Peridigm
DLR-IB-FA-BS-2019-33

Peridigm Installation Guide

For Peridigm versions $\geq$ 1.4.1

Martin Rädel

DLR German Aerospace Center
Composite Structures and Adaptive Systems
Structural Mechanics
Braunschweig
Copyright © 2018 German Aerospace Center (DLR)

Permission is granted to copy, distribute and/or modify this document under the terms of
the BSD Documentation License. A copy of the license is included in the section entitled
“BSD Documentation License”.

Dieses Dokument darf unter den Bedingungen der BSD Documentation License vervielfältigt,
distribuiert und/oder modifiziert werden. Eine Kopie der Lizenz ist im Kapitel “BSD Docu-
mentation License” enthalten.
## Contents

| List of Figures | vii |
| List of Listings | viii |

1. **About Peridigm** | 1 |
   1.1. Description | 1 |
   1.2. Addresses | 1 |
   1.3. Hardware requirements | 2 |
   1.4. Features | 2 |
      1.4.1. Example 1 | 2 |
      1.4.2. Example 2 | 2 |
      1.4.3. Example 3 | 2 |
   1.5. License | 3 |
   1.6. Peridigm working environment | 3 |
   1.7. Dependencies | 3 |

2. **Linux in a virtual machine** | 6 |
   2.1. Download Linux Distribution | 6 |
   2.2. Install the virtual machine | 6 |
      2.2.1. Download and install virtualization software | 6 |
      2.2.2. Create the virtual machine | 9 |
      2.2.3. Create a shared folder between host and virtual machine | 16 |
      2.2.4. Install the operating system in the virtual machine | 17 |
      2.2.5. User modifications to use shared folders | 23 |
      2.2.6. Save the virtual machine state | 23 |

3. **Peridigm Linux installation** | 25 |
   3.1. Tested combinations | 25 |
      3.1.1. CPU-Architecture & operating system | 25 |
      3.1.2. Library and system combinations | 27 |
   3.2. Basics | 28 |
      3.2.1. Preliminary remarks | 28 |
      3.2.2. Root .bashrc file | 29 |
3.2.3. Fortran & C & C++-compiler ........................................ 30
3.2.4. CMake ................................................................. 31
3.2.5. MPI ................................................................. 33
3.2.6. Python ......................................................... 36
3.3. System libraries ..................................................... 38
  3.3.1. Necessary libraries ........................................ 38
  3.3.2. Libraries that are not supposed to be installed yet ........ 38
3.4. Libraries .......................................................... 39
  3.4.1. Boost ......................................................... 39
  3.4.2. HDF5 ......................................................... 40
  3.4.3. NetCDF-C .................................................. 42
  3.4.4. Trilinos .................................................... 43
3.5. Peridigm .......................................................... 45
  3.5.1. Download ................................................... 45
  3.5.2. Compiling & installation .................................. 47
  3.5.3. After building and installing ................................ 48
  3.5.4. Install a local version on the cluster ...................... 48
  3.5.5. Use of Docker ............................................. 49
4. Running Peridigm .................................................... 50
5. Install ParaView ...................................................... 51
  5.1. Linux installation ............................................... 51
  5.2. Windows installation .......................................... 52
6. Install everything for FETranslator ............................... 53
  6.1. Linux ............................................................ 53
    6.1.1. Java ....................................................... 53
    6.1.2. NetCDF-C ................................................ 56
  6.2. Windows ........................................................ 56
    6.2.1. Java ....................................................... 56
    6.2.2. NetCDF-C ................................................ 57
Appendix A. Build-scripts for Libraries ...................... 59
  A.1. Boost ........................................................... 59
    A.1.1. Boost 1.55.0 ............................................ 59
    A.1.2. Boost 1.60.0 ............................................ 60
  A.2. HDF5 ........................................................... 60
  A.3. NetCDF-C ...................................................... 61
  A.4. Trilinos ....................................................... 61
A.5. Peridigm .................................................. 63
  A.5.1. CMake script for Peridigm .......................... 63
  A.5.2. Script for cloning Peridigm from GitHub and compiling on the STM-
          Cluster ............................................ 64
A.6. Make a script executable ................................. 67
A.7. Modifications of .bashrc ................................ 68
  A.7.1. .bashrc for Peridigm 1.4.1 .......................... 68
  A.7.2. .bashrc for Peridigm 1.5 ............................. 69

Appendix B. FAQ ............................................. 71

Appendix C. Useful Linux commands ......................... 73

Appendix D. Non-necessary tools & scripts .................. 75
  D.1. NEdit ................................................. 75
  D.2. RM-LATEX ............................................. 75
      D.2.1. Linux ........................................ 76
      D.2.2. Windows ...................................... 76

Appendix E. This document .................................. 77
  E.1. Repository .......................................... 77
  E.2. Typesetting ......................................... 77
  E.3. Structure ........................................... 78

BSD Documentation License ................................ 79
# List of Figures

1.1. Impact failure ............................................. 2  
1.2. Tensile test ............................................... 2  
1.3. Fragmentation ............................................. 3  
1.4. Peridigm working environment tree ....................... 4  
1.5. Peridigm package dependencies .......................... 5  

2.1. Allow VirtualBox access to devices ....................... 8  

3.1. *Peridigm GitHub* repository screenshot .................. 46
List of Listings

A.1. Install script for Boost 1.55.0 .................................................. 59
A.2. Install script for Boost 1.60.0 ..................................................... 60
A.3. Install script for HDF5 ............................................................. 60
A.4. Install script for NetCDF-C ...................................................... 61
A.5. CMake script for Trilinos ....................................................... 61
A.6. CMake script for Peridigm ...................................................... 63
A.7. Script for cloning Peridigm from GitHub and compiling on the STM-Cluster ........................................... 64
A.8. Modified .bashrc-file to set environment variables for Peridigm 1.4.1 ........................................ 68
A.9. Modified .bashrc-file to set environment variables for Peridigm 1.5 ........................................ 69
1. About Peridigm

1.1. Description

Peridigm is an open-source computational peridynamics code developed at Sandia National Laboratories for massively-parallel multi-physics simulations. It has been applied primarily to problems in solid mechanics involving pervasive material failure. Peridigm is a C++ code utilizing foundational software components from Sandia’s Trilinos project and is fully compatible with the Cubit mesh generator and ParaView visualization code.

Peridigm development began under the Physics & Engineering Models element of the US DOE’s Advanced Simulation and Computing (ASC) program. The project was led by Michael Parks and managed by John Aidun. Subsequent funding has been provided by the US DOE through the ASC, ASCR, and LDRD programs.

1.2. Addresses

Official homepage: https://peridigm.sandia.gov/
Source code repository: https://github.com/peridigm/peridigm
Release snapshots: Must be requested via Download Registration Form on https://peridigm.sandia.gov/content/download-registration-form
User guide [1]: http://www.sandia.gov/djlittl/docs/PeridigmV1.0.0.pdf

1.3. Hardware requirements

Currently, there are no hardware requirements available or known. Publications, reports and presentations continuously mention the massive parallelization possible and presumably also necessary to run Peridigm.
Information on the performance of *Peridigm* on different systems will be added as soon as it is available.

### 1.4. Features

#### 1.4.1. Example 1

The simulation of impact and brittle fracture displayed in Figure 1.1 was achieved using explicit transient dynamics, the linear peridynamic solid constitutive model, short-range force contact, and a critical stretch bond failure law.

Peridynamics provides a natural framework for capturing pervasive material failure and fracture.

#### 1.4.2. Example 2

*Peridigm* is capable of performing explicit dynamic, implicit dynamic, and quasi-static time integration.

The tensile test simulation presented in Figure 1.2 was attained using an elastic correspondence constitutive model and quasi-static time integration. Pre- and post-processing were carried out using Sandia’s *Cubit* mesh generator and *ParaView* visualization code.

#### 1.4.3. Example 3

The fragmentation of an expanding cylinder, shown in Figure 1.3, was simulated using the linear peridynamic solid constitutive model and critical-stretch bond failure rule. Initial velocities for each node in the discretization were specified via user-supplied analytic expressions.

*Peridigm* utilizes the RTCompiler function parser to process C-style expressions for the specification of input parameters, including initial and boundary conditions.
1. About Peridigm

1.5. License

As of 04.02.2016 Peridigm is distributed under the three-term BSD license.

The current license terms can be obtained from:
https://peridigm.sandia.gov/content/license.

1.6. Peridigm working environment

Peridigm is a standalone tool for the handling of peridynamic models. Sandia National Labs suggests the use of CUBIT as the mesh generator and pre-processor and ParaView for post-processing. However, CUBIT is not publicly available free of charge except for U.S. government agencies. Any visualization package capable of reading and displaying ExodusII-format [2] data may be used to visualize the output of a Peridigm simulation. The creators of Peridigm use ParaView.

This installation guide focusses on the installation of the core program Peridigm. Several libraries are required as well as some basic tools Peridigm or one of its libraries is dependent on. All required tools for the use of Peridigm in a working environment are shown in Figure 1.4.

![Peridigm working environment tree](image)

Figure 1.4.: Peridigm working environment tree
The installation guide features descriptions to the installation of all basic tools, libraries and *Peridigm* itself.

1.7. Dependencies

Below the dependencies between the individual tools and packages are shown.

![Peridigm package dependencies](image)

Figure 1.5.: *Peridigm package dependencies*
2. Linux in a virtual machine

If no native Linux-based operating system is available on a device it is possible to install a Linux distribution inside a virtual machine. The device used to install the virtual machine in is called the host in the description.

In case you have a native Linux operating system on your device you can skip the following steps and go to chapter 3.

