MFiX Tutorial
Screw feeder with varying screw rotational velocity

Sathish Sanjeevi and Jeff Dietiker

National Energy Technology Laboratory, USA

November, 2019

1 Introduction

In this tutorial, a screw feeder is simulated with varying rotational velocity of the screw using user-defined functions (UDFs). The objectives of this tutorial are to:

- Create a screw made of particles. This is achieved by reading a input file (`particle_input.dat`) containing the particle positions representing the screw. The screw particles and the granular flow particles are differentiated by using scalar variable called `des_usr_var` .
- Use keyframe data to compute the screw rotational speed at different timesteps.
- Apply computed rotational velocity to update only the screw particles' position and velocity.

2 Keyframes

This section provides a general description of keyframe data usage. Keyframes are tables with numerical data to be used in simulations, and are currently implemented through UDFs. Such tables are useful to perform dynamic simulations, for example, a time dependent inflow boundary. Each keyframe is titled in the form `data_kf_<ID>.txt`. Here, ID is a four-padded integer representing the keyframe ID, which will typically be associated with the same boundary condition ID (BCIDs), although ID and BCID are independent from each other (it is the user's responsibility to decide how to use the keyframe data). The user-defined keyword `read_kf(ID)` allows reading a given keyframe file. Each keyframe file has the following structure:

```
nrows, nvar
interpolation_type
!  header
 x_1   y_{11} ...   y_{1n}
 .     .            .
 .     .            .
 x_m   y_{m1} ...   y_{mn}
```

Here, `nrows(=m)` is the number of data rows in the keyframe file, and `nvar(=n)` is the number of dependent variables. The first column defines a list of increasing values for the independent variable `x`. In this tutorial, we are using keyframes to define transient data, so `x` correspond to time. Other columns define values of the dependent variables `y`, for example velocity at corresponding time. The data in all columns are floating point (real) scalars. For vector quantities, users need to specify the 3 components of the vector using 3 columns (for example, if two scalars and one vector are defined, then `n=5`). `interpolation_type` defines the available interpolation types, namely `linear` or `step`. Once the keyframe data is defined, users can evaluate `y` at any `x`-value between the data points. Two interpolation schemes are available: 1) Linear interpolation, where `y` is assumed to vary linearly between two consecutive points (interval $[x_i; x_{i+1}]$) , and 2) Step function which returns the constant y-value y_i when the `x` value lies within the interval $[x_i; x_{i+1}]$. The actual keyframe reading (`read_keyframe_data`) and interpolation (`interpolate_keyframe_data`) functions are located in `usr_mod.f` file.

2.1 A keyframe example

The following example explains the actual value of a variable `var1` depending on the `interpolation_type`. Consider the following keyframe file:

```
6, 1
<interpolation_type>
!x                      var1
0.0                     0.5
2.0                     0.0
4.0                     2.5
6.0                     6.0
8.0                     4.0
10.0                    3.0
```

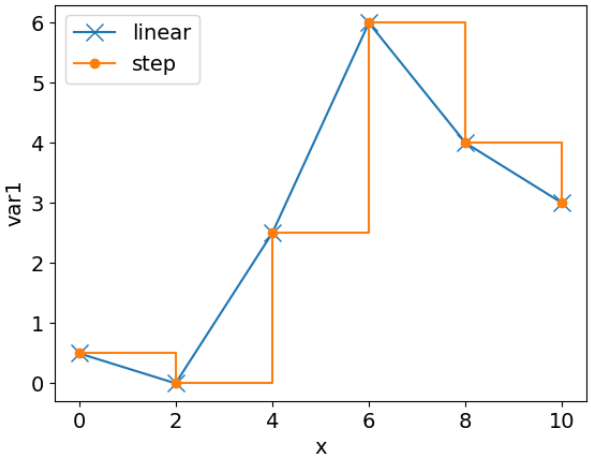


Figure 1: Profiles of `var1` when `interpolation_type = linear` or `step`.

The actual profile of `var1` is illustrated in figure 1 depending on `interpolation_type = linear` or `step`. Symbols represent the keyframe data, lines represent the interpolated data. Values of `var1` are kept constant outside the `x`-range. For example, consider the following keyframe file:

```
3, 1
<interpolation_type>
!x                      var1
4.0                     2.5
6.0                     6.0
8.0                     4.0
```

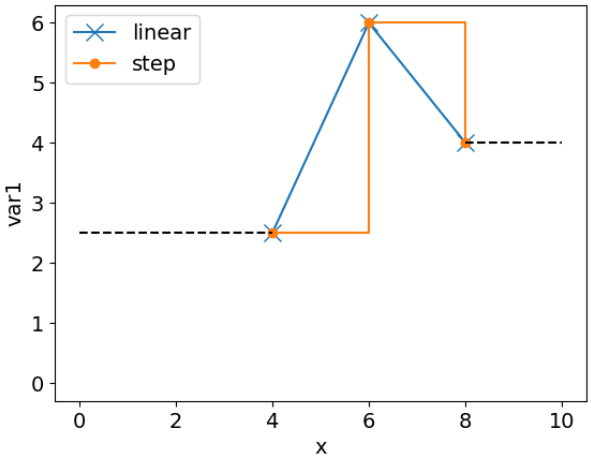


Figure 2: Value of `var1` outside the keyframe `x`-range is indicated with dashed line. Behaviour outside the specified keyframe `x`-range is independent of `interpolation_type`.

