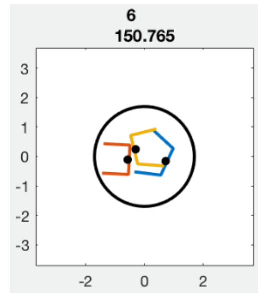
Notes on construction and running of the Smarticles simulation code

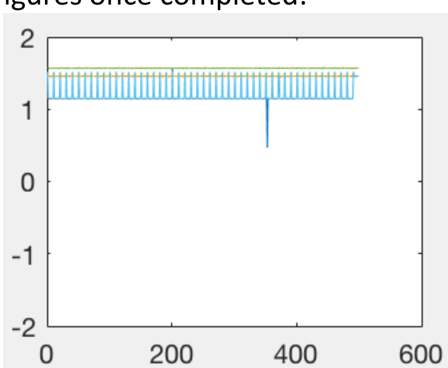
For questions, contact: Pavel Chvykov pchvykov@gmail.com
Updated Sept 22, 2020

Running the simulation:

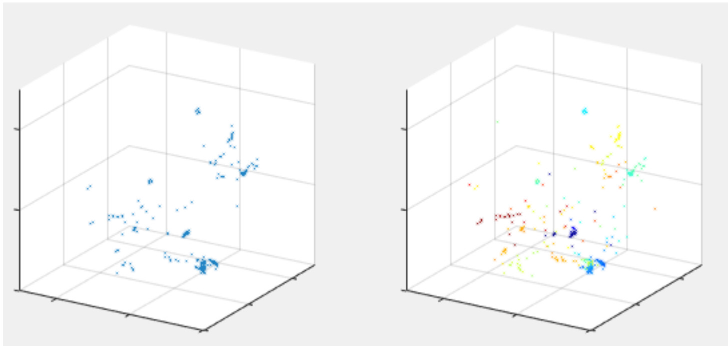


To run the simulation, directly run the **smarticles_super3.m** file. As it is currently set up, it should immediately show a live visualization of the 3-smarticle simulation executing the distinct period-3 gait used for several of the paper figures (designated Drive A in Fig. 3) – should look like the picture on the right. We have provided compiled _mex files for Mac and Windows which should allow the code to run as is. If there are issues or if you're using another OS, refer to the code generation instructions at the end of this file.

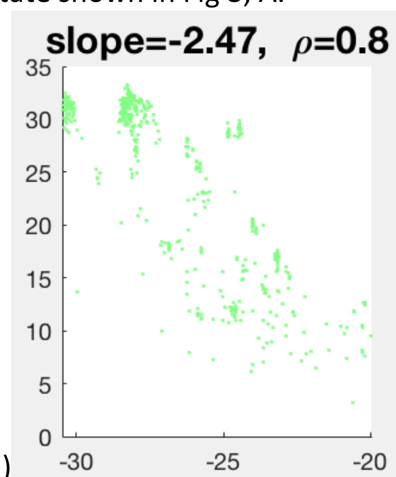
The visualization is slow to render – simulation can be sped up significantly by showing it less frequently. For this change fpp (frames per period) variable from 24 to 1/24 (or turn off entirely by setting livePlot=0). Once the simulation finishes at current settings, (50 runs, with 40 periods each), it will automatically run the data analysis code densVrattCorr1.m, which should produce 3 figures once completed:

1) 

a sanity check, showing that the stroboscopic configuration selection chooses time-points when arms are in U-shape: $(\alpha_1, \alpha_2) \approx \left(\frac{\pi}{2}, \frac{\pi}{2}\right)$.

2) 

steady-state distribution in 3D configuration-space projection on the left (compare to Fig 3, A), and same points, colored by rattling on the right. The sampling of this run isn't quite good enough to produce the figures in the paper, and the relaxation time to steady-state was not quite long enough, but we already see that low-rattling configurations agree with the steady-state shown in Fig 3, A.

3) 

The log steady-state density vs rattling correlation plot, as in Fig 1 bottom, with linear-fit slope (γ) and correlation coefficient shown above. In this simulation the run lengths were chosen such that both the steady-state regions and the rest of the configuration space both get sampled appropriately, and so the correlation is clearly seen – but this may not be the case for other parameters, and the plot might only show the left or the right end of this correlation. To do this reliably, we should sample the steady-state distributions, and the rest of the space in two separate simulation runs, and combine the results (see fig.S1 D,E).

Varying parameters:

File: smarticles_super3.m (running simulation)