2.1. Download Linux Distribution

For Peridigm we need a Linux distribution for the installation inside the virtual machine. Here, openSUSE is used. To download the latest version of openSUSE go to:

https://en.opensuse.org/Main_Page

In the top bar of the homepage go to the download section and choose Latest stable release. On the newly loaded page choose to download the DVD image as installation medium and perform the download onto your device. Beware, the image is quite big.

2.2. Install the virtual machine

2.2.1. Download and install virtualization software

For the current case VirtualBox is chosen as the virtualization environment. VirtualBox is available for enterprise and home use and is freely available for both use-cases as Open Source Software under the terms of the GNU General Public License (GPL) version 2. The following chapter describes the setup of the then current version of openSUSE in a VirtualBox on your host operation system. VirtualBox is currently available for
2. Linux in a virtual machine

Windows  ➤  OSX  ➤  Linux  ➤  Solaris

host operating systems. The current release of VirtualBox is available from:
https://www.virtualbox.org/

Go to the download section and choose the installer for your host operating system. The following description is valid for a Windows host and is tested for the 64bit variant of Windows 7. After the download execute the binary installer and follow the instructions of the setup wizard. In the custom setup step simply let all selected items enabled. Afterwards simply complete the wizard.

During the installation simply answer the following questions with install to allow access on the devices for the virtual machine.
2. Linux in a virtual machine

(a) Install USB-Controller

(b) Install network adapter

(c) Install network service

Figure 2.1.: Allow VirtualBox access to devices
2. Linux in a virtual machine

2.2.2. Create the virtual machine

After the installation is complete start the VirtualBox Manager.

Before we create a new virtual machine

- From the menu bar: Click File
- Click Preferences
- In the General tab select the Default Machine Folder to be a folder on a harddrive with sufficient amount of free memory to handle your virtual machine size, here 200GB
Now we can create a new virtual machine. Click New in the VirtualBox Manager main window. A dialog appears that guides you through the setup of the virtual machine.

1. Select the virtual machine name, type and version
   - Choose a name that includes the version & the selection automatically jumps to your choice
   - Choose the 64bit variant

2. A new folder with your chosen name is created in the directory you specified in the General tab of the Preferences toolbar menu
   - Click Next
2. Set the amount of RAM you grant your virtual machine
   ✗ The more RAM *Peridigm* has available the better
   ✗ But your host operating system also still needs some RAM left to work with
   ✗ Choose an integer multiple of 1024 as amount of RAM
   ✗ Leave the host operating system at least 2GB of RAM
   ✗ You can select values in the red marked area

   ➤ Click *Next*

3. Create a virtual hard disk to install your distribution on
   ➤ Here we select the *Create a virtual hard disk now* option
2. Linux in a virtual machine

- Click Create
- Select VMDK (Virtual Machine Disk) to allow use of this virtual hard disk in other virtualization tools like VMWare.

- Click Next
- Select Fixed size
- Do not select Split into files less than 2GB
1. Click Next
2. Setup virtual hard disk name and size
3. By default the name is identical to the virtual machine name
4. If you click the folder button you can change the location of the virtual hard disk file. By default it is saved in your virtual machine folder in the directory specified in the General tab of the Preferences toolbar menu
5. Set the size of the virtual hard disk to the amount you want and have free, here 200GB. Despite being saved in binary format, Peridigm result files can be quite big and for explicit time integration a lot of them are created. Despite a Linux distribution does not need nearly as much hard disk space as a Windows installation, at least 120GB are proposed as virtual hard disk size.

![Create Virtual Hard Disk](image)

6. Click Create
7. Wait for the creation to finish
8. Afterwards you have a new virtual machine
2. Linux in a virtual machine

4. Before we install the Linux distribution inside the virtual machine, we have to configure some settings

➤ Select the newly created virtual machine and click Settings
➤ In the General tab
  - Select Advanced
  - Select: Shared Clipboard Bidirectional
  - Drag’n’Drop Bidirectional

![VirtualBox Manager](image)
Ignore the warning about *Invalid settings detected* if this is only a RAM issue.

In the **System** tab:
- Select **Processor**
- Set **Processor(s)** to your preferred value
- At least one CPU should be left for the host machine.

In the **Display** tab:
2. Linux in a virtual machine

- Select Screen
- Set Video Memory to maximum

Click OK

All modifications to the virtual machine preferences in step 4 can be modified after the installation of the virtual machine distribution in case the virtual machine is shut down.

2.2.3. Create a shared folder between host and virtual machine

1. In the host operating system:
   - Open a Windows Explorer
   - Create a shared folder for the file exchange between the host and the virtual machine operating system anywhere it suits you or use an existing one, here the shared folder is E:\virtualization\Transfer
   - Right-Click the newly created folder and click Properties
   - In the Shared Folders/Freigabe tab
     - Click on Freigabe
     - In the dialog appearing click on the combobox arrow and select Anyone/-Jeder
     - Click Freigabe
     - Click Close
2. Linux in a virtual machine

Be aware that the folder is visible in your whole network

2. Inside the virtual box manager:
   ➤ Select the newly created virtual machine and click Settings
   ➤ In the Shared Folders tab
     - Click the + folder button on the right
     - In Folder Path: create or add a transfer folder between your host system and your virtual machine
     - Select Auto-mount
   - Click OK

2.2.4. Install the operating system in the virtual machine

Now, we can install the virtual machine operating system:

DLR – DLR-IB-FA-BS-2019-33
2. Linux in a virtual machine

1. Start the installation

   ➤ Select the virtual machine in the VirtualBox Manager and click Start

   ➤ Select the openSUSE image from section 2.1
2. **Linux in a virtual machine**

2. Setup the installation

→ Click **Start**

2. In the openSUSE boot menu select **Installation**

→ Set the Language and Keyboard layout to your preferred option
2. Linux in a virtual machine

- Click Next
- In the Installation Options to not toggle on any of the options
- Click Next
- Use the Suggested Partition and click Next
- Click on the Map to select your country to set Clock and Time Zone and click Next
- Use KDE Desktop in Desktop Selection and click Next
2. **Linux in a virtual machine**

Setup the first user name & password
- You can use the same user and root password if you are and will always be the only user of your virtual machine
  - User: *Username*  
    *Password*  
    13112
  - Admin: *Password*  
    dlr-fa-13112-bs
  - Click next

In the next windows click *Install*

3. **After installation**

After the installation is complete virtual system restarts. Now select *Boot from Hard Disk*
2. Linux in a virtual machine

Select the normal version and press Enter

Login with your password

Open a shell
2. Linux in a virtual machine

- Login as root user, perform

  ```
  zypper refresh
  zypper update
  ```

  And select yes to perform an operating system update

- Restart the virtual machine operating system

Ta-daa you have a Linux installation inside of a virtual machine.

2.2.5. User modifications to use shared folders

In order for the Linux users to use the virtual machine shared folders

1. Open YaST2
2. In the Security and Users tab open User and Group Management
3. Select the user and click Edit
4. Go to Details tab
5. On the right select users and vboxsf as Additional Groups and click OK
6. Restart your virtual system

The shared folders are mounted under /media/. In the current case the single shared folder is accessible under /media/sf_transfer/.

2.2.6. Save the virtual machine state

After the operating system installation it is recommended to save a snapshot of the current virtual machine state. Thus, it is always possible to reset your virtual machine to this state in case anything goes wrong during the Peridigm installation.

To create a snapshot:

1. Open the VirtualBox Manager
2. In the upper right corner select Snapshots
3. Click the little camera button to take a snapshot of the current virtual machine state.
3. *Peridigm* Linux installation

3.1. Tested combinations

3.1.1. CPU-Architecture & operating system

3.1.1.1. System 1 - STM-Laptop

<table>
<thead>
<tr>
<th>Hardware:</th>
<th>CPU:</th>
<th>Name:</th>
<th>Intel Core i7 M620</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Architecture:</td>
<td>x86_64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cores:</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Threads:</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clock rate:</td>
<td>2.67GHz</td>
<td></td>
</tr>
<tr>
<td>RAM:</td>
<td>Amount:</td>
<td>8Gb</td>
<td></td>
</tr>
<tr>
<td>Software:</td>
<td>OS:</td>
<td>Name:</td>
<td>openSUSE</td>
</tr>
<tr>
<td></td>
<td>Version:</td>
<td>42.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type:</td>
<td>64bit</td>
<td></td>
</tr>
</tbody>
</table>
3. Peridigm Linux installation

3.1.1.2. System 2 - Virtual machine

Hardware: Host: CPU: Name: Intel Core i7-4600U
Architecture: x86_64
Cores: 2
Threads: 4
Clock rate: 2.1GHz
RAM: Amount: 8Gb
VM: CPU: Cores: 2
RAM: Amount: 6Gb
Software: Host: VirtualBox 5.0
VM: OS: Name: openSUSE
Version: 42.1
Type: 64bit

3.1.1.3. System 3 - STM-Cluster

Hardware: CPU: Name: Intel Xeon E5-2407
Architecture: x86_64
Number: 8
Cores: 4
Threads: 4
Clock rate: 2.2GHz
RAM: Amount: 32Gb
Software: OS: Name: SUSE Linux Enterprise Server
Version: 11
Patchlevel: 3
Type: 64bit
### 3.1.2. Library and system combinations

