README file for PFMCal.

A. Butykai, F. M. Mor, R. Gaál, P. Domínguez-García, L. Forró, S. Jeney

1. ORIGINAL `PFMCAL.M`

1.1 Software requirements

MatLab 2011a or later, Statistics toolbox. Tested in MatLab 2016a.

1.2 Installation of the software

Unzip `CalibrationSoftware.zip` archive to the destination path. No further installation steps are needed.

1.3 Executing the software

The calibration software can be executed from MatLab (MathWorks Inc.) by running the `PFMCal.m` file. Select either 'Add to path' or 'Change current directory' options.

1.4 Using the software

Please refer to Section 7 in the original manuscript: **Computer Physics Communications, 196 (2015) 599**

1.5 Test data

Test data are located in the `Testdata` folder. `Si_Spheres` archive contains experimental data measured on silica spheres. `Si_splinters_non_spherical` folder contains experimental data measured on fragmented silica particles with random shape. `Synthetic_non_spherical` folder contains 100 generated datasets with random error for the validation of the non-spherical model. `Synthetic_spherical` folder contains 100 generated datasets with random error for the validation of the spherical model.

2. EXTENDED VERSION

2.1 Software requirements

The extended code can be executed without modification using MatLab or GNU Octave. The code has been tested in Linux and Windows operating systems.

2.2 Installation of the software

Unzip `CalibrationSoftware.zip` archive to the destination path. The extended files are inside the `AUTO` folder. For running the files an installation of MatLab 2011 (or later) or GNU Octave 4.0 (or later) is needed.

Regarding Octave, the code uses `Forge`* packages `strut` and `optim`. Issues may appear in relation with those packages in Octave 4.0 (not expected in later versions). To avoid these errors, update the packages in Octave command window by using the following commands:

```
pkg install -forge struct
pkg install -forge optim
```

*`https://octave.sourceforge.io/`

2.3 Executing the software

The extended calibration software can be executed from MatLab command window or GNU Octave by running the files `PFMCal_auto.m` and `PFMCal_histo.m` located inside folder `AUTO` (Linux-style complete path: `./CalibrationSoftware/AUTO`). Therefore, for the semi-automatic process of calibration, enter in the MatLab/Octave command window:

>PFMCal_auto

For running the thermal noise statistics method, simply write:

>PFMCal_histo

There is no need to enter a path to the data files, because it is included in those scripts. The test data is loaded from the folder `Testdata` (Path: `./CalibrationSoftware/AUTO/Testdata`). The loading of the files have been written to be compatible with the different path separators depending on the operative system (tested in Windows and Linux). By default, the water data is selected, but it can be changed to acetone data by editing directly the main files `PFMCal_auto.m` or `PFMCal_histo.m`. Inside these two files, the input data needed for each running are defined. This input data are defined in the following sections.

2.4 Using the software

2.4.1 PFMCal_auto

The analytical solution of Langevin equations for an optically trapped spherical particle with no-slip boundaries in a Newtonian fluid with hydrodynamic effects only depends numerically on the following timescales:

$$\tau_f \equiv \frac{\rho_f a^2}{\eta};\ \tau_p^* \equiv \frac{m^*}{6\pi\eta a};\ \tau_k \equiv \frac{6\pi\eta a}{k} \tag{1}$$

where a is the bead radius and ρ_p its density, ρ_f is the density of the fluid, while $m^* \equiv m_p + m_f/2$ is an effective mass which modifies the mass of the particle, m_p, because of the influence of hydrodynamics, being m_f the mass of the displaced fluid. By knowing the characteristics of the beads and the fluid, such as a, ρ_f and ρ_p, the calibration method allows to calculate these timescales and, consequently, we will obtain the viscosity of the fluid, η, and the optical force spring constant, k.[†]

Additionally to these input values, we have to obtain the value of the MSD plateau which is going to be used in the computation of the timescales. The point to be selected must be in the region where the noise begins to be noticiable, to obtain in which time value this effect is important. and the election of this value has been observed to greatly vary the obtained η value. This point is automatically calculated by the function `getMSDPlateau`. The method calculate the second derivative of the logarithm of the MSD and then obtains a modulation function when adding a point to the array of data. More clearly, we define a function:

$$f(t) \equiv \frac{d^2}{dt^2} \log \mathrm{MSD(t)}$$

which is shown in Fig. 1 (left). This graph corresponds to the acetone MSD data contained in the `AUTO/Testdata` folder.

