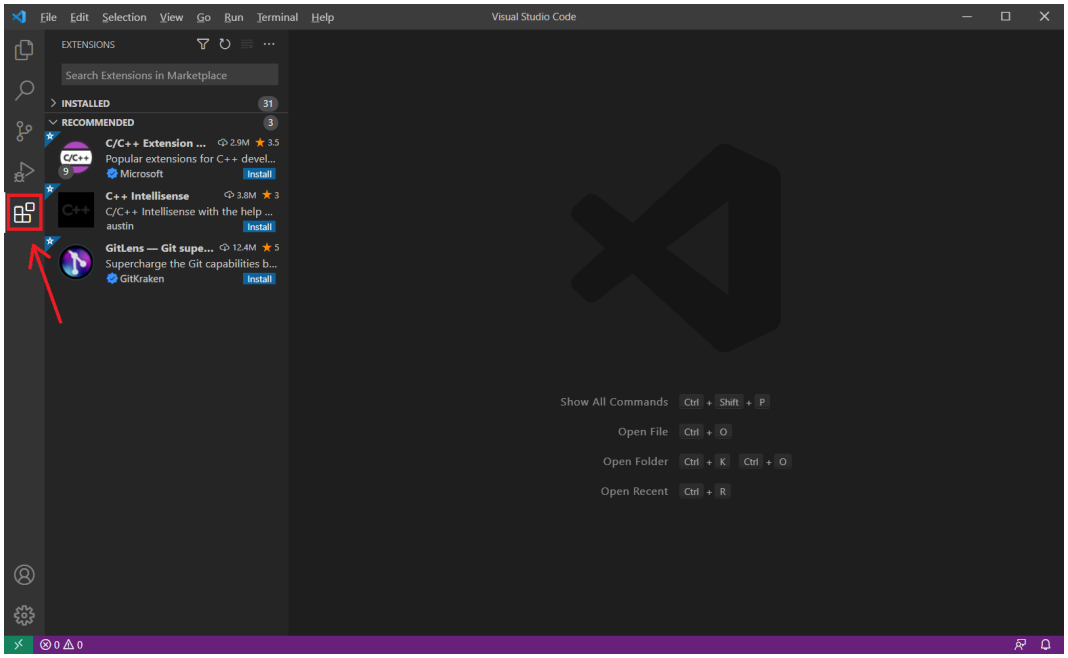

Figure E.2: Extensions tab in VSCode.

Once installed, Mac and Linux users can immediately start using VSCode by opening a folder using either the `code` command or opening a folder from inside VSCode:

- By navigation to File → Open Folder

- By pressing `Ctrl-K` followed by `Ctrl-O`. (Pressing `Ctrl-O` by itself opens a file.)

The `Visual Studio Code Remote - WSL` extension lets you use the Windows Subsystem for Linux (WSL) as your full-time development environment right from VS Code. You can develop in a Linux-based environment, use Linux-specific toolchains and utilities, and run and debug your Linux-based applications all from the comfort of Windows. The extension runs commands and other extensions directly in WSL so you can edit files located in WSL or the mounted Windows filesystem (for example /mnt/c) without worrying about pathing issues, binary compatibility, or other cross-OS challenges. Windows users (who are working on WSL) can install the Remote Development extension pack in VSCode through the link

```
https://marketplace.visualstudio.com/items?itemName=
ms-vscode-remote.vscode-remote-extensionpack
```

or the Extensions tab (see Figure E.3). After installing, you can open your project in VSCode by navigating to the correct folder in your Ubuntu terminal and then running

```
code .
```

This will open VSCode in your windows environment, login to your WSL and open the current folder (assuming you have added code to your path as specified before).

