Description of the simulation code for a circular corral

Matthew Durey, Paul A. Milewski, Zhan Wang

This MATLAB code (which was written using MATLAB 2018a) constructs the fundamental matrices and evolves the iterative map for the pilot-wave system given in equation (2.16) of the main text. The user should modify and run the file `Example.m`. The user has the option of running code for a corral geometry considered in the paper, for which the Faraday threshold is provided in the data file `GamF_vec_freq80.mat` consistent with figure 4 (Option 1 – line 25), or choosing their own parameter values (Option 2 – line 33). In line 49, the number of angular modes m must be chosen, by defining the integer `m_max`. The larger the corral radius, R_c, the larger the number of modes required to resolve the dynamics. For 5 mm $\leq R_c \leq$ 15 mm, we consider `m_max` in the range 15 to 30. Note that the larger the value of `m_max`, the slower the code runs. Finally, the user has the option of changing the droplet properties, such as the radius, impact phase or skidding friction coefficient. The default values are those used in the paper.

The first stage of the computation is the assignment of initial conditions, as specified in line 85 of `Example.m`. The initialisation set here is identical in form to those used in the simulation study presented in §5 of the main text. Next, the fundamental matrices, $\boldsymbol{M}_m$, are computed in the program `get_fundamental_matrices.m`. The user need not modify this program, which follows the algorithm described on page 10. Specifically, for each angular mode, m, zeros of Bessel functions are first computed using `zerobess.m` (available for download from MATLAB Central). Then the LU-decomposition of the mass matrix for the DtN computation (see equations (2.7)–(2.9)) is performed by the function `make_DTN_matrices.m`. Then the matrix $\boldsymbol{M}_m$ is computed, iterating over the program `ODE_RK4.m`, which should not be modified. Each fundamental matrix is stored in the cell array `Mat_cell`. Last, a vector of memories, $\mathcal{M}_m$, is computed based on the eigenvalues of $\boldsymbol{M}_m$, where these memories are stored in the vector `Me_vec` (this is an informative quantity and is not used anywhere in the simulation).

Following this pre-computation and initialisation, the iterative map (2.16) is computed in `Run_simulation.m` over the number of jumps `num_jumps` prescribed by the user in line 23 of `Example.m`. This program calls the function `get_grad_wavefield.m`, which compute the gradient of the free surface beneath the droplet at each impact. Following the conclusion of the simulation, summary plots are presented in `Plot_simulation.m`. Each plot is described by the comments in the program.

If you have any questions about the functionality of this code, please e-mail Matthew Durey (`mdurey@mit.edu`).