[†]For further details see: M. Grimm, T. Franosch, S. Jeney, High-resolution detection of brownian motion for quantitative optical tweezers experiments, Phys. Rev. E 86 (2012) 021912.

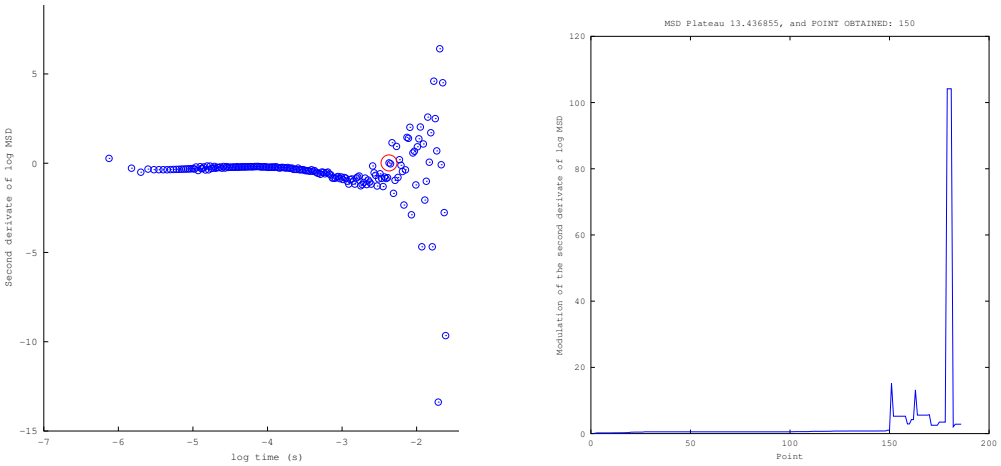

Figure 1. Left: Logarithm of the second derivative of the mean-squared displacement ($f(t)$ in the text), showing the noise affecting the calculation of the plateau of the MSD. The red circle marks the approximate point chosen by the user where the noise begins to be relevant. Right: Modulation function of $f(t)$ (M_n in the text): the beginning of noise is detected by choosing the array element (point) corresponding to the first peak in the figure.

The red point in the graph is calculated automatically by the code through a modulation function:

$$M_n \equiv \frac{\max(f_n) - \min(f_n)}{\max(f_n) + \min(f_n)}$$

where $n = 1..$lenght of $f(t)$. We obtain the value of n where the signal M_n grows abruptly by imposing a threshold value $M_n > 10$. Then, the code returns the point n when noise is relevant in the MSD and calculates the MSD plateau value. This can be shown in Fig. 1 (right). [‡]

After that, using the input data a, ρ, ρ_p, T and the obtained MSD plateau value, the software automatically calculates the calibration curves for the MSD and VAF, and it obtains β_{MSD}, β_{VAF}, k and η.

Regarding the PSD, we use the classic methodology for Newtonian fluids, to obtain the PSD corner frequency, f_c, and then β_{PSD} and k. The PSD of the bead in the fluid has a Lorentzian form PSD $(f) = S_0 f_c^2/(f_c^2 + f^2)$, where S_0 is a constant. This quantity can be approximated $\mathrm{PSD}(f) \simeq S_0 f_c^2 f^{-2}$ in the limit of high frequencies. Then, the calibration factor is $\beta_{\mathrm{PSD}} = k_B T/(6\pi^3 \eta a S_0 f_c^2)$. Note here that η and a have to be previously known to obtain this β_{PSD}, and therefore we use the η value obtained from the calibration of the MSD and VAF.