The full profile of `var1` in actual simulations is shown in figure 2. As seen, the first and last values of the keyframe values are extended outside the `x`-range (dashed line), both for `linear` and `step` interpolation schemes.

3 Problem setup

This tutorial performs a Discrete Element Method (DEM) simulation of particles transported using a screw feeder. This is a pure granular flow (the gas phase is ignored). There are two groups of particles. Regular particles and controlled particles. Regular particles are initially arranged into a structured array inside the hopper. Their position and velocity is determined from the usual DEM method. Controlled particles representing the screw geometry are assigned a position and velocity so they represent a rotating screw. The two groups of particles are differentiated inside the code by assigning a different scalar value for the screw particles and the regular DEM particles. The screw particles do not undergo position updates due to particle collisions but rather a prescribed position update based on the screw speed. On the other hand, the regular particles undergo collision and their positions evolve accordingly.

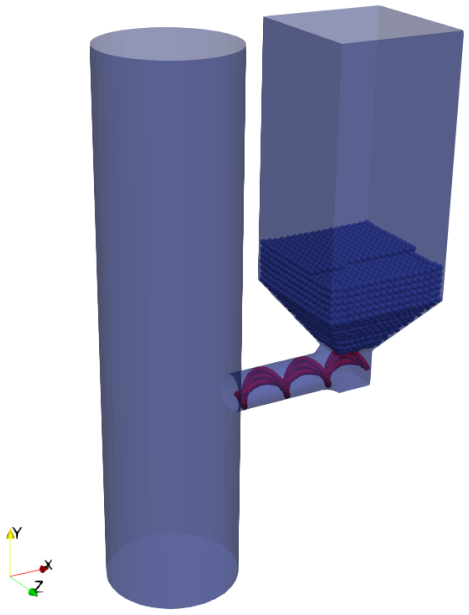


Figure 3: The screw feeder with the screw made of particles in red and the regular particles in blue.

The schematic of the screw feeder is shown in figure 3. The screw motion will depend on keyframe file `data_kf_0001.txt`, which contains the time (first column) and screw velocity (rotations per second) in the second column. For each simulation timestep, the rotational speed of the screw is linearly interpolated between keyframe values. We briefly explain the functions of different files available for this tutorial:

- `usr_mod.f` - contains all the necessary functions for reading and interpolating keyframe data.
- `usr0.f` - calls the function to read keyframe files and allocates required variables. Also, a scalar variable `DES_USR_VAR` is used to distinguish particles making up the screw and the other regular particles. Screw particles identified using $y < 0.1$ are flagged with `DES_USR_VAR=1` whereas default value for the scalar is zero.
- `usr1_des.f` - utilizes the interpolated data (revolutions per second) and converts it to radians per second.
- `cfnewvalues.f` - contains routines to describe the collective screw-like behaviour for particles with `DES_USR_VAR=1`. Compare with original file at `mfix/model/des/cfnewvalues.f`.
- `des_time_march.f` - the call for `usr1_des` is moved earlier (compare with original file: `mfix/model/des/des_time_march.f`). This enables proper use of interpolated data for a given time.

- `usr_init_namelist.f` and `usrnlst.inc` - Both files let the solver and GUI known about the new keyword `read_kf` which is an addition in this tutorial. This is a Boolean flag used in the UDFs to read a given keyframe file.

Interested users can study the above files to understand how the implementation works, and adapt it to their specific needs. Below is the keyframe file (`data_kf_0001.txt`) used in this tutorial:

```
5, 1
linear
! time    var1
0.0       0.0
1.0       2.0
1.5       2.0
2.0       0.0
3.0       2.0
```

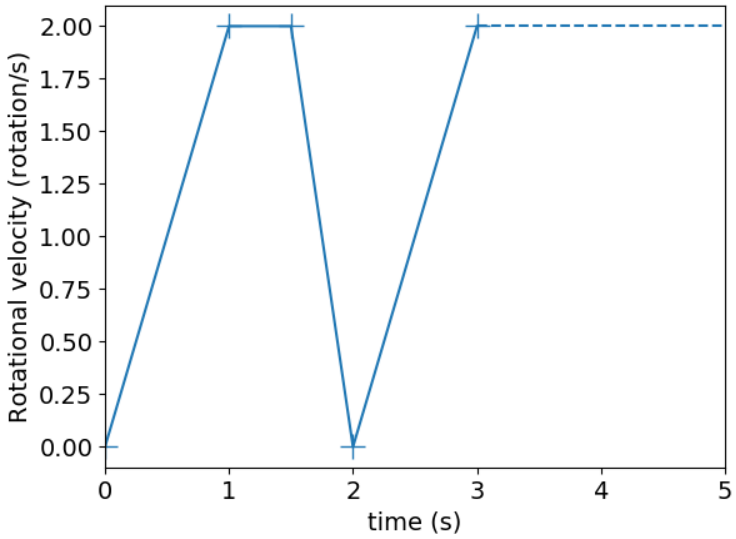
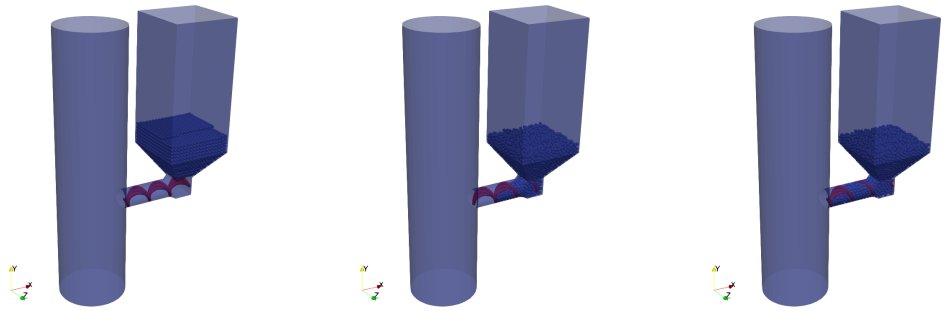


