In the previous chapters, we discussed various classes that implement MD features and LAMMPS tools, such as **pair styles**, **fixes**, **computes**, **variables**, and **groups**. From this chapter onward, we will write custom features into LAMMPS, starting with **custom pair potentials**.

In this chapter, we will construct custom pair potentials to describe the process of writing and incorporating them in a LAMMPS input script. In the process, we will use the content covered in the previous chapters to program LAMMPS according to our requirements.

We will cover the following pair potentials in this chapter:

- Writing a harmonic potential
- Writing a height-dependent pair potential
- Writing a tangential friction-based pair style for spherical atoms

By the end of this chapter, you will have learned how to write your own pair styles and how to connect information from previous chapters to custom pair styles.
Technical requirements

To execute the instructions in this chapter, you need a text editor (for example, Notepad++ or Gedit) and a platform to compile LAMMPS (for example, a Linux Terminal).

You can find the full source code used in this chapter here: https://github.com/PacktPublishing/Extending-and-Modifying-LAMMPS-Writing-Your-Own-Source-Code

This is the link to download LAMMPS: https://lammps.sandia.gov/doc/Install.html. The LAMMPS GitHub link is https://github.com/lammps/lammps, where the source code can be found as well.

Writing a harmonic potential

In this section, we will write a simple pairwise potential, that is, a harmonic potential, between two atoms.

A radially-symmetric harmonic potential tries to exert a force upon an atom pointed in the direction of the minimum energy location. The functional form of the harmonic potential, \( V_H\), is given by the following:

\[
V_H(r) = \frac{1}{2} k_s p (r - r_0)^2 - \varepsilon_0
\]

From the preceding equation, \( k_s p \) is the spring constant, \( r_0 \) is the equilibrium distance, and \( \varepsilon_0 \) is the well-depth of this harmonic potential. The radial force from this potential, \( F_H(r) \), is, therefore, given as follows:

\[
F_H(r) = -\frac{dV_H(r)}{dr} = -k_s p (r - r_0)
\]
Accordingly, the \( x, y, z \) force components \((F_{Hx}, F_{Hy}, F_{Hz})\) of \( F_H \) are calculated as follows:

- **x-component, \( F_{Hx}(x) \):**
  \[
  F_{Hx}(x) = -k_{sp}(r - r_0) \frac{x}{r}
  \]

- **y-component, \( F_{Hy}(y) \):**
  \[
  F_{Hy}(y) = -k_{sp}(r - r_0) \frac{y}{r}
  \]

- **z-component, \( F_{Hz}(z) \):**
  \[
  F_{Hz}(z) = -k_{sp}(r - r_0) \frac{z}{r}
  \]

To implement this harmonic pair style, we choose a similar pair style and make modifications. Among the many choices available, we will choose **Pair Morse**, which we analyzed earlier in *Chapter 5, Understanding Pair Styles*.

The following sections will describe the required changes to convert **Pair Morse** into a harmonic pair potential.

**Changing class names**

To change the class names, we will follow these steps:

1. First, we make copies of `pair_morse.cpp` and `pair_morse.h`, and rename them `pair_harmonic.cpp` and `pair_harmonic.h`, respectively. If the filenames start with `pair`, then they are automatically appended into `style_pair.h` during compilation. If not, they have to be manually added to it before compilation.
2. Then, in the newly created `pair_harmonic.h` file, we change the class names from Morse to Harmonic, as shown in the following screenshot:

![Figure 9.1 – Changes in class names between pair_morse.h (left) and pair_harmonic.h (right)](image)

On line 16, the pair style name (`PairHarmonic`) and the name of the pair in the LAMMPS input script (`harmonic`) are defined. The class name is changed to `PairHarmonic` in line 27 and lines 29 to 30.

3. Variables defined in `pair_morse.h` include $D_0, \alpha, r_0$, which need to be changed to $k_{sp}, \varepsilon_0, r_0$ in `pair_harmonic.h`, as shown in the following screenshot:

![Figure 9.2 – Variable changes between pair_morse.h (left) and pair_harmonic.h (right)](image)

The variable for global cutoff (`cut_global`), the arrays for local cutoff (`cut[]`), and the offset in potential (`offset[]`) are kept unchanged while the array to facilitate Morse potential calculation (`morse1[]`) is removed in `pair_harmonic.h`. As you can see in the preceding screenshot, on line 48, the quantity $D_0$ (`d0[]`) is changed to $k_{sp}$ (`ksp[]`), and the quantity $\alpha$ (`alpha[]`) is changed to $\varepsilon_0$ (`e0[]`) whereas the cutoff (`r0[]`) is unchanged (I have placed my initials, SM, next to the line to indicate the modification made):