<table>
<thead>
<tr>
<th>Setup</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>STM-Laptop</td>
<td>VirtualBox</td>
<td>VirtualBox</td>
<td>STM-Cluster</td>
</tr>
<tr>
<td>Compiler</td>
<td>gcc-fortran</td>
<td>gcc-fortran</td>
<td>gcc-fortran</td>
<td>gcc-fortran</td>
</tr>
<tr>
<td>Fortran</td>
<td>4.8.5</td>
<td>4.8.5</td>
<td>4.8.5</td>
<td>4.8.5</td>
</tr>
<tr>
<td>Basics</td>
<td>gcc 4.8.5</td>
<td>gcc 4.8.5</td>
<td>gcc 4.8.5</td>
<td>gcc 4.8.5</td>
</tr>
<tr>
<td>C</td>
<td>gcc-c++</td>
<td>gcc-c++</td>
<td>gcc-c++</td>
<td>gcc-c++</td>
</tr>
<tr>
<td>C++</td>
<td>4.8.5</td>
<td>4.8.5</td>
<td>4.8.5</td>
<td>4.8.5</td>
</tr>
<tr>
<td>CMake</td>
<td>3.4.3</td>
<td>3.4.3</td>
<td>3.5.1</td>
<td>3.5.2</td>
</tr>
<tr>
<td>MPI</td>
<td>Open MPI</td>
<td>1.10.2</td>
<td>1.10.2</td>
<td>1.10.4</td>
</tr>
<tr>
<td>MPICH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Python</td>
<td>2.7.11</td>
<td>2.7.9</td>
<td>2.7.9</td>
<td>2.6.9</td>
</tr>
<tr>
<td>Libraries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boost</td>
<td>1.55.0</td>
<td>1.55.0</td>
<td>1.60.0</td>
<td>?</td>
</tr>
<tr>
<td>HDF5</td>
<td>1.8.16</td>
<td>1.8.16</td>
<td>1.10.0</td>
<td>?</td>
</tr>
<tr>
<td>NetCDF-C</td>
<td>4.4.0</td>
<td>4.4.0</td>
<td>4.4.0</td>
<td>4.4.1</td>
</tr>
<tr>
<td>Trilinos</td>
<td>12.4.2</td>
<td>12.4.2</td>
<td>12.6.1</td>
<td>?</td>
</tr>
<tr>
<td>Peridigm</td>
<td>1.4.1</td>
<td>1.4.1</td>
<td>1.5.0</td>
<td>1.4.1</td>
</tr>
</tbody>
</table>
3.2. Basics

The following section describes the installation of the required basic packages for the Peridigm libraries. The subsections are in the order required for a proper installation.

3.2.1. Preliminary remarks

3.2.1.1. Installation shell

The following description is valid for an installation using the bash as shell. If you use any other shell, e.g. ksh, csh or their descendents you have to modify the environment variable parts of this guide or simply type bash and Enter in your tcsh to switch shells.

3.2.1.2. Download directories

During the course of this installation guide it will be necessary to download several source code packages. In the instructions the key $DOWNLOAD_DIR is the identifier for the download directory. The scripts in the appendix of this documents assume that

```
$DOWNLOAD_DIR=/usr/local/src/
```

You are free to choose any other folder as the download directory. If you do so, you have to modify the source code path in the scripts in the appendix accordingly.

3.2.1.3. PATH variable for user defined installation directories

It is assumed that the current installations are performed for all users of the device. Thus, the global installation directory /usr/bin/ is used.

However, the installation directory for each tool can be changed, e.g. with

```
./configure --prefix="/home/$USER/TOOL"
```

If this is done, the path to the executables have to exported to the PATH variables.

```
export PATH="$PATH:/home/$USER/TOOL/bin"
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/home/$USER/TOOL/lib/"
```
This has to performed for each individual installation directory.

NEVER EVER FORGET THE $PATH: AND $LD_LIBRARY_PATH: IN THE BEGINNING OR A NEW EMPTY PATH VARIABLE WILL BE CREATED AND ADDED TO. YOU WILL NOT BE ABLE TO USE ANYTHING USEFUL ON YOUR DEVICE BECAUSE ALL ALIASES WILL BE DELETED.

3.2.1.4. Installation user

All installations are performed as root user. To make sure that the end user is allowed to use the installed programs make sure that they have the necessary permissions. The default installation directory for installations with a package manager or zypper is /usr/bin. You can check if the permissions are correct by opening a terminal, navigation to /usr/bin and the command

```
ls -l | grep PARTOFTOOLABBREVIATION
```

For `ls -l | grep gcc` the result might look something like this:

```
lrwxrwxrwx 1 root root 7 5. Feb 14:07 cc -> gcc-4.8
lrwxrwxrwx 1 root root 7 5. Feb 14:07 gcc -> gcc-4.8
-rwxr-xr-x 1 root root 755680 29. Okt 18:02 gcc-4.8
```

In the first column the current rights are specified. The first symbol shows if the current entry is a link or not. The nine symbol afterwards are the three individual rights for user, group and other. The three individual rights are r-read, w-write and x-execute.

If the permissions are not set correctly, consult the documentation of `chmod`.

3.2.2. Root .bashrc file

You usually login as a normal user to openSUSE but changes to the system are performed as root user.

During this installation guide modifications of the .bashrc, a hidden file in the user home directory are requested. For the installation process as root user it is necessary that these modifications also take effect for the root user.
3. Peridigm Linux installation

To achieve this a symbolic link is created in the root home directory to the modified `.bashrc` in the ordinary user directory. To achieve this open a console and perform the following steps:

```
su  # Switch to root user
```
```
cd  # Change to root home
```
```
lne -sf /home/$USERNAME/.bashrc .  # Create symbolic link
```

This way, the root `.bashrc` file is always an identical copy of the one of the user with the name `$USERNAME` here.

3.2.3. Fortran & C & C++-compiler

*Peridigm* as well as *Python* require an acceptable C or C++-compiler. *Trilinos* additionally needs a Fortran-compiler. Here, the free *GNU Compiler Collection* versions, short *GCC* are used. The current release and further informations can be found on [https://gcc.gnu.org/](https://gcc.gnu.org/)

Currently, there are two main versions available, *GCC*, which is basically *GCC* version 4.8, as well as *GCC5*. The installation of the used *Python* currently seems not to work with *GCC5*. Additionally, *Trilinos* needs a compiler that is C++11 compliant and thus needs *GCC* version **4.7.2 or later**. Therefore *GCC* version 4.8 is used. If using Intel compilers, version 13 or later is required by *Trilinos*.

Normally, the *GCC* repository is already part of an *openSUSE* distribution. To check the availability of the *GCC*-repository in your *openSUSE* distribution open a terminal as root and use the following command to get a list of all repositories:

```
zypper repos
```

### Installation with YaST2

To install the Fortran-, C- and C++-compilers of *GCC* with the package manager perform the following steps:

1. Open *YaST2*
2. Click on *Install software*
3. Go to the *Search* tab
4. Search for *GCC*
5. Check *gcc-fortran*, *gcc* and *gcc-c++*
6. Click on *apply*
3. Peridigm Linux installation

Installation from source

ToDo

Installation with openSUSE-repository

To use zypper open a terminal as root. Use the following commands to install Fortran-, C- and C++-compilers of GCC from the repositories. Answer the questions if installation shall continue with yes.

```bash
zypper install gcc-fortran
zypper install gcc
zypper install gcc-c++
```

The installation usually is performed to /usr/bin/. If another installation directory is used, it has to be made sure, that this directory is part of the $PATH-variable. To check if this is the case, open a terminal type

```bash
echo $PATH
```

The installation directory has to be an entry of the printed string.

3.2.4. CMake

CMake is cross-platform free and open-source software for managing the build process of software using a compiler-independent method. It is maintained by Kitware. The official homepage of is CMake

https://cmake.org/

Trilinos release 12.4 and higher use a CMake build system, which requires CMake version 2.8.11 or newer.

Installation from source

In order to install CMake from the official or any other binary source open a terminal and login as root. Change directory to the designated download folder, e.g. /usr/local/src/ and perform the following steps:
3. Peridigm Linux installation

```bash
cd $DOWNLOAD_DIR
wget http://www.cmake.org/files/v3.4/cmake-3.4.3.tar.gz  # Download
unzip

cd cmake-3.4.3
./configure --prefix=/usr/local/bin/cmake-3.4.3 > configure_cmake.log 2>&1
make > make_cmake.log 2>&1  # build
make install > make_install_cmake.log 2>&1  # install
```

If you want to see the available configuration options, run the command below in the terminal.

```bash
./configure --help
```

In order to configure the installation directory of CMake before installation, run the command below:

```bash
./configure --prefix=/opt/cmake
```

After installation without any errors you can verify the installation by running the command below:

```bash
/usr/local/bin/cmake-3.4.3/bin/cmake -version
```

The output should look something like below (depending upon CMake version you are installing).

```bash
cmake version 3.4.3
```

Afterwards, the CMake-directory has to be added to the PATH environment variable

```bash
export PATH=$PATH:/usr/local/bin/cmake-3.4.3/bin
```

**Installation with openSUSE-repository**

For the installation of the build process manager zypper visit:

```bash
http://software.opensuse.org/download.html?project=server%3Airc&package=cmake
```

You can either choose the 1-Click-installation or add the repository and install manually. For the latter login to a terminal as root and type
zypper addrepo http://download.opensuse.org/repositories/server:irc/
   openSUSE_Leap_42.1/server:irc.repo
zypper refresh
zypper install cmake

3.2.5. MPI

For parallel computations on multiple cores an implementation of the Message Passing Interface is required. Only one of the following possibilities is required. The current implementation uses Open MPI.

3.2.5.1. Open MPI

Open MPI is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. The current version and further information can be found at

https://www.open-mpi.org

Installation with YaST2

To install Open MPI with the package manager perform the following steps:

1. Open YaST2
2. Click on Install software
3. Go to the Search tab
4. Search for Open MPI
5. Check Open MPI
6. Click on apply
3. Peridigm Linux installation

Installation with openSUSE-repository

The Open MPI-repository is part of the openSUSE distribution. Therefore, it can be directly installed from the system repositories.

