Tutorial 1 — First Co-Simulation

Overview

This INTO-CPS tutorial will show you how to:

1. Set up a co-simulation project in Visual Studio Code
2. Run a co-simulation from within Visual Studio Code

Requirements

The tutorial assumes that you have the following pieces of software installed:

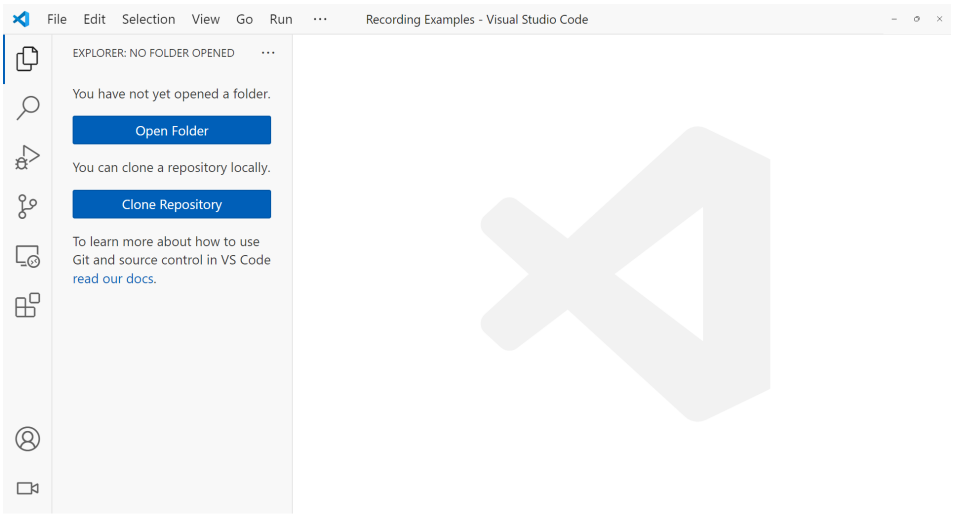
- Visual Studio Code
- `Cosimulation Studio` extension for Visual Studio Code
- Java SE Runtime Environment 11, newer versions may also work
- `Maestro` COE (Co-simulation Orchestration Engine) w. Web API

Follow Tutorial 0 if any of these are missing from your system.

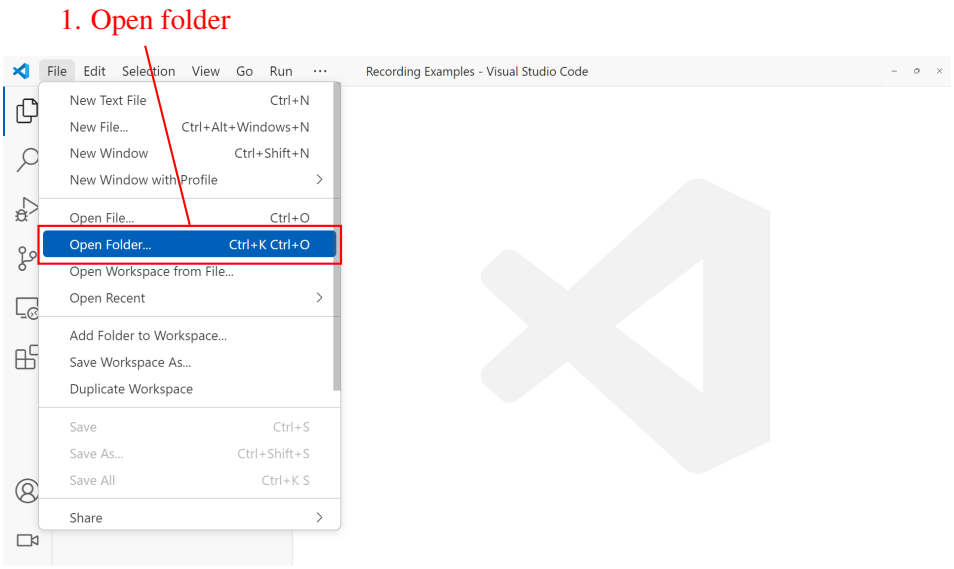
1 Set up a co-simulation project in Visual Studio Code

Any workspace folder containing a `cosim.json` simulation configuration file is automatically treated as a co-simulation project, enabling features such as autocompletion, linting and integration with the `Maestro` COE. Setting up a project is thus as simple as opening a folder in Visual Studio Code and writing a `cosim.json` file. This section of the tutorial describes this process.