Figure 4: The rotational speed (solid lines) of the screw using the data (+) from keyframe file. Dashed line indicate the rotational speed extrapolated for remainder of the simulation.

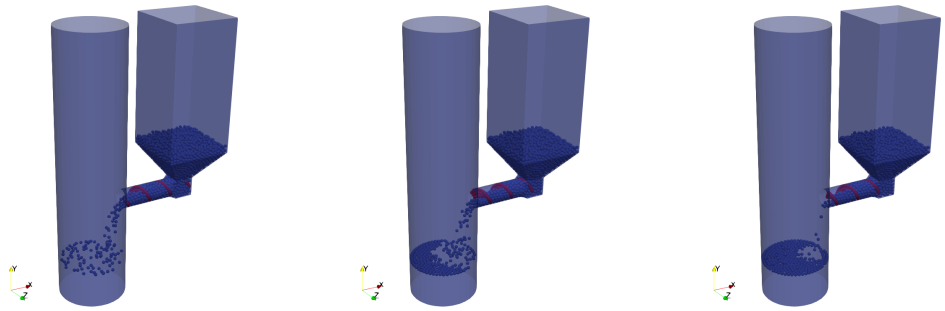
The plot of the screw rotational velocity based on the keyframe data is shown in figure 4. It should be noted that the velocity is specified only until $t = 3$ seconds in the keyframe file. However, the total simulation is run for 5 seconds. The keyframe UDFs are programmed such that it would continue using the last specified keyframe value throughout the rest of simulation.

4 Results

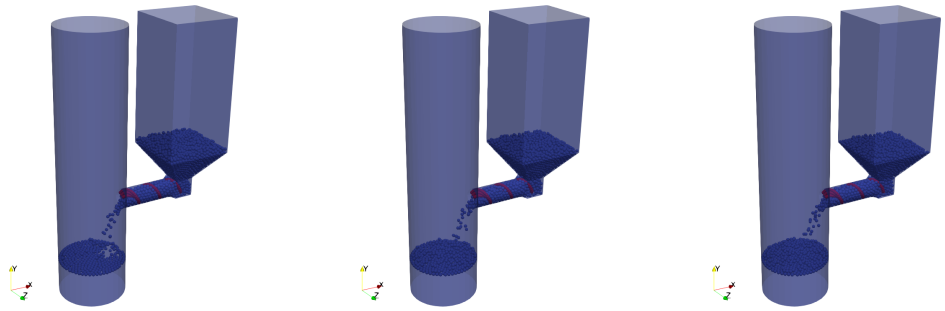
The granular flow due to varying rotational speeds of a screw feeder is shown in figure 5.



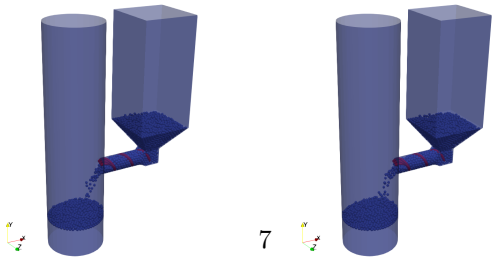
(a) $t = 0$s (b) $t = 0.5$s (c) $t = 1$s



(d) $t = 1.5$s (e) $t = 2$s (f) $t = 2.5$s



(g) $t = 3$s (h) $t = 3.5$s (i) $t = 4$s



(j) $t = 4.5$s (k) $t = 5$s

Figure 5: The granular flow of particles using the screw feeder at various timesteps. Particles making up the screw are shown in red and regular particles in blue (using `DES_USR_VAR`).

5 Notes

- Since this tutorial involves UDFs, the custom solver must be built prior to running the simulation.
- Once the custom solver is built, users can modify and save the keyframe file to adjust the rotational speed of screw. There is no need to build the custom solver when modifying the keyframe data.
- The user-defined keyword `read_kf` is accessible through the GUI Advanced pane. To completely turn off the keyframe data for the screw, set `read_kf(1)=False`.