To use zypper open a terminal as root. Use the following commands to install Open MPI from the repositories. Answer the questions if installation shall continue with yes.

\[
\text{zypper install openmpi}
\]

Installation from source

The Gzipped tarball source files can be obtained from

\[
\text{https://www.open-mpi.org/software/ompi/}
\]

In the subfolder choose the version of your liking. Change directory to the designated download folder, e.g. /usr/local/src/ and perform the following steps:

\[
\begin{align*}
\text{cd} & \quad \text{DOWNLOAD_DIR} \\
\text{wget} & \quad \text{https://www.open-mpi.org/software/ompi/v1.10/downloads/openmpi-1.10.2.tar.gz} \\
\text{tar} & \quad \text{xvfz} \quad \text{openmpi-1.10.2.tar.gz} \quad \# \quad \text{unzip} \\
\text{cd} & \quad \text{openmpi-1.10.2} \quad \# \quad \text{go into directory} \\
\text{./configure} & \quad \text{--prefix=/usr/local/lib/openmpi-1.10.2} \quad \text{> configure_openmpi.log} \quad 2>&1 \\
\text{make} & \quad \text{> make_openmpi.log} \quad 2>&1 \quad \# \quad \text{build} \\
\text{make altinstall} & \quad \text{> make_install_openmpi.log} \quad 2>&1
\end{align*}
\]

If no previous version of Open MPI exists use \texttt{make install} instead of \texttt{make altinstall}.

Set the PATH variables

Unfortunately the Open MPI installation does not work out of the box. You need to set the PATH and LD_LIBRARY_PATH variables and edit a configuration file first.

The LD_LIBRARY_PATH must be set so that mpi4py can find the Open MPI libraries.

In bash do for 32-bit

\[
\begin{align*}
\text{export PATH} & \quad \text{=$PATH:/usr/local/lib/openmpi-1.10.2/bin} \\
\text{export LD_LIBRARY_PATH} & \quad \text{=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/lib}
\end{align*}
\]
3. Peridigm Linux installation

or 64-bit

```bash
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/lib64
```

We recommend you add this line to your `.bashrc` file in case you use Bash or call `setenv` and edit the `.cshrc` file if you use a C shell so that the variable is set correctly for all sessions. For the modification of user `.bashrc` file for all libraries, please consult section A.7.

3.2.5.2. MPICH

`MPICH` is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard.

**Use the operating system distribution**

ToDo

**Use 1-click install**

Go to

https://software.opensuse.org/package/mpich

and choose 1-Click-Install or download the rpm-file from the source.

**Installation with openSUSE-repository**

ToDo
### Installation from source

Try

```bash
cd $DOWNLOAD_DIR
zypper si -d mpich2 # install the build deps
# for the previous version
wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz
tar xvfz mpich-3.2.tar.gz # unzip
cd mpich-3.2 # go into directory
./configure > configure_mpich.log 2>&1
make > make_mpich.log 2>&1 # build
make altinstall > make_install_mpich.log 2>&1 # install
```

### 3.2.6. Python

Use the operating system distribution

*Python* is already part of an openSUSE standard installation since also system components require python. The installed version can be shown in the terminal by the command

```
python -V
```

Packages are available for both *Python 2.7* as well as *Python 3.x*. A parallel installation if *Python 2* and *Python 3* possible without problems or package conflicts.

To update the python distribution to the newest available state in the OS repositories, open a terminal, login as `root` and use the following command

```
zypper update python
```

Additionally, `python-devel` is required, so

```
zypper install python-devel
```

### Perform a new installation

If no initial version of *Python* is present in the operating system it is necessary to download the source and install the source. For the latest or required version of *Python* visit

http://www.python.org/download/
For the installation, open a terminal and change directory to /home/USERNAME/bin for a single-user installation or /usr/local/bin for an installation for all users.

```
cd $DOWNLOAD_DIR
wget https://www.python.org/ftp/python/2.7.11/Python-2.7.11.tgz  # Download
          # unzip
tar xvfz Python-2.7.11.tgz
   # go into directory
   cd Python-2.7.11
  ./configure
        # build
       make
         # install
        make altinstall
```

Afterwards, you are free to delete the downloaded Gzipped source tarball, here.

```
cd $DOWNLOAD_DIR
   rm Python-2.7.11.tgz  # delete
```
3. System libraries

3.3.1. Necessary libraries

Install or update the following system libraries in advance of the Peridigm library installation.

```bash
zypper install libbz2-devel
zypper install zlib-devel
zypper install m4
zypper install blas
zypper install lapack
zypper install libX11-devel
```

3.3.2. Libraries that are not supposed to be installed yet

Open YaST2 and go to Software Management. In the Search field type HDF5 and look whether an older version than the HDF5 version you want to use is already installed on the system.

This is the case if you have octave-forge-netcdf installed on your system. If you do not really use this package you can uninstall it in YaST2 together with the installed HDF5 libraries.

If you really need octave-forge-netcdf which comes with HDF5-1.8.15 and have it installed already you can skip the installation of HDF5 described here and change the rest of the installation to use version 1.8.15 of HDF5.
3.4. Libraries

3.4.1. Boost

*Boost* provides free peer-reviewed portable C++ source libraries. *Boost* libraries are intended to be widely useful, and usable across a broad spectrum of applications. For more informations and the current release visit

http://www.boost.org/

*Peridigm* requires *Boost*, version **1.37 or later**, including the *regex* and *unit_test* compiled libraries. *Boost* installations on many systems include header files only. This is not sufficient, the required libraries must be compiled and installed. To ensure proper execution of *Peridigm* and its unit tests, add the *Boost* directory `$INSTALL_DIR/lib` to your `LD_LIBRARY_PATH` (Linux) and/or `DYLD_LIBRARY_PATH` (Mac) environment variables.

**Use 1-click install**

There is a *RPM Package Manager* file available on the *openSUSE* homepage. However, *Boost* requires some additional libraries to be compiled specifically. Therefore, it is not tested if the following installation with the 1-click install option is sufficient for *Peridigm*. It is recommended to use the installation using the source files from a Gzipped tarball as described in the next paragraph.

Go to

https://software.opensuse.org/package/boost

and choose 1-Click-Install or download the rpm-file from the source. Be cautious, the version offered is not necessarily up to date. Consult

http://www.boost.org/

for the current release.
Installation from source

The current release of Boost is available from

http://www.boost.org/

and sourceforge:

http://sourceforge.net/projects/boost/files/boost/

Originally, I tried to install the then-current version 1.60. Unfortunately, this led to compilation errors in combination with the current version of the GCC-compiler. After searching for a solution of the issue, version 1.55 is recommended.

First, the additional libraries `libbz2-devel` and `zlib1g-dev` have to be installed in section 3.3 before we can install Boost. Additional a new root shell must be opened to load the new environment variable additions for Open MPI.

```bash
cd $DOWNLOAD_DIR
wget http://sourceforge.net/projects/boost/files/boost/1.55.0/
  boost_1_55_0.tar.gz
  tar xvfz boost_1_55_0.tar.gz          # unzip
  cd boost_1_55_0                      # go into directory

Afterwards, create the Boost build script as described in section A.1. In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created install script and execute it with

```bash
./install_boost-1.55.0.sh > install_boost.log 2>&1
```  

The printout of the installation is written to `install.log`. It should be checked if all components of Boost are compiled correctly.

Afterwards, the Boost-directory has to be added to the LD_LIBRARY_PATH environment variable

```bash
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/boost-1.55.0/lib
```

3.4.2. HDF5

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. For further information visit
https://www.hdfgroup.org/HDF5/

HDF5 version **1.8.9 or newer** is required by NetCDF-C and the SEACAS Trilinos package. HDF5 should be configured with the --enable-parallel option.

### Installation with openSUSE-repository

HDF5 is available in an openSUSE-repository and can be installed using zypper. However, it is recommended to use the manual install with the Gzipped tarball to make sure the correct options are set.

```bash
zypper addrepo http://download.opensuse.org/repositories/home:ocefpaf/
              openSUSE_Tumbleweed/home:ocefpaf.repo
zypper refresh
zypper install hdf5
```

### Installation from source

The HDF5 source code is available from

https://www.hdfgroup.org/HDF5/

Download the source code for your platform

```bash
cd $DOWNLOAD_DIR
wget http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.16.tar.gz
  # unzip
tar xvfz hdf5-1.8.16.tar.gz
  # go into directory
cd hdf5-1.8.16
```

Afterwards, create the HDF5 build script as described in section A.2. In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created install script and execute it with

```bash
./install_hdf.sh > install.log 2>&1
```

Afterwards, the HDF5-directory has to be added to the PATH and LD_LIBRARY_PATH environment variable

```bash
export PATH=$PATH:/usr/local/bin/hdf5-1.8.16/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/hdf5-1.8.16/lib64
```
3. Peridigm Linux installation

3.4.3. NetCDF-C

NetCDF-C (Network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. Distributions are provided for Java and C/C++/Fortran. For more information visit

http://www.unidata.ucar.edu/software/netcdf/

NetCDF-C is required by the Trilinos SEACAS package. NetCDF-C should be configured with the --disable-netcdf-4 and --disable-dap options.

Installation from source

The NetCDF-C source code is available from the NetCDF-C homepage. Download the source code for your platform

```
cd $DOWNLOAD_DIR
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4.4.0.tar.gz
unzip netcdf-4.4.0.tar.gz # unzip
cd netcdf-4.4.0 # go into directory
```

Prior to compiling NetCDF-C, it is recommended that you modify the file netcdf.h in $DOWNLOAD_DIR/netcdf-4.4.0/include/ to better support large-scale Peridigm simulations. Modify the following #define statements in the netcdf.h file. Change the values to match what is given below.