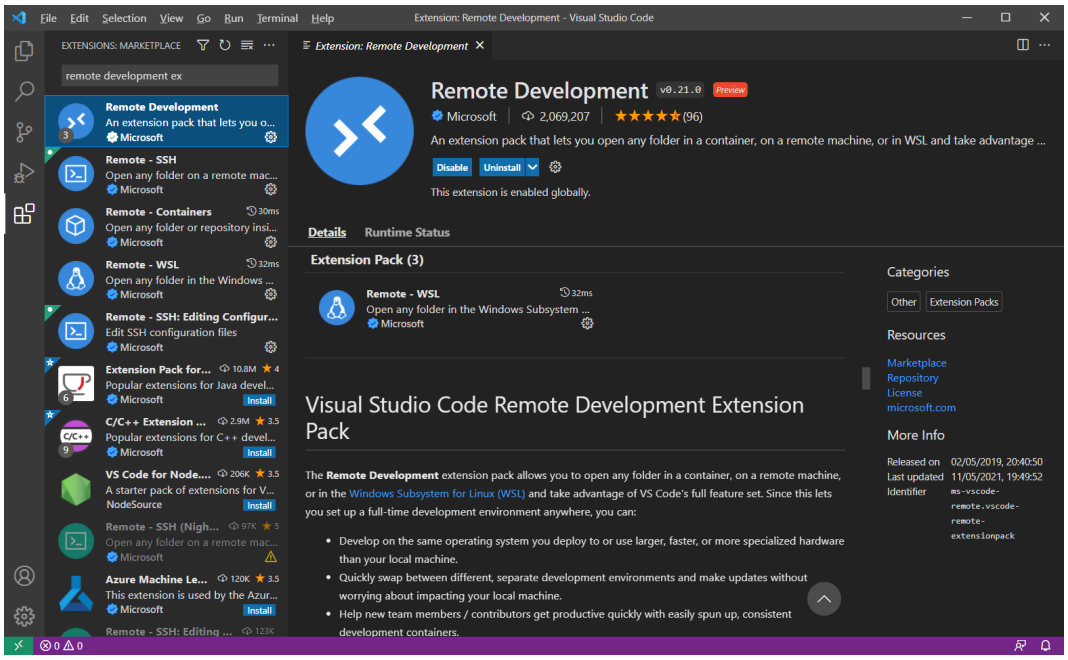

Figure E.3: Remote extension pack in VSCode.

Once you have opened your folder, you can navigate through it on the left side of the window. Clicking a file will open it in the editor to the right of the file explorer (see Figure E.4). A typical workflow will include first editing a source file (like `distribution.c` in the example), saving it (`Ctrl-S`) and then running `make` in the source directory. You can quickly open this directory in the terminal (or any other for that matter) by right clicking the folder and then `Open in integrated terminal` (see Figure E.5).