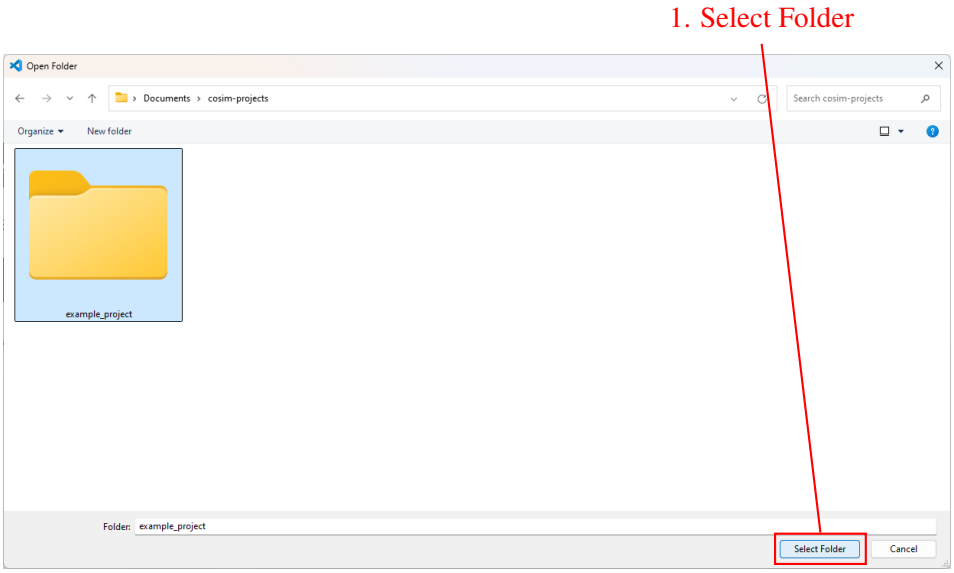
Step 1. Launch Visual Studio Code. It is not important if the view is empty as the one below or if Visual Studio Code opens up an existing project.



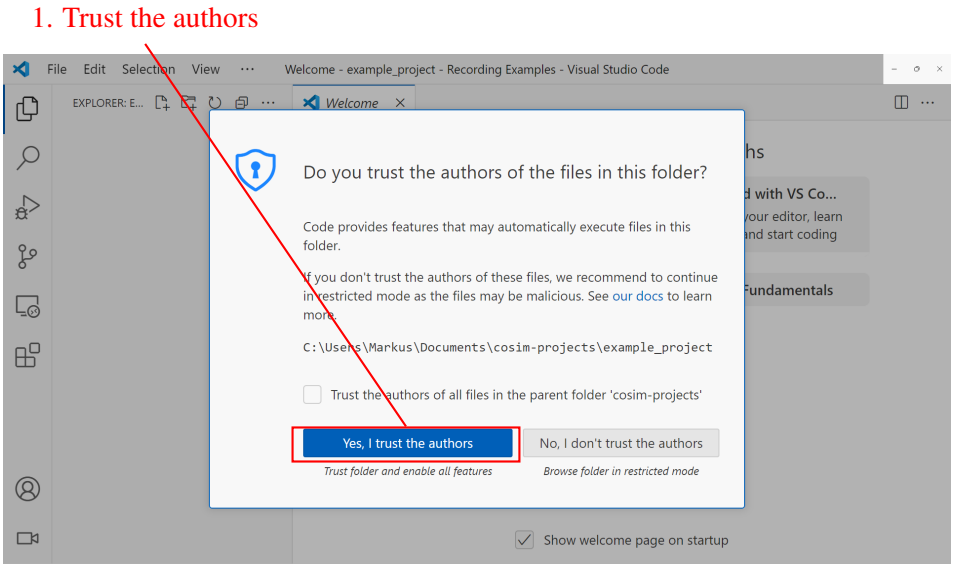
Step 2. To open the folder that will contain the project, select *File* > *Open folder...*.