The execution of `PFMCal_auto.m` finally provides Fig. 2, plotting the experimental MSD and VAF and their theoretical curves. It also plots the experimental PSD and a fitting to the Lorentzian curve. The characteristic timescales and time limits are plotted with vertical lines. This graph is automatically saved in a folder created by the software named `output`. In this folder we save the calibration graph in 'pdf' and 'eps' formats and a text filed named `output_data.txt` which contains all the input and output data.

[‡]This calculation is not shown during the execution of the code but these graphs can be shown and saved by changing a boolean parameter in `getMSDPlateau` function.

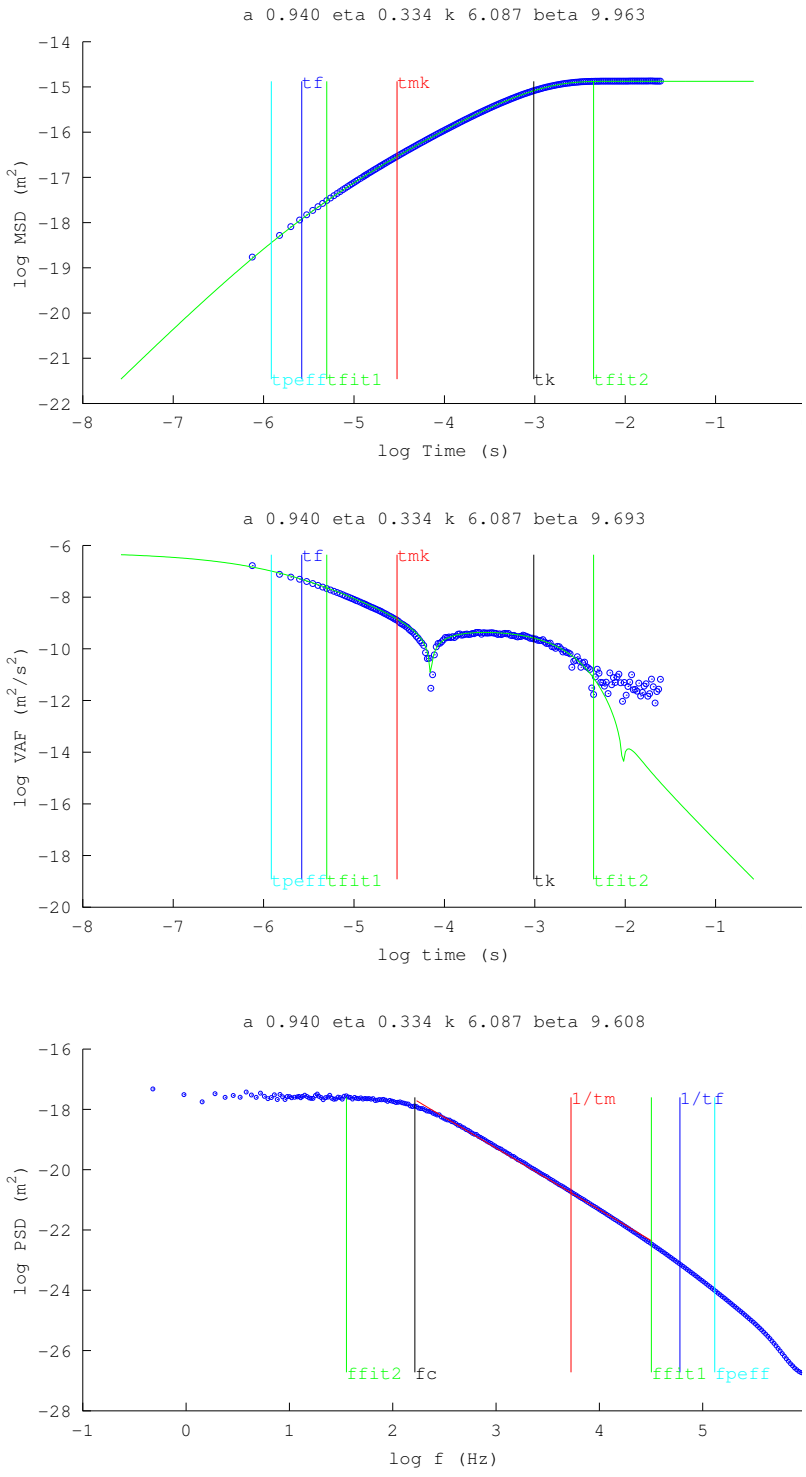

Figure 2. Experimental MSD and the VAF with their theoretical curves. The experimental PSD is plotted with a fitting to the Lorentzian curve. Vertical lines: tf corresponds to τ_f, tpeff is τ_p^*, tk is τ_k. The two time limits defining where the methods are applied are tfit1 and tfit2.