```
#define NC_MAX_DIMS 65536
#define NC_MAX_ATTRS 8192
#define NC_MAX_VARS 524288
#define NC_MAX_NAME 256
#define NC_MAX_VAR_DIMS 8
```

Afterwards, create the NetCDF-C build script as described in section A.3. In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created install script and execute it with

```
./install_netcdf.sh > install.log 2>&1
```

Due to an apparent glitch in the NetCDF-C installer, in some cases it may be necessary to manually copy the file $DOWNLOAD_DIR/netcdf-4.4.0/include/netcdf_par.h from the source distribution into the installation include subdirectory.
3. Peridigm Linux installation

```bash
cp $DOWNLOAD_DIR/netcdf-4.4.0/include/netcdf_par.h /usr/local/bin/netcdf-4.4.0/include/
```

Afterwards, the NetCDF-C-directory has to be added to the PATH and LD_LIBRARY_PATH environment variable

```bash
export PATH=$PATH:/usr/local/bin/netcdf-4.4.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/netcdf-4.4.0/lib64
```

3.4.4. Trilinos

The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. A unique design feature of Trilinos is its focus on packages. For more information visit

https://trilinos.org/

A number of Trilinos packages are required by Peridigm. The Trilinos source code distribution includes the full set of Trilinos packages, each of which may be activated or deactivated using CMake build options, as described below. It is recommended that Makefiles be created by running `cmake` from the command line, as opposed to using the `ccmake` GUI.

The current release of Trilinos can be obtained from the download section of the Trilinos homepage. The download needs a short registration with a valid email-address. The download link is likely to be not reachable without the registration.

```bash
cd $DOWNLOAD_DIR
wget http://trilinos.csbsju.edu/download/files/trilinos-12.4.2-Source.tar.gz
tar xvfz trilinos-12.4.2-Source.tar.gz
```

Trilinos does not allow the use of the directory with the source-files for the further progress of the installation. Therefore, create a new folder

```bash
mkdir trilinos-12.4.2
```

and copy the file from section A.4 to the new folder. Change the line for the Open MPI- and the Trilinos-source-directory (last line) if necessary.

In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created path and execute it with
3. **Peridigm Linux installation**

![](image)

Once *Trilinos* has been successfully configured, it can be compiled and installed as follows:

```bash
./cmake_trilinos.cmake > cmakeopts.log 2>&1
```

If there occur any errors during the compilation of *Trilinos* visit section B. For compiling with `make -j 4` more than 8GB of RAM are necessary. If you do not know if there were any compilation errors have occurred due to the long duration of the compilation process, repeat

```bash
make
```

after the original compilation with `make -j 4`. Only failed compilations are repeated. Afterwards perform

```bash
make install
```

The final installation can be found in the folder specified in `CMAKE_INSTALL_PREFIX:PATH` in the script from section A.4.

Afterwards, *Trilinos* has to be added to the `PATH` and `LD_LIBRARY_PATH` environment variables to later use the *Trilinos* decomposition features for the model decomposition for calculation on multiple processors.

```bash
export PATH=$PATH:/usr/local/bin/trilinos-12.4.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/trilinos-12.4.2/lib
```
3. Peridigm Linux installation

3.5. Peridigm

3.5.1. Download

3.5.1.1. Download the official release

The current official release of Peridigm can be obtained from the download section of https://peridigm.sandia.gov/

The download needs a short registration with a valid email-address. Download the .tgz file of your preferred Peridigm version to $DOWNLOAD_DIR. Unpack the archive:

```
cd $DOWNLOAD_DIR

tar xvfz Peridigm_1.4.1.tgz
```

3.5.1.2. Download the latest master version from GitHub

The Peridigm repository is available from GitHub and can be downloaded from: https://github.com/peridigm/peridigm

To obtain the latest master release go to that homepage and select the master branch in the top left and click on Download zip in the top right corner as shown by the red rectangles in Figure 3.1 or click this link.

Unpack the archive:

```
cd $DOWNLOAD_DIR

unzip peridigm-master.zip

mv peridigm-master Peridigm-1.4.1-Source
```

3.5.1.3. Checkout the latest version from GitHub

With svn

Use kdesvn to checkout the latest version. Checkout https://github.com/peridigm/peridigm.git/trunk
Figure 3.1: Peridigm GitHub repository screenshot
With git

Use *git* on the cluster:

```bash
git clone https://github.com/peridigm/peridigm.git
```

### 3.5.2. Compiling & installation

*Peridigm* utilizes the *CMake* build system. It is recommended that Makefiles be created by running *cmake* from the command line, as opposed to using the *ccmake* GUI.

The installation here is described for the then official *Peridigm* version 1.4.1. The steps for version 1.5 are identical besides the changes in folder names.

*Peridigm* does not allow the use of the directory with the source-files for the further progress of the installation. Therefore, create a new folder

```bash
mkdir Peridigm-1.4.1
```

and copy the file from section A.5 to the new folder. Change the lines for the library paths and the *Peridigm*-source-directory (last line) if necessary.

In order to use the script make it executable as described in section A.6. Open a terminal as root, change directory to the created path and execute it with

```bash
./cmake_peridigm.cmake > cmakeopts.log 2>&1
```

Once *Peridigm* has been successfully configured, it can be compiled and installed as follows:

```bash
make -j 4
make install
```

The default location for the created binary is `/usr/local/bin/`. In case you want to create the binary at a different location add the following line to the script from section A.5

```bash
-D CMAKE_INSTALL_PREFIX=/PATH/TO/DESTINATION
```

After installation make sure to change permissions of the installation directory for the necessary users and groups.
3. Peridigm Linux installation

### 3.5.3. After building and installing

[1] recommends to run `ctest` in your build directory after building and installing Peridigm to run all tests and confirm that you have a clean build. Alternatively, Peridigm offers an own test suite which is used here.

To be able to execute the test you first have to temporarily change the source and installation folder owner. Thus perform the following commands before executing the test.

```
chown -R $USERNAME:$GROUPNAME Peridigm-1.4.1
chown -R $USERNAME:$GROUPNAME Peridigm-1.4.1-Source
```

Before starting the test-suite as a normal user, open a new shell or use an existing one and re-register your username to load the current state of the `.bashrc`-file:

```
su - $USERNAME
```

The Peridigm test suite is then run from the terminal as non-root user as follows:

```
make test
```

Remember to revert the modifications to the Peridigm directory ownership after it is assured that all tests are passed with:

```
chown -R root:root Peridigm-1.4.1
chown -R root:root Peridigm-1.4.1-Source
```

### 3.5.4. Install a local version on the cluster

1. Login to the STM cluster and move to a directory of your convenience
2. Clone your local Peridigm version from GitHub with `git clone` or download the master zip-file and unpack, see section 3.5.1.1
3. Allocate cluster-node exclusively for building

   ```
salloc --exclusive
   ```

4. Load build and library environment

   ```
   . /cluster/software/slurm/etc/env.d/mpibuild.sh
   . /cluster/software/slurm/etc/env.d/peridigm.sh
   ```

5. Change directory to the folder, where the unpacked Peridigm source folder is located

   ```
   cd ~/src/peridigm
   ```

6. Create a build-directory and go to the new directory
3. Peridigm Linux installation

mkdir peridigm-build

cd peridigm-build

7. Save the following lines in the file cmake_peridigm.cmake. Change the path in the code to the correct location.

```bash
cmake \\
-D CMAKE_BUILD_TYPE:STRING=Release \\
-D CMAKE_INSTALL_PREFIX=/home/USER/peridigm-build \\
-D CMAKE_CXX_FLAGS:STRING='\-O2 \-Wall \-std=c++11 \-pedantic \-Wno-long-long \-fttrapv \-Wno-deprecated' \\
/home/USER/peridigm-master
```

8. Call CMake via terminal with the given code:

```bash
dir cmake_peridigm.cmake
```

9. Make

```bash
make -j 8
```

10. Test

```bash
make test
```

11. Create the executable

```bash
make install
```

12. Exit salloc shell

```bash
exit
```

13. The executable is now located at `~/src/peridigm/peridigm-build/bin`

14. For how to execute the local version using the cluster queuing-system look at the Peridigm User Guide

A complete script for cloning Peridigm from GitHub, compiling and installing on the cluster can be found in subsection A.5.2.

3.5.5. Use of Docker

4. Running *Peridigm*

A dedicated description on how to run *Peridigm* can be found in the *Peridigm* Users Guide which is part of the same repository as this document.
5. Install ParaView

ParaView is an open source multiple-platform application for interactive, scientific visualization. The current release and further informations can be found on http://www.paraview.org/

ParaView was developed to analyze extremely large datasets using distributed memory computing resources.

5.1. Linux installation

Use 1-click install

Go to https://software.opensuse.org/download/package?project=science&package=paraview and choose 1-Click-Install or download the rpm-file from the source.

Installation with openSUSE-repository

ParaView is part of the openSUSE Science Repository. The repository can be included into the package management.