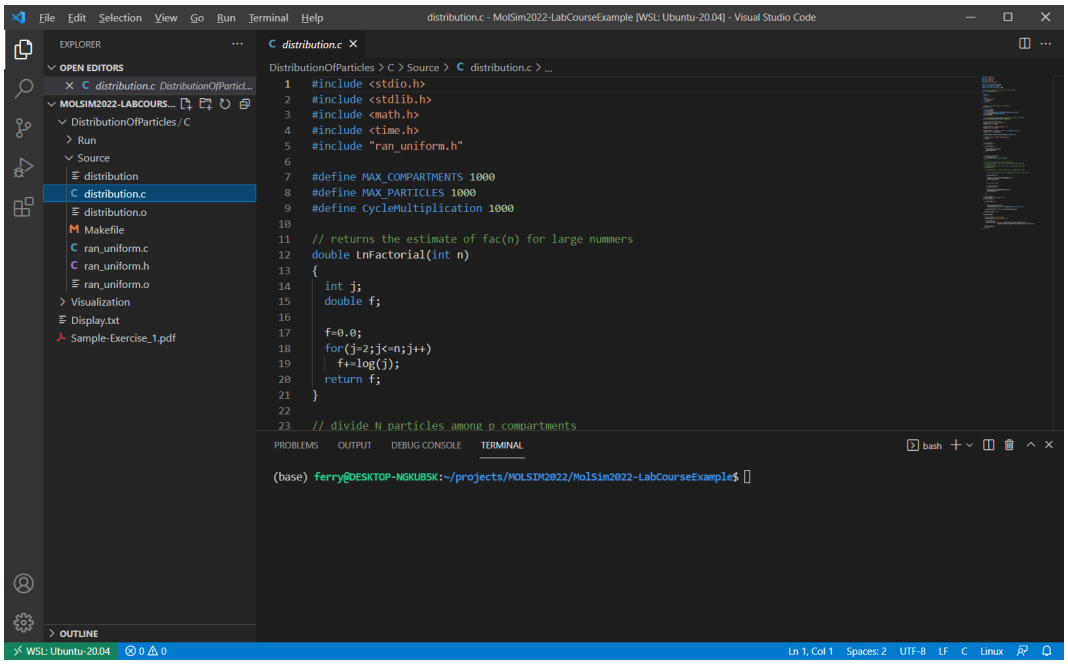

Figure E.4: Opening a file from the opened folder in VSCode.

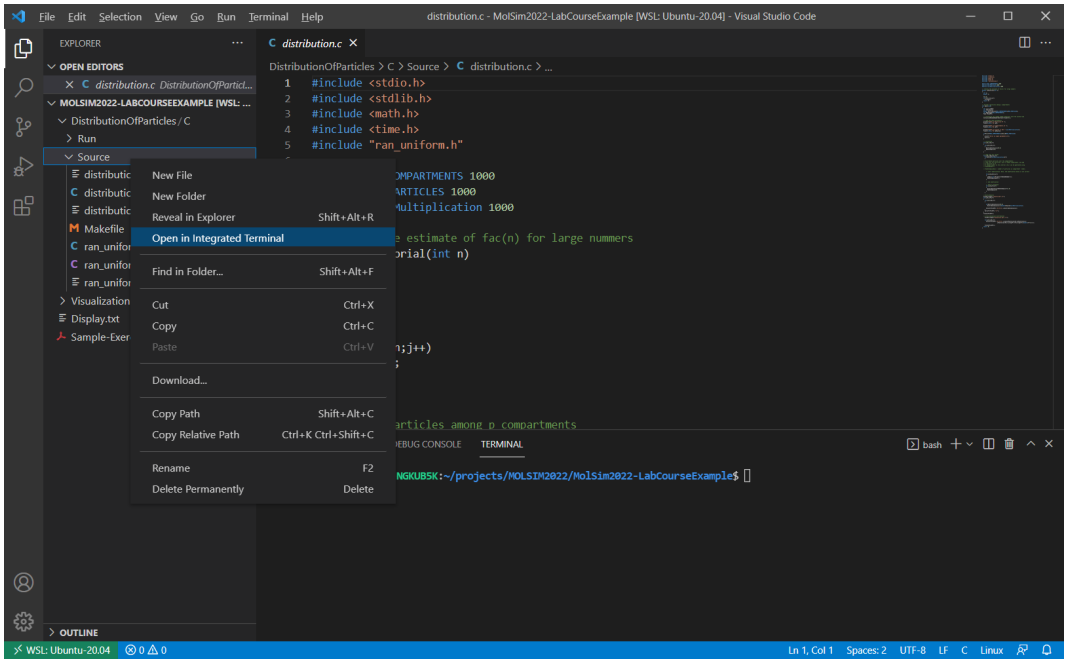

Figure E.5: Open a folder in the terminal.