2.4.2 PFMCal_histo.m

Additionally to the methods explained above, we also include a calibration method by using the experimental histogram of particle positions. From the probability density $p(x)$, the potential can be deduced as $E(x) = -k_B\,T\,\ln p(x) + C$, where C is a constant related to the potential offset and can be neglected. In the case of a spherical bead in an ideal optical trap, $E(x) = \frac{1}{2}k_E^2 x$, only depending on the trap stiffness, k. Here, $p(x)$ is a Gaussian distribution and, through the equipartition theorem: $\frac{1}{2}k\left\langle x^2\right\rangle = \frac{1}{2}k_B T$, we have $k_{\sigma^2} \equiv k_B\,T/\sigma^2$, where $\sigma^2 = \left\langle x^2\right\rangle$ is the variance of the probability density.

If we use some additional method to obtain a value for the trap stiffness, i.e., by calculating averaged values from the calibration of MSDs for independent water measurements, we can calculate new calibration factors, β_E, from the harmonic potential curve and β_{σ^2}, from the Gaussian distributions.

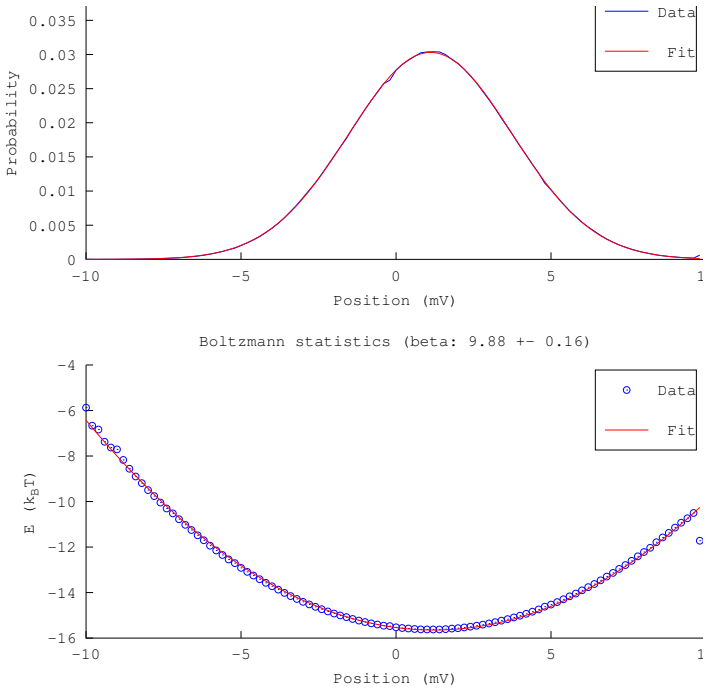

Figure 3. Calibration factors obtained from the fitting to a harmonic potential curve, β_E, and from Gaussian distributions, β_{σ^2}.

The fittings and calculation of these calibration factors is done by using the script `PFMCal_histo.m`. The histograms data are read from the `AUTO/Testdata` folder and the k value is defined inside `PFMCal_histo.m`. This code fits the data to a Gaussian function and to a harmonic potential, obtaining very similar calibration values between them and in comparison with the calibration methodology of `PFMCal.m` and `PFMCal_auto.m` (see Table in the new paper submission). The fitting graph is like Fig. 3, which is saved in the `output` folder. The input and output data regarding this calibration code are saved in a file named `output_data_histo.txt`.

All the fittings results (texts and graphs for water and acetone data) are shown in the folder `./AUTO/results_auto`.