```
zypper addrepo http://download.opensuse.org/repositories/science/
    openSUSE_Leap_42.1/science.repo
zypper refresh
zypper install paraview
```
5. Install ParaView

Installation from source

The source code of the current ParaView release is available from the download section of the ParaView homepage. For an installation using the source code and CMake please consult

http://www.paraview.org/Wiki/ParaView:Build_And_Install

5.2. Windows installation

Go to the download section of the ParaView homepage and download the binary installers for your windows operating system architecture. Afterwards, simply run the executable installer and follow the instructions.
6. Install everything for \textit{FETranslator}

\textit{FETranslator} is a \textit{Java}-based tool to translate models between finite element software. \textit{FETranslator} implements the conversion of meshes from commercial FE tools into the format that \textit{Peridigm} is capable of using as a discretization for the creation of peridynamic collocation points.

In order to use the \textit{FETranslator} an implementation of the \textit{Java} Runtime Environment (JRE) is necessary. To translate the mesh into binary format the tool \textit{ncgen} from \textit{NetCDF-C} is required.

6.1. Linux

6.1.1. Java

\textit{openSUSE} comes with a pre-installed version of the \textit{openJDK} which is a free and open source implementation of the Java Platform, Standard Edition (Java SE). \textit{openJDK} should be perfectly capable of running \textit{FETranslator}. To see if and which version of \textit{Java} is installed on your system open a shell and type:

\begin{verbatim}
java -version
\end{verbatim}

However, since additions and changes to the \textit{FETranslator} can be necessary, a \textit{Java}-capable IDE is required. The Oracle \textit{Java} Development Kit (JDK) offers an integrated solution with the JRE and \textit{NetBeans} as IDE.

6.1.1.1. Install only Java development kit (JDK)

1. Go to: \url{http://www.oracle.com/technetwork/java/javase/downloads/index.html}
2. Click on \textit{Java Platform (JDK)}
3. Accept the License Agreement
4. Open a shell and type \texttt{lsb_release -a} and check your operating system architecture (32bit: i586; 64bit: x86_64)
5. Click on the according rpm file for your Linux version (here: jdk-8u91-linux-i586.rpm for 32bit or jdk-8u91-linux-x64.rpm for 64bit, we use 64bit)
6. In the dialog choose *Save File* and save the file somewhere convenient on your system
7. Open a root shell or a normal shell and switch to root user with

```
su -
```
8. Change directory to the folder where the RPM file is located
9. Type

```
zypper install jdk-8u91-linux-x64.rpm
zypper install update-alternatives
update-alternatives --install /usr/bin/java java /usr/java/jdk1.8.0_91/bin/java 1065
update-alternatives --install /usr/bin/javac javac /usr/java/jdk1.8.0_91/bin/javac 1065
update-alternatives --install /usr/bin/jar jar /usr/java/jdk1.8.0_91/bin/jar 1065
update-alternatives --install /usr/bin/javaws javaws /usr/java/jdk1.8.0_91/bin/javaws 1065
update-alternatives --config java
java -version
nedit /home/$USERNAME/.bashrc &
```

10. Add `export JAVA_HOME=/usr/java/jdk1.8.0_91/` to the .bashrc-file

### 6.1.1.2. Install Java development kit (JDK) with NetBeans

2. Click on NetBeans with JDK
3. Accept the License Agreement
4. Open a shell and type `lsb_release -a` and check your operating system architecture (32bit: i586; 64bit: x86_64)
5. Click on the according sh file for your Linux version (here: jdk-8u91-nb-8_1-linux-x64.sh)
6. In the dialog choose *Save File* and save the file somewhere convenient on your system
7. Open a root shell or a normal shell and switch to root user with

```
su -
```
8. Change directory to the folder where the .sh file is located
9. Change the installer file's permissions so it can be executed:

```
chmod u+x <installer-file-name>
```

---

*DLR – DLR-IB-FA-BS-2019-33*
6. Install everything for FETranslator

10. Type

./<installer-file-name>

11. In the installation wizard:

a) At the Welcome page of the installation wizard, click Next.

b) At the JDK Installation page, specify the directory where to install the JDK, here /usr/local/java/jdk1.8.0_91, and click Next.

c) At the NetBeans IDE Installation page, do the following:
   - Specify the directory for the NetBeans IDE installation (here /usr/local/java/netbeans-8.1)
   - Accept the default JDK installation to use with the IDE or specify another JDK location.

d) Accept the default JDK installation to use with the IDE or specify another JDK location.

e) Click Next

f) Review the Summary page to ensure the software installation locations are correct.

g) Click Install to begin the installation.

h) At the Setup Complete page, provide anonymous usage data if desired, and click Finish

i) When the installation is complete, you can view the log file, which resides in the following directory: ~/.nbi/log.

12. Type

zypper install update-alternatives
update-alternatives --install /usr/bin/java java /usr/local/java/jdk1.8.0_91/bin/java 1065
update-alternatives --install /usr/bin/javac javac /usr/local/java/jdk1.8.0_91/bin/javac 1065
update-alternatives --install /usr/bin/jar jar /usr/local/java/jdk1.8.0_91/bin/jar 1065
update-alternatives --install /usr/bin/javaws javaws /usr/local/java/jdk1.8.0_91/bin/javaws 1065
update-alternatives --config java
java -version
nedit /home/$USERNAME/.bashrc &

13. Add

export JAVA_HOME=/usr/java/jdk1.8.0_91/
6. Install everything for FETranslator

6.1. Install everything for FETranslator

and

export PATH=$PATH:/usr/local/java/netbeans-8.1/bin
to the .bashrc-file

14. Start a new shell with `su - $USERNAME` and type `java -version` to see if the correct version is active
15. Start a new shell with `su - $USERNAME` and type `netbeans &` to start the IDE
16. Perform update inside the IDE if asked for

If problems occur during any `update-alternatives --install` try

```
update-alternatives --install /usr/bin/java java /usr/local/java/jdk1.8.0_91/bin/java 1
update-alternatives --install /usr/lib64/browser-plugins/javaplugin.so javaplugin /usr/local/java/jdk1.8.0_91/jre/lib/amd64/libnpjp2.so 1 --slave /usr/bin/javaws javaws /usr/local/java/jdk1.8.0_91/bin/javaws
update-alternatives --install /usr/bin/javac javac /usr/local/java/jdk1.8.0_91/bin/javac 1 --slave /usr/bin/jar jar /usr/local/java/jdk1.8.0_91/bin/jar
```

Now you can set the Java priorities with:

```
update-alternatives --config java
update-alternatives --config javac
update-alternatives --config javaplugin
```

6.1.2. NetCDF-C

The NetCDF-C-tool `ncgen` is required to convert the ascii mesh file into the binary format readable by Peridigm. NetCDF-C should already be installed to use Peridigm. If the additions to the PATH-variable from subsection 3.4.3 are set, no further actions have to be performed.

6.2. Windows

6.2.1. Java

6. Install everything for FETranslator

2. Click on NetBeans with JDK for Development Kit and Netbeans or just Java Platform (JDK)
3. Perform the installation

6.2.2. NetCDF-C

To test the FETranslator under Windows it is necessary to have ncgen available. ncgen is available as part of pre-built NetCDF-C libraries. To install the latest release

1. Go to: http://www.unidata.ucar.edu/software/netcdf/
2. Click Pre-built Windows Binaries for the latest version of NetCDF-C
3. Go to Latest Release (NetCDF-C X.Y.Z), here NetCDF-C 4.4.0
4. Download the executable matching your system, here netCDF4.4.0-NC4-64.exe
5. Execute the installer
6. Add the bin folder of the installation path, here D:\Programme\netCDF 4.4.0\ to the Windows PATH-Variable:
   a) Open the Windows Control Panel (Systemsteuerung)
   b) Open System
   c) Click Advanced System Settings (Erweiterte Systemeinstellungen)
   d) In the Advanced tab open Environment Variables
   e) Under User variables for USERNAME select PATH
   f) Click Edit
   g) Under Value of the variable add the path to the bin folder of the NetCDF-C installation separated by a semicolon (;), here: ;D:\Programme\netCDF 4.4.0\bin\n   h) Click OK multiple times

Now you can use ncgen in a command-window:

ncgen.exe -o $OUTPUTFILENAME.g $INPUTFILENAME.g.ascii
Bibliography


Appendices
A. Build-scripts for Libraries

In the following sections, the build scripts for the libraries for Peridigm are collected. These are Bash-scripts or CMake-files.

The scripts are taken from https://peridigm.sandia.gov/ and modified slightly if necessary. The files are provided with UTF-8 encoding. Please modify to your needs if necessary.

A.1. Boost

A.1.1. Boost 1.55.0

Open a text editor, copy the following code into a file and save as install_boost-1.55.0.sh

Listing A.1: Install script for Boost 1.55.0

```
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Run the Boost bootstrap script
./bootstrap.sh

# add using mpi to user-config.jam
echo "using mpi ;" >> tools/build/v2/user-config.jam

# Compile and install Boost using the Boost's bjam build system
./b2 install --prefix=/usr/local/lib/boost-1.55.0/
```

Alternatively, you can download the script from within this document.
A. Build-scripts for Libraries

A.1.2. **Boost 1.60.0**

Open a text editor, copy the following code into a file and save as `install_boost-1.60.0.sh`

```bash
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Run the Boost bootstrap script
./bootstrap.sh

# add using mpi to project-config.jam
echo "using mpi ;" >> project-config.jam

# Compile and install Boost using the Boost's bjam build system
./b2 install --prefix=/usr/local/lib/boost-1.60.0/
```

Alternatively, you can download the script from within this document.

A.2. **HDF5**

Open a text editor, copy the following code into a file and save as `install_hdf.sh`

```bash
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Configure HDF5
./configure --prefix=/usr/local/bin/hdf5-1.8.16/ --enable-parallel

# Make and install HDF5
make -j 4
make install
```
Alternatively, you can download the script from within this document.

### A.3. NetCDF-C

Open a text editor, copy the following code into a file and save as `install_netcdf.sh`

Listing A.4: *Install script for NetCDF-C*

```bash
# Set environment variables for MPI compilers
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif77

# Configure NetCDF
./configure --prefix=/usr/local/bin/netcdf-4.4.0/ --disable-netcdf-4 --disable-dap

# Make, test, and install NetCDF
make -j 4
make check
make install
```

Alternatively, you can download the script from within this document.

### A.4. Trilinos

Open an editor, copy the following code into the file and save as `cmake_trilinos.cmake`. The final line marks the path to the Trilinos source directory, which is named `$DOWNLOAD_DIR` in the documentation.

Listing A.5: *CMake script for Trilinos*