By default, VSCode does not run your `.bashrc` or `.bash_profile` on opening a terminal. In order to get this functionality, you have to configure the terminal settings by:

- Opening a new terminal (Terminal → New Terminal or Ctrl+Shift+`).
- Clicking on the dropdown menu next to the plus sign on the bottom right of the screen (above the terminal; see Figure E.6).
- Search for `linux` and find the Terminal>Integrated>Env:Linux header (see Figure E.7). We need to edit the linux settings because we are logging into a linux (WSL) environment.
- Open the settings by clicking on `edit in settings.json`. This should place your cursor beneath the line

  ```
  "terminal.integrated.profiles.linux":
  ```

  inside the curly brackets. If not, you can try closing and opening the settings again or manually find the line to edit. See Figure E.8.
- Add the lines of code present in Figure E.8 below

  ```
  "terminal.integrated.profiles.linux":
  ```

  These will make it so that the bash command is run with the -l argument, loading your `.bashrc` and `.bash_profile`. It is also recommended to set the default terminal to bash:

  ```
  "terminal.integrated.default.profile.linux": "bash"
  ```

  Note that the editor in the figure might have more or other settings than you have. You can ignore this.

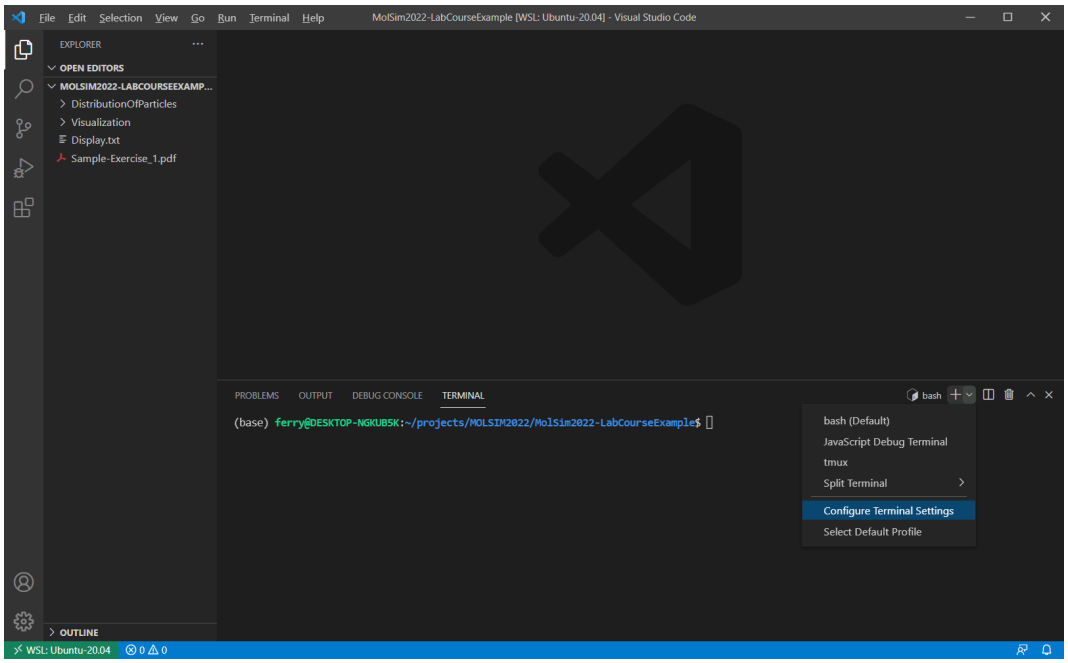

Figure E.6: Opening the terminal configuration settings.