Variable name [default value] – explanation

- A [0.9] – Smarticle arm length
- B [1] – Smarticle body length
- latFric [0.2] – lateral friction parameter (amount of smcle-smcle friction)
- Nsm [3] – number of smarticles
- tRes [2*pi/200] – time-step (can in principle change this from ~0.03, but the simulation may be brittle at other values)
- prd [2*pi/tRes=200], stRes [prd/50=4] – period length and time-resolution, for data storage ONLY (do not affect simulation)
- nCyc [30] – number of 2*pi periods in one continuous run (choose multiples of 3 for stability)
- t – time-domain for one realization. If this is some multiple of what is specified by nCyc, the simulation will rerun from the same IC that many times (with different noise realizations)
- windSize [`(B+2*A)/3*sqrt(Nsm)*1.05`]– window in which smcles are randomly initialized. If ring is present, sets ring diameter also
- fricR [Inf], dragRv [0] – ring friction coefficient and constant translation velocity (externally dragged)
- inertCoeff [0] – smcle inertia – keep this fraction of velocity from last tick (1-frictionless, 0-overdamped)
- resDist [100] – limit to how far a smcle is allowed to move in one tick (lower => more smooth simulation, but more likely to jam)
- livePlot [1], plFrom [0], fpp [24] – flag to show live plot during simulation, starting at time plFrom, at resolution of fpp frames per period
- T [1E-3] – amplitude of additional Brownian noise on smcles (temperature)
- armAmpRnd [0] – amplitude of random variation in gait amplitude (as arms don't go exactly to 90 deg each time)
- armSpdRnd [0] – amplitude of random fluctuations in motor speed
- ordIC [false] – false: initialize smcles randomly, true: initialize by picking configurations from existing run with experiment index oIx
    - false: if nullCrd exists, initializes from these coordinates, else drops smarticles at random and resolves collisions [latter may introduce certain biases in configuration sampling]
- expIx – loop over different experiments, labelling stored data by this index. There are a number of pre-set gaits that can be selected by this index:
    - expIx=1: random gait (Fig.4 right column, fig.S6, Drive 1)
    - expIx=2: random gait, correlated across the ensemble (fig.S6, Drive 2)
    - expIx=3: square gait (Fig.2, fig.S6, Drive 3)
    - expIx=20: Drive A gait (Fig.3 A, fig.S6, Drive A)
    - expIx=4: period-3 distinct periodic gait, pattern determined by the rng(.) seed

  - expIx > 100: period-4 distinct periodic gait, pattern determined by the rng(.) seed
  - $4 <$ expIx ≤ 5: add random moves to base deterministic gait at expIx=4 (so expIx=5 – fully random, same as expIx=1) – Fig 4.
  - $20 <$ expIx < 21: Drive A gait, increasing smarticle inertia to frictionless at 21
  - expIx=21: mixed gait (mix Drive A with gait set by rng(.) seed) – Fig.3C
- repIx – loop over different realizations of the same experiment
- fricCoeff [0.1] – list of friction coefficients for different smarticles
- Nwall [0] – number of immobile "wall" smarticles (currently only objects allowed in simulation is circular ring and smarticles – so can build some walls / mazes / geometry out of immobile smarticles, setting their fricCoeff high and arm angles fixed
- patt – defines gait pattern – sequence of arm angles, which are traversed at some max motor speed maxV. tStep – dwell-time at each arm-coord; nSeg – pattern length, set to large for fully random gait. nRet – periodicity with which we return to U-shape

Output data arrays

------raw data: --------

- tAll – timing data, 3 columns giving: 1 -- run-time; 2 -- run index; 3 -- experiment index
- crdDatAll – 3D array with Smarticle coordinates: dim1 – Smarticle index; dim2 – (x, y, \theta, \alpha_1, \alpha_2): coordinates, orientation, and arm angles; dim3 – time
- ringAll – ring coordinates

------processed data: --------

- piDatAll – rotation-invariant observables, as described in Materials and Methods. dim1 – observable in this order: $\left\{\chi_1^{(y)}, \chi_2^{(y)}, \dots, \chi_1^{(x)}, \chi_2^{(x)}, \dots, \chi^{(\theta)}\right\}$, dim2 – time

File: densVrattCorr1.m (analyzing data)

- oEx – index of the experiment to compute the steady-state densities
- nEx – index of experiment(s) to compute rattling (can be an array – then concatinated)
- crdCorr [100] – length of time-averaging for rattling calculation
- projSS – Boolean array giving the dimensions of piDatAll to include in configuration space
- oSt [150] – strobe interval to sample the steady-state
- selOPss – Boolean array to select time-points included in steady-state
- ordDat – configuration space points sampling the steady-state
- nullDat -- configuration space points sampling all regions of interest
  - nullDatFull – roll-out trajectories starting from each of these points
- seedDat – seed points for comparing rattling with steady-state (see Materials and Methods)

Code Architecture

smarticles_super3.m is the latest run-file for the simulation
It sets all the parameters, has loops for multiple simulation runs, defines the gait, initialized the smarticles, sets the confining potential / ring, runs the smarticle simulation in time-ticks, and saves data (some of it) – all in that order.

The main work-horse of the software is resolveCollisions_mex function – which resolves collisions in the ensemble of smarticles at each time-tick. This is a matlab-compiled (to C/C++) function, which uses the files:

- smcle2coord.m – convert the 5-tuple defining smarticle $(x, y, \theta, \alpha_1, \alpha_2)$ to (x, y) coordinates of each of the 4 smarticle corners [runs very often]
- pivotArm.m – uses analytical force balance calculation to approximate the coordinates of the pivot point for a single smarticle if it is pushed by the arm
- pairCollision.m – resolves a pair-wise smarticle collision: given two smarticle coordinates, check if they are colliding, and if so, return new coordinates
- pushBoundary.m – resolves collisions between smarticles and ring
- resolveCollisions.m – loops over all smarticle pairs and ring in random order, until all collisions are resolved. Implements some other routines to avoid/detect jamming and improve simulation stability (see Materials and Methods).

The _mex function is then compiled by Matlab Coder package, via the script:

```
codegen resolveCollisions -args
{coder.typeof(0,[1000,5],[1,0]),1,coder.typeof(0,[1000,1],[1,0]),coder.typeof(0,[3,3,1000,1000],[
0,1,1]),[0,0,0],0}
```

We similarly need to compile the smcle2coord_mex function using the line:

```
codegen smcle2coord -args {coder.typeof(0,[1000,5],[1,0])}
```

After the simulation finishes, some data processing is run by symmetrize_smarticle.m file – which constructs rotation-invariant configuration variables piDat out of the smarticle coordinates, as defined in Materials and Methods.

densVrattCorr1.m then analyses the data, calculating steady-states and rattling, making the configuration-space density plots, calculates rattling, plots correlations.