MFiX Tutorial
Conveyor belts with varying belt velocities

Sathish Sanjeevi and Jeff Dietiker

National Energy Technology Laboratory, USA

November, 2019

1 Introduction

In this tutorial, two conveyor belts are simulated with varying belt velocities using user-defined functions (UDFs). The objectives of this tutorial are: 1) Show how to apply a tangential velocity along an STL file to represent moving walls. 2) Use keyframe data to specify transient velocities to each belt.

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
```

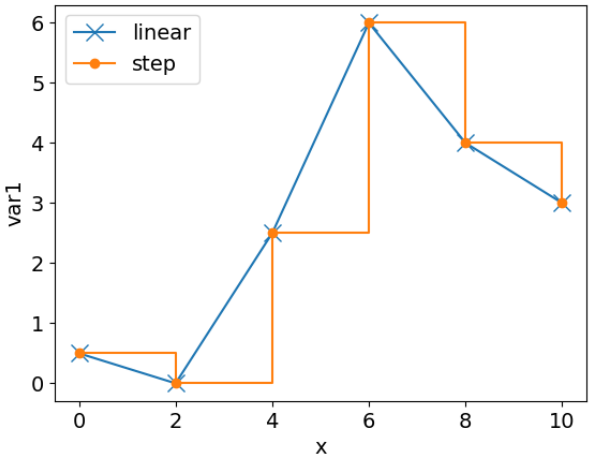


Figure 1: Profiles of `var1` when `interpolation_type = linear` or `step`.

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

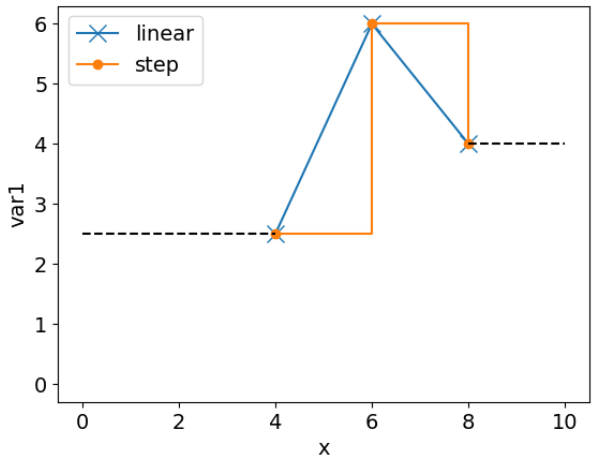


Figure 2: Value of `var1` outside the keyframe `x`-range is indicated with dashed line. Behaviour outside the specified keyframe `x`-range is independent of `interpolation_type`.

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 being transported by two conveyor belts. This is a pure granular flow (the gas phase is ignored). A mass inlet boundary condition drops particles on the top belt, which transports them in the negative z-direction. They then fall onto the bottom belt, which carries them in the negative x-direction. Each belt's velocity varies in time. The top belt velocity is initially zero and is ramped up to 3 m/s between 2 and 4 seconds. The bottom belt's velocity cycles between zero and 5 m/s over time (see also figure 4).

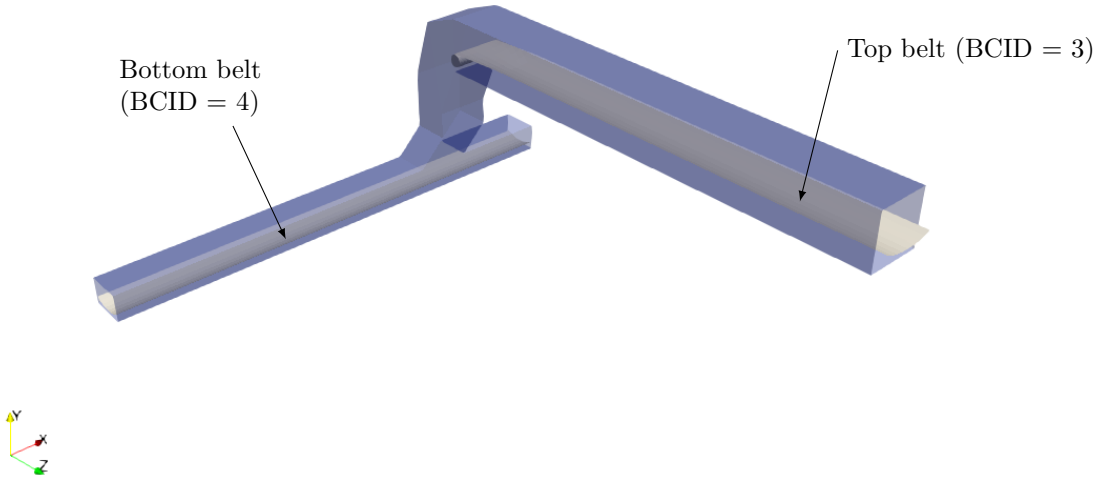


Figure 3: The top and bottom conveyor belts

The top and bottom conveyor belt have BCIDs 3 and 4 respectively as shown in figure 3. The corresponding `data_kf_<ID>.txt` files contain the time (first column) and belt velocity magnitude in the second column. For each simulation timestep, the velocities of the conveyor belts are 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. Also contains the functions for returning appropriate velocity corresponding to each belt.
- `usr0.f` - calls the function to read keyframe files and allocates required variables.
- `usr1_des.f` - utilizes the interpolated data (velocity magnitude) and converts it to velocity vector.
- `calc_collision_wall_mod.f` - assigns the belt velocity as tangential relative velocity for the wall boundaries. It should be noted that the belts (STL faces) do not move in our simulation. Rather, the belt motion is mimicked by adding tangential velocity (i.e. belt velocity) to the wall boundary (belt).
- `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 are the two keyframe files used in this tutorial:

`data_kf_0003.txt`:

```
4, 1
linear
!time    var1
0.0      0.0
2.0      0.0
4.0      3.0
10.0     3.0
```

`data_kf_0004.txt`

```
25, 1
linear
!time    var1
0.0      0.0
4.0      0.0
5.0      5.0
9.0      5.0
10.0     0.0
14.0     0.0
15.0     5.0
19.0     5.0
20.0     0.0
```

(only first 12 lines shown)

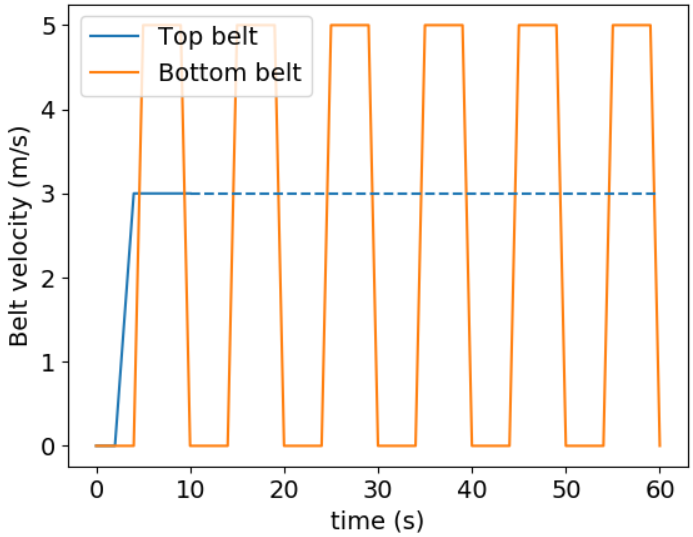
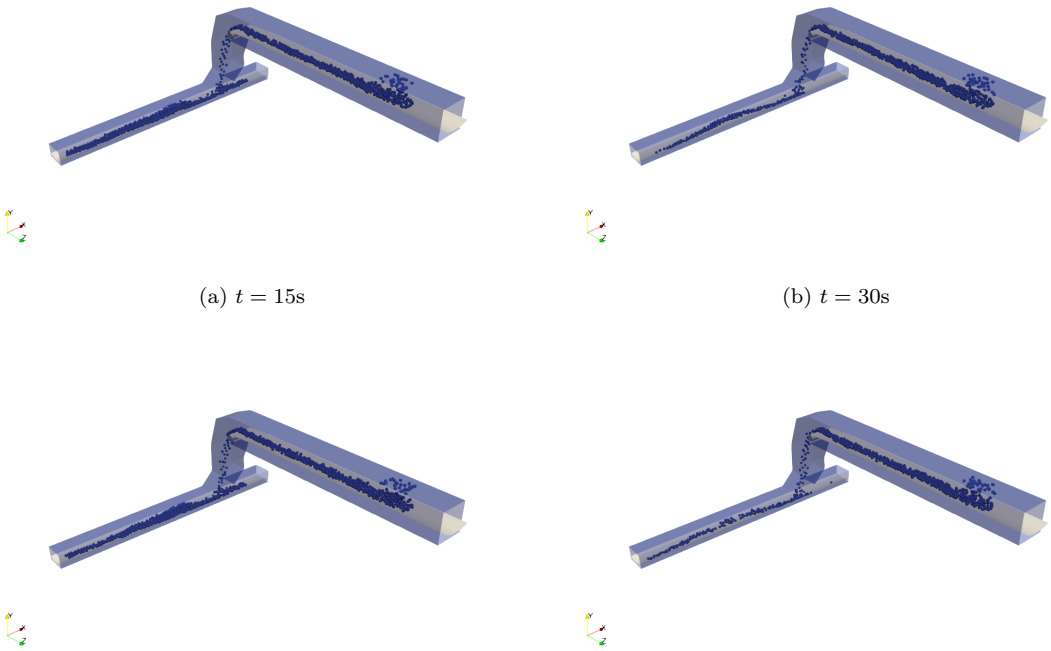


Figure 4: The specified velocities (solid lines) for the top and bottom belts using the keyframe files. Dashed line indicate the velocity extrapolated for rest of the simulation.

The plot of the specified velocities for the top and bottom belts based on their keyframe data

are shown in figure 4. It should be noted that the top belt velocity is specified only until $t = 10$ seconds. However, the total simulation is run for 60 seconds. The keyframe UDFs are programmed such that it would continue using the last specified value throughout the whole simulation. In other words, the top belt velocity would be maintained 3m/s even for $t > 10$ seconds.

4 Results



(a) $t = 15$s (b) $t = 30$s

(c) $t = 45$s (d) $t = 60$s

Figure 5: Snapshot of conveyor belts at different time. The varying belt velocities, specifically the bottom belt, can be observed at different timesteps.

The granular flow due to varying velocities of conveyor belts is shown in figure 5. In our case, specifically the bottom belt has a lot of velocity fluctuations which would result in varying particle concentrations in the bottom belt. The same is observed in our results with higher concentration of particles at low belt speeds and vice versa at high belt speeds.

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 files to adjust the belt velocities. There is no need to build the custom solver when modifying keyframe data.
- The user-defined keyword `read_kf` is accessible through the GUI Advanced pane. To completely turn off the keyframe data for a belt (say the bottom belt), set `read_kf(4)=False`. This will set the default zero velocity for the bottom belt.