```bash
rm -f CMakeCache.txt
rm -rf CMakeFiles/
cmake -D CMAKE_INSTALL_PREFIX:PATH=/usr/local/bin/trilinos-12.4.2/ \\
-D MPI_BASE_DIR:PATH="/usr/local/lib/openmpi-1.10.2/"
```
A. Build-scripts for Libraries

-D CMAKE_CXX_FLAGS:STRING="-O2 -std=c++11 -pedantic -ftrapv -Wall -Wno-long-long" \
-D CMAKE_BUILD_TYPE:STRING=RELEASE \n-D Trilinos_WARNINGS_AS_ERRORS_FLAGS:STRING="" \n-D Trilinos_ENABLE_ALL_PACKAGES:BOOL=OFF \n-D Trilinos_ENABLE_Teuchos:BOOL=ON \n-D Trilinos_ENABLE_Shards:BOOL=ON \n-D Trilinos_ENABLE_Sacado:BOOL=ON \n-D Trilinos_ENABLE_Epetra:BOOL=ON \n-D Trilinos_ENABLE_EpetraExt:BOOL=ON \n-D Trilinos_ENABLE_Ifpack:BOOL=ON \n-D Trilinos_ENABLE_AztecOO:BOOL=ON \n-D Trilinos_ENABLE_Amesos:BOOL=ON \n-D Trilinos_ENABLE_Anasazi:BOOL=ON \n-D Trilinos_ENABLE_Belos:BOOL=ON \n-D Trilinos_ENABLE_ML:BOOL=ON \n-D Trilinos_ENABLE_Phalanx:BOOL=ON \n-D Trilinos_ENABLE_Intrepid:BOOL=ON \n-D Trilinos_ENABLE_NOX:BOOL=ON \n-D Trilinos_ENABLE_Stratimikos:BOOL=ON \n-D Trilinos_ENABLE_Thyra:BOOL=ON \n-D Trilinos_ENABLE_Rythmos:BOOL=ON \n-D Trilinos_ENABLE_MOOCHO:BOOL=ON \n-D Trilinos_ENABLE_TriKota:BOOL=OFF \n-D Trilinos_ENABLE_Stokhos:BOOL=ON \n-D Trilinos_ENABLE_Zoltan:BOOL=ON \n-D Trilinos_ENABLE_Piro:BOOL=ON \n-D Trilinos_ENABLE_Teko:BOOL=ON \n-D Trilinos_ENABLE_SEACASioss:BOOL=ON \n-D Trilinos_ENABLE_SEACAS:BOOL=ON \n-D Trilinos_ENABLE_SEACASBlot:BOOL=ON \n-D Trilinos_ENABLE_Pamgen:BOOL=ON \n-D Trilinos_ENABLE_EXAMPLES:BOOL=OFF \n-D Trilinos_ENABLE_TESTS:BOOL=ON \n-D TPL_ENABLE_Matio:BOOL=OFF \n-D TPL_ENABLE_HDF5:BOOL=ON \n-D HDF5_INCLUDE_DIRS:PATH="/usr/local/bin/hdf5-1.8.16/include" \n-D HDF5_LIBRARY_DIRS:PATH="/usr/local/bin/hdf5-1.8.16/lib" \n-D TPL_ENABLE_Netcdf:BOOL=ON \n-D Netcdf_INCLUDE_DIRS:PATH="/usr/local/bin/netcdf-4.4.0/include" \n-D Netcdf_LIBRARY_DIRS:PATH="/usr/local/bin/netcdf-4.4.0/lib"
A. Build-scripts for Libraries

-D TPL_ENABLE_MPI:BOOL=ON \\
-D TPL_ENABLE_BLAS:BOOL=ON \\
-D TPL_ENABLE_LAPACK:BOOL=ON \\
-D TPL_ENABLE_Boost:BOOL=ON \\
-D Boost_INCLUDE_DIRS:PATH="/usr/local/lib/boost-1.55.0/include" \\
-D Boost_LIBRARY_DIRS:PATH="/usr/local/lib/boost-1.55.0/lib" \\
-D CMAKE_VERBOSE_MAKEFILE:BOOL=OFF \\
-D Trilinos_VERBOSE_CONFIGURE:BOOL=OFF \\
/usr/local/src/trilinos-12.4.2-Source/

Alternatively, you can download the script from within this document.

A.5. Peridigm

A.5.1. CMake script for Peridigm

Open an editor, copy the following code into the file and save as cmake_peridigm.cmake. The final line marks the path to the Peridigm source directory, which is named $DOWNLOAD_DIR in the documentation.

Listing A.6: CMake script for Peridigm

```
rm -f CMakeCache.txt
rm -rf CMakeFiles/

cmake \\
-D CMAKE_BUILD_TYPE:STRING=Release \\
-D Trilinos_DIR:PATH="/usr/local/bin/trilinos-12.4.2/lib/cmake/Trilinos/" \\
-D CMAKE_C_COMPILER:STRING="/usr/local/bin/openmpi-1.10.2/bin/mpicc" \\
-D CMAKE_CXX_COMPILER:STRING="/usr/local/bin/openmpi-1.10.2/bin/mpicxx" \\
-D BOOST_ROOT="/usr/local/lib/boost-1.55.0/" \\
-D CMAKE_CXX_FLAGS:STRING="-O2 -Wall -std=c++11 -pedantic -Wno-long-long 
    -ftrapv -Wno-deprecated" \\
/usr/local/src/Peridigm-1.4.1-Source
```

Alternatively, you can download the script from within this document.
A. Build-scripts for Libraries

A.5.2. Script for cloning Peridigm from GitHub and compiling on the STM-Cluster

Listing A.7: Script for cloning Peridigm from GitHub and compiling on the STM-Cluster

```bash
#!/bin/bash

# Header
# Install Peridigm from github repository master
#
# Requirements:
# Allocate exclusive node before using this script:
# salloc --exclusive
#
# Revisions: 2017-12-22 Martin Raedel <martin.raedel@dlr.de>
# Initial draft
#
# Contact: Martin Raedel, martin.raedel@dlr.de
# DLR Composite Structures and Adaptive Systems
#
# __/|__
# /_/_/_/
# www.dlr.de/fa/en  |/ DLR
#
# Content
#
# Variables

# The directory where all the magic happens - should not exist in advance
basedir=$HOME'/Documents/Peridigm/20171222EC_/

# Path from where to clone
githubclonepath='https://github.com/peridigm/peridigm.git'
```
# Internal directory names
builddir='build'
srcdir='src'

# File names
file_cmake='cmake_peridigm.cmake'
file_cmake_log='cmake_peridigm.log'
file_make_log='make.log'
file_make_test_log='make_test.log'
file_make_install_log='make_install.log'

file_peridigm_bin='Peridigm'

# Number of CPUs for make
make_cpus = 8

#--------------------------------------
# Script
#--------------------------------------

#------------------
# Load build environment
#------------------

echo 'Load build environment'
. /cluster/software/slurm/etc/env.d/mpibuild.sh
. /cluster/software/slurm/etc/env.d/peridigm.sh

#------------------
# Folder structure
#------------------

echo 'Create directory structure'
if [ -d ${basedir} ]; then # Control will enter here if $DIRECTORY doesn't exist.
  echo 'Directory ${basedir} already exists. Exit.'
  exit 0
fi

mkdir ${basedir}

cd ${basedir}
mkdir ${builddir}
mkdir ${srcdir}

cd ${srcdir}

#------------------
# Clone from GitHub
#------------------

echo 'Clone from GitHub'
git clone ${githubclonepath}
cd ../${builddir}

#------------------
# Create cmake file
#------------------

echo 'Create and execute cmake file'

# Fill file
echo 'rm -f CMakeCache.txt' >> ${file_cmake}
echo 'rm -rf CMakeFiles/' >> ${file_cmake}
echo '' >> ${file_cmake}
echo 'cmake -D CMAKE_BUILD_TYPE:STRING=Release ' >> ${file_cmake}
echo ' -D CMAKE_INSTALL_PREFIX='${basedir}${builddir}' ' >> ${file_cmake}
echo ' -D CMAKE_CXX_FLAGS:STRING=-O2 -Wall -std=c++11 -pedantic -Wno-long-long -ftrapv -Wno-deprecated' >> ${file_cmake}
echo '${basedir}${srdir}/peridigm/' >> ${file_cmake}

# Change permissions to make cmake-file executable
chmod u+x ${file_cmake}

# Execute cmake-file
./${file_cmake} > ${file_cmake_log} 2>&1

#------------------
# Make
#------------------

echo 'make'
made -j ${make_cpus} > ${file_make_log} 2>&1
echo 'make test'
make test > ${file_make_test_log} 2>&1
if grep -q Error ${file_make_test_log}; then
    echo ' make test contains failed tests'
else
    echo ' all tests passed'
fi

echo 'make install'
make install > ${file_make_install_log} 2>&1

#------------------
# Comment
#------------------

#--------------------------------------
# Clean
#--------------------------------------

Alternatively, you can download the script from within this document.

A.6. Make a script executable

In order to use a script file in the shell for installation you must first make the text file executable. Therefore, open a terminal, change directory to the folder the individual script is located and make the script executable for the user with

chmod u+x $SCRIPTNAME.sh
A. Build-scripts for Libraries

A.7. Modifications of .bashrc

When an interactive shell that is not a login shell is started, bash reads and executes commands from ~/.bashrc, if that file exists. You can find the .bashrc file in your user home directory /home/$USER/ with ls -al.

The following listings shows a modified .bashrc file which includes the exportation of the significant libraries in the $PATH and $LD_LIBRARY_PATH environment variables. The header is not printed.

Be aware:

➤ In case you use a 32bit operating system, or in some cases also for 64bit operating system, the lib64-folders must be changed to lib.
➤ The entries to the $PATH and $LD_LIBRARY_PATH variables should be added step-by-step after the installation of the individual tool. Otherwise, the install scripts might find pre-compiled items and use these instead of creating new binaries with the current settings.

A.7.1. .bashrc for Peridigm 1.4.1

Listing A.8: Modified .bashrc-file to set environment variables for Peridigm 1.4.1