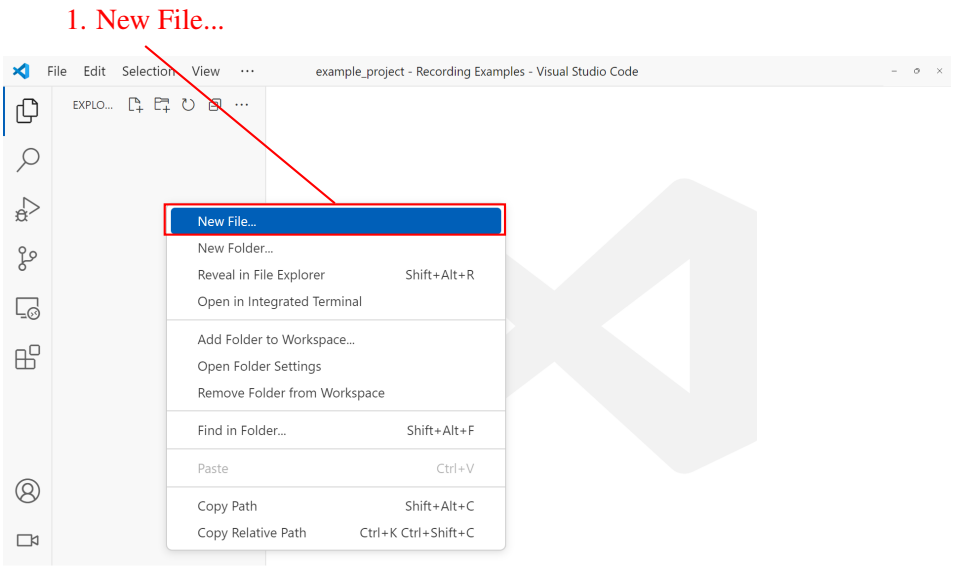
Step 3. Select an empty folder or create a new one.



Step 4. A pop-up will appear, asking you to confirm that you trust the authors of the files in the selected folder. You need to confirm for the language features of the extension to work.

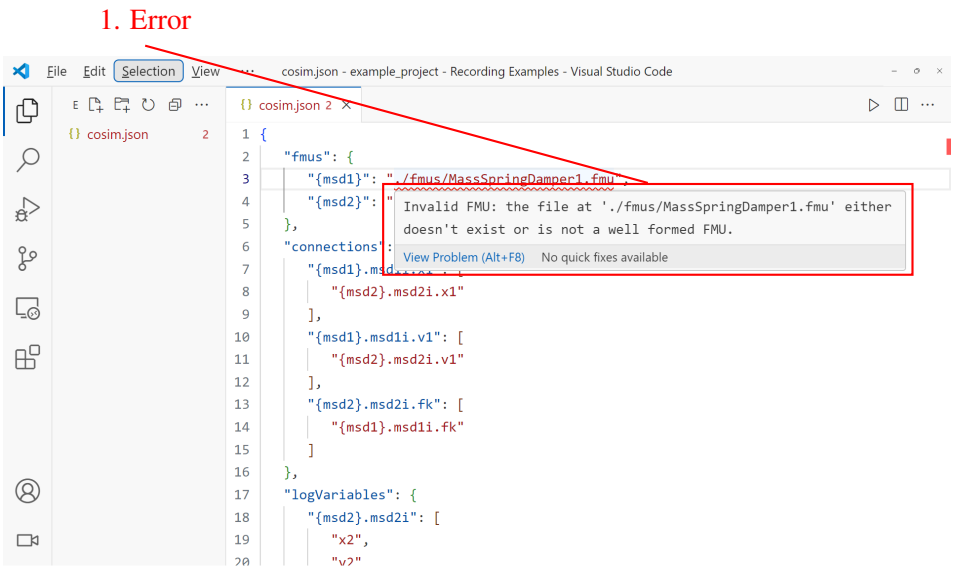


Step 5. Create a new file named `cosim.json`.

