# Setup of PEST++ and PSDDF

## PEST and PSDDF Versions used (at the time of writing)

* PSDDF – v2.1 March 2003
* PEST 4.3.11

# Files required to run PSDDF

|  |  |
| --- | --- |
| **File Name** | **Purpose** |
| Addamse.tmp | 1. Contains the name of the PSDDF to be executed (“Model\_input” in the below input) 2. Couples PSDDF with addamse |
| “Model\_Input”.psi | Model inputs for PSDDF |
| PSDDF.exe | PSDDF program |
| PSDDF\_input.txt | Input name for the PSDDF |

# Files required to run PEST

|  |  |
| --- | --- |
| **File Name** | **Purpose** |
| “Template File”.tpl | PEST template file which controls which variables are read from the PSDDF input file. |
| “Instruction File”.ins | PEST Instruction file – Tells PEST how to read data from the specified output file |
| “Control File”.pst | PEST Control file – plays 2 major roles   1. Defines which files are the template and instruction files for this model run and the corresponding PSDDF files 2. Defines the parameters required for PEST’s iteration process (i.e. derivative type, error tolerance, objective function weights, etc.) |
| Run\_PestPSDDF.bat | Batch file that runs PSDDF and cleans up the folder so that PEST is able to run |
| Pestpp-“model\_Type”.exe | User chosen model from the pestpp suite. Refer to pestpp manual or link: <https://github.com/usgs/pestpp> for more information |

There are two different software packages sometimes referred to as PEST. The one linked to on GitHub is PEST++, also referred to as PESTPP. The other software package is just referred to as PEST. PEST is the historic version and future updates will be targeted to PEST++. PEST++ is a more generalized, has better capabilities, and is open source. PEST++ and PEST use similar file inputs, but there are some important differences. Refer PEST++ user manual for more information. The version of the PEST++ manual available at the time of writing this document is available in the “Notes” folder in the repo.

## Notes on the creation of PEST++ input files

### Steps to create Template file

1. Determine which PSDDF parameters PEST++ should have access to (e.g. choosing to iterate on the specific gravity of the first material). These will be the input parameters that PEST++ have access to solve the inverse problem and/or do a sensitivity analysis on
2. Make a copy of the PSDDF input (.psi) file and save it so that it has a file extension of .tpl. To do this:
   1. Open the .psi file in notepad
   2. Click “Save as”
   3. Change the “Save as type” selection to “All files”
   4. Save as “Arbitrary filename”.tpl. The filename doesn’t matter but the extension must be .tpl
3. On the first line of the .tpl file type ptf, a space, then the delimeter of your choice. For example in the Specific gravity example this line of the .tpl file is “ptf #”. PEST++ sets some restrictions on the delimiter chosen, refer to the instruction manual for more information.
   1. It’s recommended the delimiter be left as a pound sign (#).
   2. The delimiter is used to surround the character spaces that the chosen iteration values should be placed when PEST++ creates the .psi file for PSDDF. Refer to the specific gravity example and look for #Gs\_val# for an example.
   3. **The naming of the variable can limit the precision PEST++ is able to iterate input values with**. For example, if the delimiter is “#” and the variable name input into the .tpl file is “t” and the variable is delimited like #t# then PEST will only be able to increment the value of t in terms of whole numbers 0-9. If the variable is delimited as #t # then PEST++ be able to place two digits in that delimited space, if the name is #t # then PEST++ can use three digits. The number of spaces between the left and right delimiter is the number of digits PEST++ can place in that space. Therefore, make the space as large as possible, within reason. PSDDF seems to delimit the .psi file by spaces so, it shouldn’t matter how long the name is, but if an error occurs with PSDDF try decreasing the variable name.

### Steps to create the instruction file

### Steps to create the control file

## Model creation checks

1. Inschek executable (Checks instructions for the PEST model and outputs what PEST reads the output from the PSDDF file to be)
2. Tempchek executable (Checks the input template file and outputs what PEST thinks the input parameters are)
3. Pestchek executable (Checks input, instruction and control file to make sure that syntax and values are being properly read. Does Inschek and Tempchek in the process but doesn’t output the read values so running Inschek and Tempchek are still valuable)
   1. Pestchek is legacy code that was written for Pest and not PEST++. Therefore, the error that says “Line 5 of file FindGs\_Control.pst: either 4 variables (old style) or 7 variables are expected on this line, with the possible addition of a value for OBSREREF.” doesn’t apply to running any of the .exe in the PEST++ suite.

Link to PEST-PSDDF coupled repo: <https://github.com/WaveHello/PSDDF-Pest_Testing>

Link to PESTPP repo: <https://github.com/WaveHello/pestpp_Moore>

## Running PEST++ with PSDDF

1. Check that all steps in the model creation check section and notes on PEST++ files have been followed
2. Open the command prompt
3. Change the directory to the one where the input files and executables are located
   1. To do this copy the directory of the folder that the input files and executables
   2. In the command prompt type “cd” then paste the copied directory
   3. Hit enter
   4. The directory displaye on the command prompt should change to the input file folder
4. In the command prompt type “pestpp-“Name of the pestpp application your trying to run” “Name of the PEST control file”” (Note: the quotation marks are just to highlight where certain names should be. They shouldn’t be typed into the command prompt.)
   1. For example, if the PEST control file has a name Control.PST and you want to run the PEST++ GLM executable type “pestpp-glm Control” into the command prompt