Comparison of Matlab:Animatlab Loop Timing Methods

Lesson # 1: Matlab is not good at maintaining a timer - including those Matlab functions designed explicitly to handle timing. Significantly affected by what else is going on on the computer.

Lesson # 2: Animatlab has at least 3 different time steps you can change. As far as I can tell, any true timing limitations have to be handled by the hardware (or in this case Matlab) controlling the timing of serial transactions. Animatlab’s time steps can be used to align Animatlab’s apparent Simulation timing to the hardware, and that’s about it.

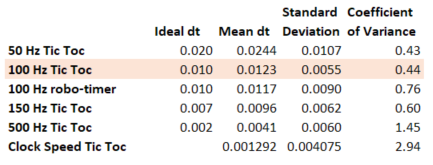
Note: Ask Wade about his jitter values for hardware timing at 150Hz. Better or worse than the Matlab timer? Add note here.

Some Background:

I tested 2 different ways of performing timed code execution to perform serial transactions at a fixed frequency.

* Robotics toolbox - rate object: super easy to use, but limited at 100 Hz
* tic and toc functions: can be used to simply time execution of a code segment, but I used it inside the serial loop to force the program to pause and wait until the dt is reached for the time step. Simple to set up, but has primarily one-sided positive error. Limited only by Matlab’s base (minimum) execution speed.
* There’s also a timer object in Matlab that I looked into, but I found it super confusing. I couldn’t figure out a way to get data back out of the required callback function, so I gave up. Also limited to 100 Hz.

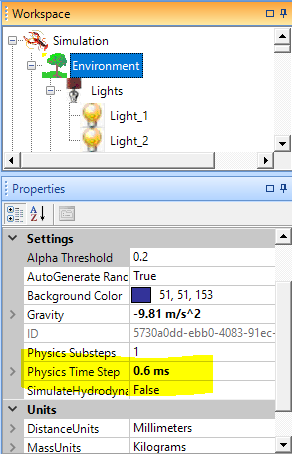
I did 5 runs each with 500 samples per run of the tic toc timer method at 50, 100, 150, 500, and 1000 Hz, and 4 runs of “No Timer” (only 4 runs, because clearly inferior). I also did 5 runs with the robo-timer method at 100 Hz for comparison. I looked at the statistics at each frequency, and determined that the tic toc method begins to break down, I.e., the coefficient of variance, standard deviation relative to the mean, gets larger as the frequency increases. Here is a summary of the results. The corresponding data is in the “Final Timing Test” Excel file.



50 Hz and 100 Hz were nearly identical in terms of coefficient of variance, but the 50 Hz runs were painfully slow. The Robo-Timer at 100 Hz no better than the tic toc timer, and tic toc timer runs above 100 Hz got worse. **So, I stuck with the 100 Hz tic toc timer.**

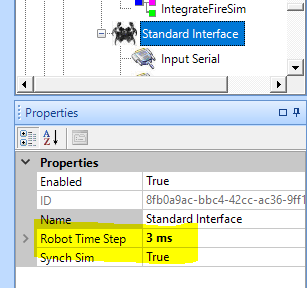
Now, how to make Animatlab’s simulation follow the same time axis as the hardware/simulation.

First, the Physics Timestep



It seems that this needs to stay pretty small for Animatlab’s simulation to work properly. Changing it doesn’t change the time scale on the simulation.

Next, the Robot Timestep



Documentation says this should be just large enough to accomodate the hardware’s slowest process. It never made much difference for me, but I’m using Matlab simulation, not hardware, although it does slightly change the time scale of the simulation. It doesn’t change it enough to fully match everything up.

Finally, the Playback Timestep

* Set Playback Control to Use Preset Value
* Then set Playback Timestep

For me, Setting Playback Control to match the physics time step results in the simulation being way off my Matlab simulation. This makes sense, because we’re not transmitting a time axis over serial to AM, only values

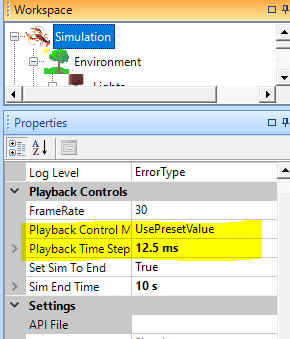
My final settings:

For running Matlab simulation with tic toc 500 Hz timer, these settings corrected my time axis:

* Physics Timestep: 0.6 ms (default)
* Robot Timestep: 3 ms (since 2ms are required for each loop/serial transaction)
* Playback Timestep: 12.5 ms

For running Matlab simulation with tic toc 100 Hz timer, these settings corrected my time axis:

* Physics Timestep: 0.6 ms (default)
* Robot Timestep: 10 ms
* Playback Timestep: 3.4 ms



This is where you can fully tune the simulation time scale to match that of the hardware/simulation. For me, increasing the playback timestep decreased the length of the simulation time axis my simulation would cover.

Note, there was nothing I could do to change the timestep for the Animatlab simulation, it was always ~0.0002. However, changing the playback timestep seems to scale the IO data over the default timesteps. Essentially, this is only a visual manipulation of the simulation data, not of the simulation mechanism itself.