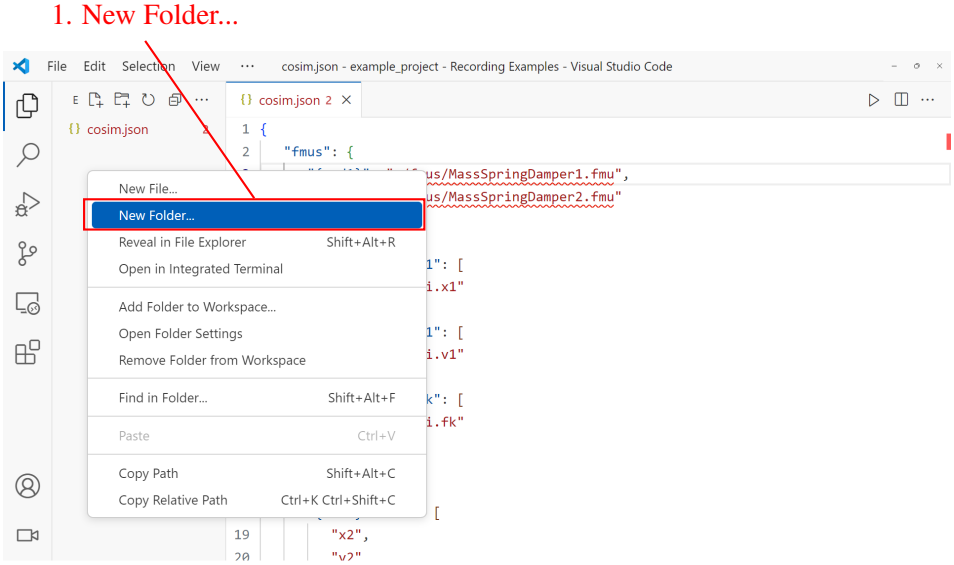


Step 6. We will be using the example found at `https://www.github.com`. Copy-and-paste the contents of the example `cosim.json` file into your own `cosim.json`.

Step 7. The paths referring to the location of the FMUs should now be underlined with a red squiggle. If you hover over the path, you should see that the reason for the error is that the extension cannot find a valid FMU at that location.