```bash
# add default lib paths
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib64

# add cmake
export PATH=$PATH:/usr/local/bin/cmake-3.4.3/bin

# add openMPI
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/lib64

# add boost
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/boost-1.55.0/lib
```

DLR – DLR-IB-FA-BS-2019-33
A. Build-scripts for Libraries

# add hdf5
export PATH=$PATH:/usr/local/bin/hdf5-1.8.16/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/hdf5-1.8.16/lib64

# add netcdf
export PATH=$PATH:/usr/local/bin/netcdf-4.4.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/netcdf-4.4.0/lib64

# add trilinos
export PATH=$PATH:/usr/local/bin/trilinos-12.4.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/trilinos-12.4.2/lib

You can [download](#) the file from within this document.

### A.7.2. .bashrc for Peridigm 1.5

Listing A.9: Modified .bashrc-file to set environment variables for *Peridigm* 1.5

test -s ~/.alias && . ~/.alias || true

# add default lib paths
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib64

# add cmake
export PATH=$PATH:/usr/local/bin/cmake-3.5.1/bin

# add openMPI
export PATH=$PATH:/usr/local/lib/openmpi-1.10.2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/openmpi-1.10.2/lib

# add boost
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/boost-1.60.0/lib

# add hdf5
export PATH=$PATH:/usr/local/bin/hdf5-1.8.16/bin
A. Build-scripts for Libraries

```bash
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/hdf5-1.10.0/lib64

# add netcdf
export PATH=$PATH:/usr/local/bin/netcdf-4.4.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/netcdf-4.4.0/lib64

# add trilinos
export PATH=$PATH:/usr/local/bin/trilinos-12.6.1/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/bin/trilinos-12.6.1/lib
```

You can download the file from within this document.
B. FAQ

The error g++ internal compiler error killed (program cc1plus) occurs while compiling with make -j N. What is that?

- The problem is probably caused by insufficient memory.
- Try typing `free -m` in a terminal while compiling to see the amount of free RAM.
- Running `make -j 4` runs lots of process which use more memory. The problem above occurs when your system runs out of memory. In this case rather than the whole system falling over, the operating systems runs a process to score each process on the system. The one that scores the highest gets killed by the operating system to free up memory. If the process that is killed is cc1plus, gcc (perhaps incorrectly) interprets this as the process crashing and hence assumes that it must be a compiler bug. But it isn’t really, the problem is the OS killed cc1plus, rather than it crashed.
- If this is the case, you are running out of memory. So run perhaps `make -j 2` instead. This will mean fewer parallel jobs and will mean the compilation will take longer but hopefully will not exhaust your system memory.

When I call `make test` after the Peridigm installation all multi-processor-tests fail. What is the problem?

- Make sure you do not execute `make test` as root user. It is not allowed to call `mpirun` as root. If you call `mpirun` as root, the job is cancelled automatically.
- In order to run the tests as a normal user make sure you set the permissions of the Peridigm installation folder to allow execution for group and others.

Everything works fine, all Peridigm tests are passed, but when I call `decomp` I get a ***HDF5 library version mismatched error*** error. Why?

- It seems `decomp` finds another version of the HDF5 library `libhdf5.so` than the one you use to compile Trilinos with.
- This can be caused by installation of other tools via the YaST2 software management which puts libraries in `/usr/lib` or `/usr/lib64`. These are found in the current `LD_LIBRARY_PATH` before the HDF5 version installed in this guide.
- Tools which bring their own version of HDF5 are for example:

<table>
<thead>
<tr>
<th>Tool/Package</th>
<th>HDF5 Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>octave-forge-netcdf</td>
<td>1.8.15</td>
</tr>
</tbody>
</table>
If these packages are installed and you do not use them, uninstall them and the HDF5 libraries as described in subsection 3.3.2. You do not have to recompile anything. Just try using decomp afterwards.

In case you really really need the tool with the other HDF5 there seems to be nothing left to to but to compile Trilinos using this version of HDF5.

I get errors when using the provided CMake-files, like [...] command not found. What can I do?

This problem may arise if you import the appended scripts under Windows.

The end-of-line character might be changed to CR-LF instead of LF

To check this problem, open Notepad++ and go to Edit → EOL Conversion → Convert to Unix Format (LF)

Save and try using the script again.
# C. Useful Linux commands

<table>
<thead>
<tr>
<th>Category</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working with archives</td>
<td><code>tar xvfj [FILENAME].tar.bz2</code></td>
<td>Extract tar.bz2 file</td>
</tr>
<tr>
<td></td>
<td><code>tar xvfz [FILENAME].tar.gz</code></td>
<td>Extract tar.gz file</td>
</tr>
<tr>
<td></td>
<td><code>unzip [FILENAME].zip -d [DESTINATION_FOLDER]</code></td>
<td>Extract zip file to folder</td>
</tr>
<tr>
<td></td>
<td><code>tar cvfz [ARCHIVENAME].tar.bz2 [FILENAMES]</code></td>
<td>Pack files to tar.bz2 file</td>
</tr>
<tr>
<td>Files, directories, links</td>
<td><code>head -n [NUM] [FILENAME]</code></td>
<td>Print the first [NUM] lines of [FILENAME]</td>
</tr>
<tr>
<td></td>
<td><code>ln -sf source [TARGET]</code></td>
<td>Creates a symbolic link</td>
</tr>
<tr>
<td></td>
<td><code>ls -al</code></td>
<td>List files (visible and hidden) in current directory</td>
</tr>
<tr>
<td></td>
<td><code>tail -f [FILENAME]</code></td>
<td>Print and follow last line of file</td>
</tr>
<tr>
<td>Jobs</td>
<td><code>watch -n [SECONDS] [COMMAND]</code></td>
<td>Repeat command automatically</td>
</tr>
<tr>
<td>Memory</td>
<td><code>df -h</code></td>
<td>display the amount of available disk space</td>
</tr>
<tr>
<td></td>
<td><code>du -sch *</code></td>
<td>display the file and folder sizes of current directory</td>
</tr>
<tr>
<td></td>
<td><code>free -m</code></td>
<td>display the amount of available RAM</td>
</tr>
<tr>
<td>Permissions</td>
<td></td>
<td>continued . . .</td>
</tr>
</tbody>
</table>
### C. Useful Linux commands

<table>
<thead>
<tr>
<th>Category</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>chmod [OPTIONS] mode file1 ...</code></td>
<td>Change permissions of files or folders</td>
</tr>
<tr>
<td></td>
<td><code>chown [OPTIONS][USER][:[GROUP]] file1 ...</code></td>
<td>Change owner and group of file or folder</td>
</tr>
<tr>
<td>System information</td>
<td><code>cat /etc/*-release</code></td>
<td>Find out My Linux Distribution Name and Version</td>
</tr>
<tr>
<td></td>
<td><code>cat /proc/cpuinfo</code></td>
<td>Find out about CPU</td>
</tr>
<tr>
<td></td>
<td><code>cat /proc/meminfo</code></td>
<td>Find out about RAM</td>
</tr>
<tr>
<td></td>
<td><code>lsb_release -a</code></td>
<td>Find out about operating system</td>
</tr>
</tbody>
</table>
D. Non-necessary tools & scripts

The installation is described for openSUSE for version 42.1.

D.1. NEdit

NEdit, the Nirvana editor, is a text editor and source code editor for the X Window System. For the installation of the editor NEdit visit:

http://software.opensuse.org/download.html?project=editors&package=nedit

You can either choose the 1-Click-installation or add the repository and install manually. For the latter login to a terminal as root and type

```
zypper addrepo http://download.opensuse.org/repositories/editors/openSUSE_Leap_42.1/editors.repo
zypper refresh
zypper install nedit
```

D.2. RM-LATeX

Download the package via the intranet (from within the DLR network or via a VPN connection):

teamsites.dlr.de/rm/latex/SitePages/Homepage.aspx

Follow the instructions given in /doc/RM-LaTeX-Guide/RM-LaTeX-Guide.pdf.
D. Non-necessary tools & scripts

D.2.1. Linux

Before using `mktexlsr set`

```bash
chmod +t texmf/
```

and

```bash
chmod go+w texmf/
```

Due to some problems in the RM-LATEX package meaningful use is only possible under Windows. If using with Linux do not use everything related to the package `dlrsecondpage`.

D.2.2. Windows

There should be no additional steps necessary.
E. This document

E.1. Repository

This document is part of the PeriDoc repository. The complete repository can be found at:

https://github.com/PeriDoX/PeriDoX

A tool implementing the subversion protocols can be used to access the repository as well as update the files or commit changes using command line options or a graphical user interface.

Following is a list of tested possible tools:

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Tool</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>TortoiseSVN</td>
<td><a href="https://tortoisesvn.net">https://tortoisesvn.net</a></td>
</tr>
<tr>
<td>Linux with KDE</td>
<td>KDESvn</td>
<td><a href="http://kdesvn.alwins-world.de/">http://kdesvn.alwins-world.de/</a></td>
</tr>
</tbody>
</table>

E.2. Typesetting

This document is typeset using the RM-LaTEX package, see section D.2.

The compilation is performed with pdflatex with the following options:

```
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %source  -- extra-mem-top=60000000
```
E. This document

E.3. Structure

This document is part of the PeriDoc repository. The following files are used from the PeriDoc_Common folder:

BSD Documentation License

Redistribution and use in source (Docbook format) and 'compiled' forms (PDF, PostScript, HTML, RTF, etc), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (Docbook format) must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, HTML, RTF, and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of the author may not be used to endorse or promote products derived from this documentation without specific prior written permission.

THIS DOCUMENTATION IS PROVIDED BY THE AUTHOR ``AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.