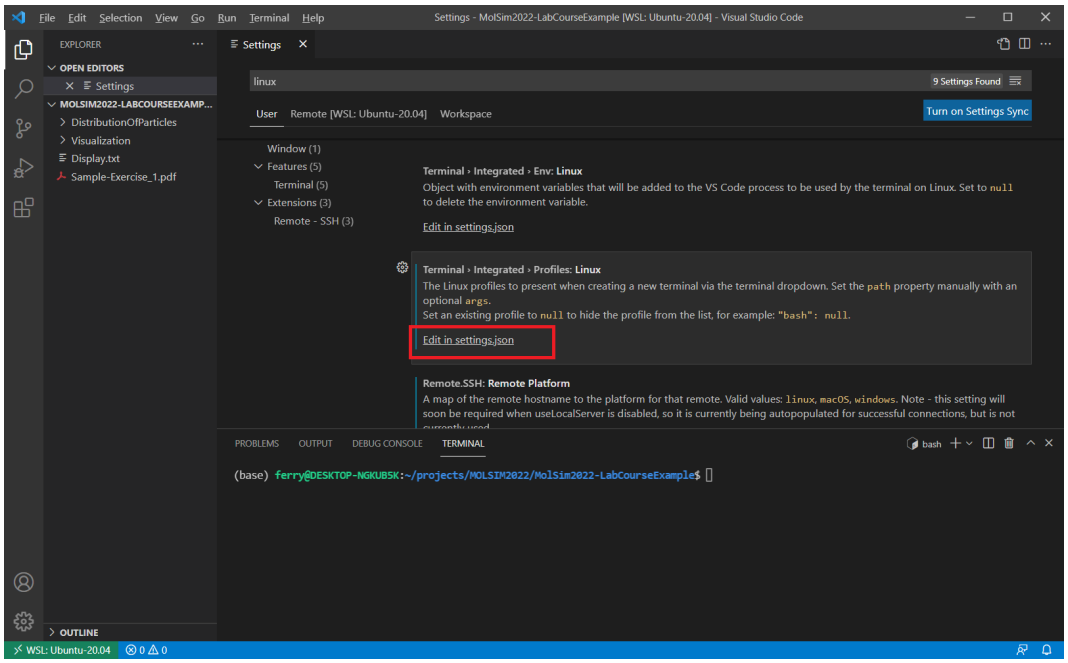

Figure E.7: Opening settings.json.

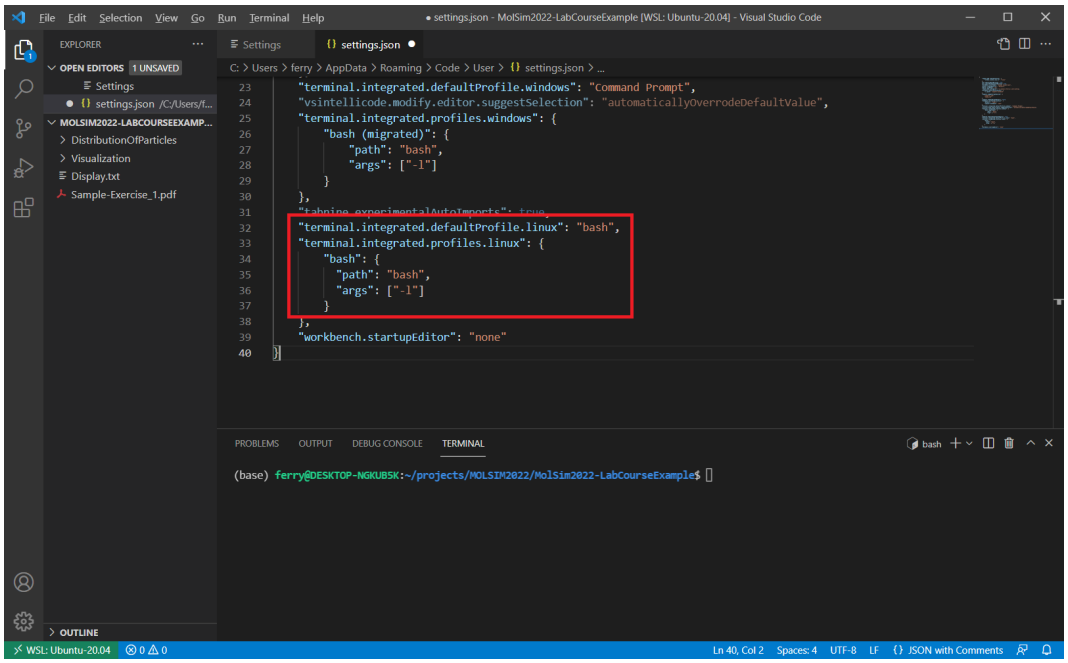

Figure E.8: Configuring the terminal settings in settings.json.