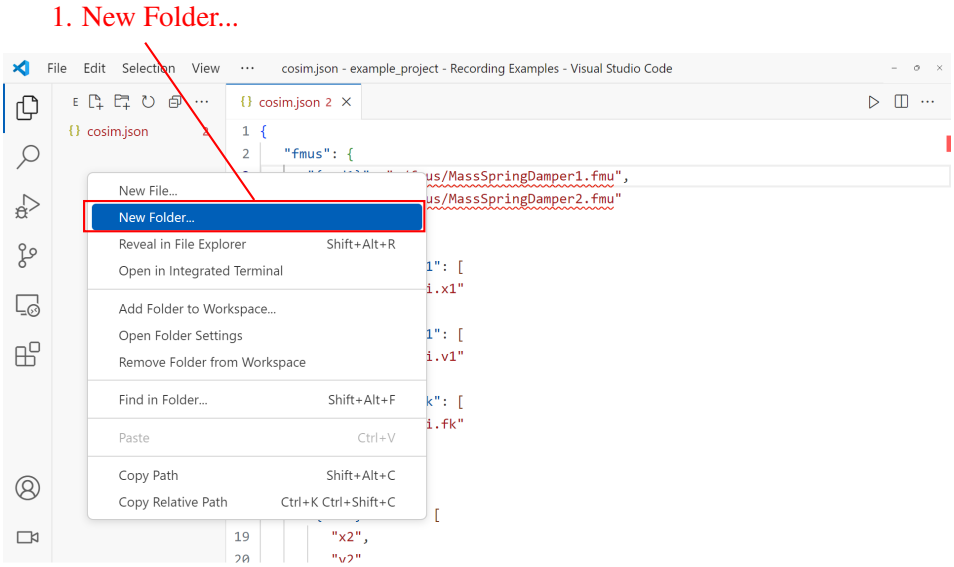


Step 8. To make the FMUs available to the extension, we will create a new folder named `fmus` in the workspace to contain the FMUs.

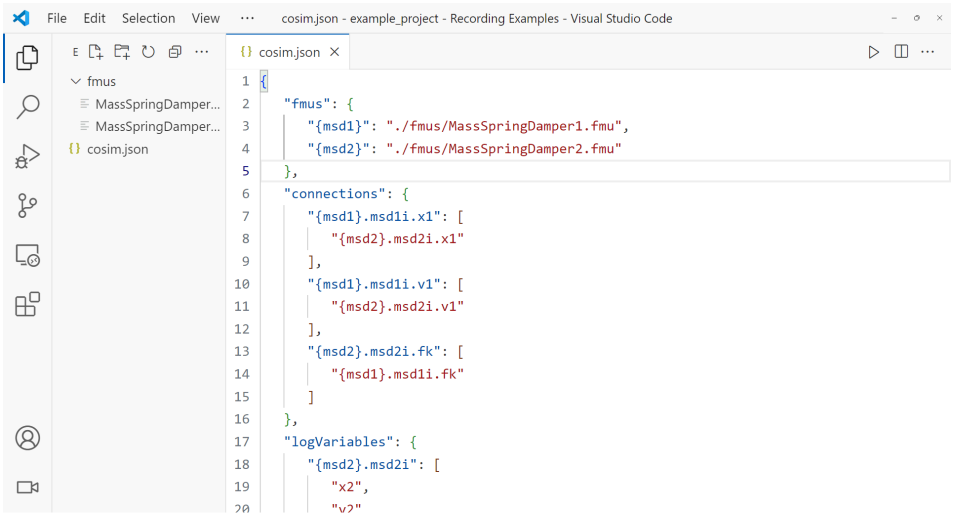


Note: The extension does not impose any restrictions on the structure of the project, so you can name the folder anything you like, remembering to update the corresponding path in `cosim.json`.

Step 9. Populate the newly created folder with the two FMUs found at `https://www.github.com`.



Step 10. The errors indicating that the FMUs were not found should now have cleared. At this point, we have a very basic project set up that is ready for simulation.

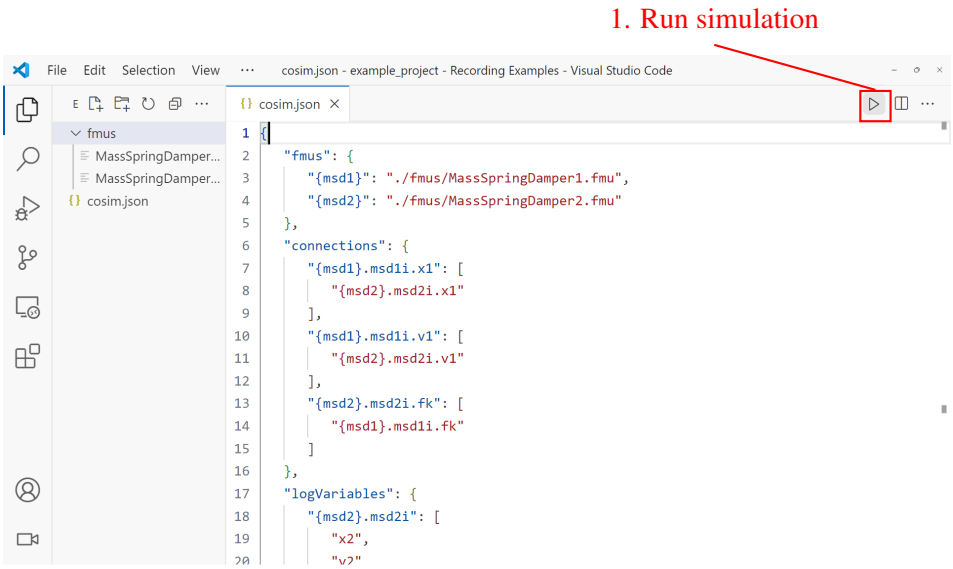


2 Run a co-simulation from within Visual Studio Code

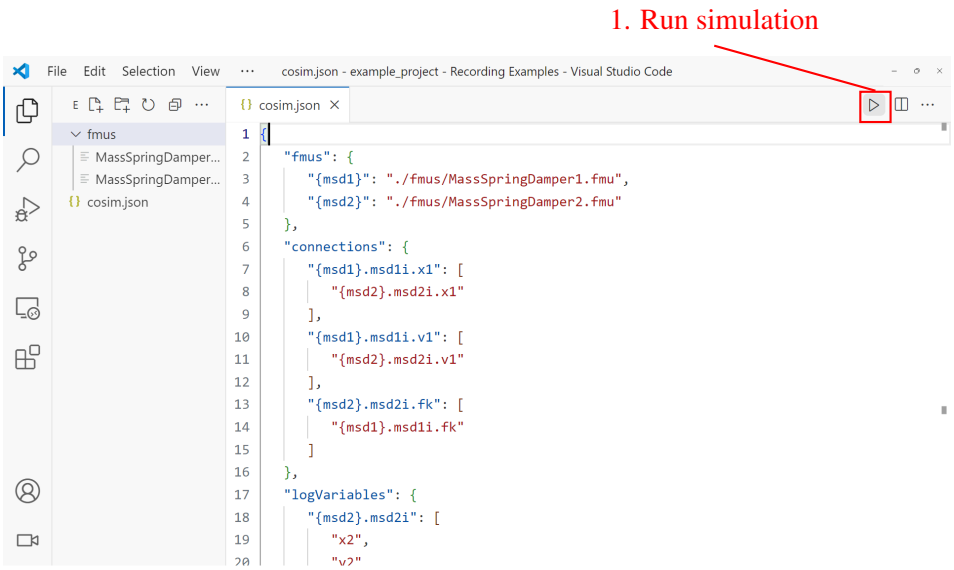
Step 11. Before running a simulation, ensure that `Maestro` is running on your local device. Open a terminal in the folder where you have installed `Maestro`, and then run `java -jar maestro-webapi-<late`
You should see something similar to the image below:

```
markus:~$ java -jar maestro-webapi-2.4.1-bundle.jar
Version: 2.4.1

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.7.RELEASE)

19:19:00.401 [background-preinit] WARN  org.springframework.http.converter.json.Jackson2ObjectMapperBuilder - For Jack
son Kotlin classes support please add "com.fasterxml.jackson.module:jackson-module-kotlin" to the classpath
19:19:01.221 [main] INFO  org.apache.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8082"]
19:19:01.222 [main] INFO  org.apache.catalina.core.StandardService - Starting service [Tomcat]
19:19:01.222 [main] INFO  org.apache.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.34]
19:19:01.269 [main] INFO  org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/] - Initializing Spring embedd
ed WebApplicationContext
19:19:02.141 [main] INFO  org.apache.coyote.http11.Http11NioProtocol - Starting ProtocolHandler ["http-nio-8082"]
```

Step 12. Run the simulation from Visual Studio Code.



Step 13. After running for a short period, the results show up in a new file



Congratulations!

You have now successfully set up and simulated your first co-simulation project in Visual Studio Code.