Pair correlation microscopy of intracellular molecular transport

Julissa Sanchez Velasquez, Ashleigh Solano, Michelle A. Digman, Enrico Gratton, Francesco Cardarelli, Elizabeth Hinde

The given code for Pair correlation microscopy is a source designed for researchers with an interest in quantitatively mapping the diffusive route a population of fluorescent proteins adopts with respect to live cell architecture with single-molecule resolution. The code was developed to promote and facilitate studies characterizing the directionality of intracellular transport, such as nucleocytoplasmic transport, and the diffusion law of a protein's trafficking event, such as DNA target search. The intent is that the presented protocol simplifies the analysis and guides the interpretation of pair correlation microscopy-derived data. Improvements and open-source data will be continuously included to make the method more widely accessible and, with this, help increase the understanding of how intracellular diffusion regulates biological function at the single-molecule level.

Files required

To start using the code for pair correlation microscopy, the "bfmatlab" folder, which is required to open TIFF files, needs to be downloaded. "bfmatlab.zip" can be downloaded from "https://www.openmicroscopy.org/bio-formats/downloads/". After downloading "bfmatlab.zip", unzip the folder and save it to the folder containing all MATLAB functions for pair correlation microscopy.

Getting Started

Confocal calibration

Open the file called "*Confocal_Calibration.m*" and follow the steps indicated within the code. Details are provided below:

1. Run Step 1 to define the folder with all MATLAB functions.
2. Run Step 2 to open the CSV file with the FCS data. The code assumes that the .csv file consists of three columns, where the third column represents the FCS data. If the input is arranged differently, make the respective changes before running this section.
3. Run Step 3 to calculate the autocorrelation function (ACF) and get the fitting parameters. For this, indicate the sampling frequency, *e.g.*, if a pixel dwell time of 10 μs was used, the sampling frequency would correspond to 100 000 s^{-1} ($1/10^{-5}$ s). Enter this value into the function as indicated:

```
[corrBins, SP_FCS] = SinglePointFCS(SP,100000);
```

NOTE: The fitting of the ACF is based on a molecule with a known diffusion coefficient (D), fluorescein with D = 440 μm^2 s^{-1}. If a different diffusion coefficient is used, make the respective modification in the file called "*FitCalibration.m*". This file can be found within the "*Calibration functions*" folder. Within this code, change the value "D" in line 55 (which is set to 440) with the desired diffusion coefficient.

4. Record the values for the point spread function (PSF)'s dimensions: the table called 'PCF_dim' contains the radius in the focal plane e^{-2} (ω) and the axial e^{-2} beam dimension (z).

pCF workflow

Open the file called "*PCF_workflow_v2.m*" and follow the steps indicated within the code. Details are provided below:

1. Run Step 1 to define the folder with all MATLAB functions.
2. Run Step 2 to open TIFF files. In this section, select a single or multiple TIFF files.
3. Run Step 3 to visualize intensity carpets before applying a detrending method. Enter a desired name in the following line within the code to save the intensity carpets to a .pdf file.

```
pdfFileName='Enter File name';
```

e.g., `pdfFileName='eGFP-mCh';`

4. Run Step 4 to detrend the data. In this step, the user can select a suitable temporal window to be used by the detrending method. Enter the desired value in the following line within the code:

```
binsize = 2000;
```

5. Run Step 5 to (*i*) remove the first 20 000 rows and (*ii*) apply a period to average equal to 100 lines (for plotting). In this section, the users can decide if they want to eliminate the first 20 000 rows by modifying the following line within the code:

Set "`skip_crop = false;`" if you want to remove the first 20 000 rows

Set "`skip_crop = true;`" if you do not want to remove the first rows

6. Run Step 6 to visualize intensity carpets after detrending. Enter a desired name in the following line within the code to save intensity carpets to a .pdf file.

```
pdfFileName='Enter File name';
```

7. Run Step 7 to indicate the first and last columns to be analyzed by the pCF. After running Step 7, two matrices called "mfirstCol" and "mlastCol" will be created in the workspace. Enter manually the first and last columns to be analyzed in those matrices (fill the first column of the matrices only). For example, if the user opened five TIFF files and wants to analyze the pCF from the first column to the 20th column for all five carpets, then the first column of the "mfirstCol" matrix should contain the values [1;1;1;1;1] and the first column of the "mlastCol" matrix should contain the values [20;20;20;20;20]. Take into account that not all five carpets need to have the same values regarding the first and last column to be analyzed, the choice depends on the user.

8. Run Step 8 to establish the input parameters to calculate the pCF. Please change the following values accordingly:

   `radius = 0;` Set the value to 0 to get the ACF. To calculate the pCF indicate the distance (in pixels) that will be analyzed. For example, if pCF6 wants to be computed, make "radius = 6".

   `sampleFreq = 616;` Change the sampling frequency according to your experiment.

   `ReverseOrder = false;` Write "true" for performing pCF in the reverse order, write 'false' to perform the pCF in the direction of the acquisition.

   Remember to activate or silence the following lines within the code (by removing or adding "%") according to the analysis that you want to perform:

   (i) If *radius* was set to 0, then activate "A_avg = cell2mat(XCF_av);" and silence "%pCF_mean = cell2mat(XCF_av);"

   (ii) If *radius* was set to a value different than 0, then silence "%A_avg = cell2mat(XCF_av);" and activate "pCF_mean = cell2mat(XCF_av);"

Optional: Make changes in the following lines within the code if you want to visualize the correlation carpets.

a) Activate "ACF_carpet = XCF_ret" and silence "%PCF_carpet = XCF_ret;" if you just calculated the ACF, or activate "PCF_carpet = XCF_ret" and silence "%ACF_carpet = XCF_ret;" if you just calculated the pCF.
b) Visualize the correlation carpets by running:

   `"avgData_S = plot_ACF_carpets_Smoothing_v9(ACF_carpet);"`

   After running this line, the code will ask if you want to smooth the carpets. If you select "yes", a new window will appear where you need to specify: (*i*) if you want smooth columns or surface by entering "1" or "2", respectively; (*ii*) the sigma value for smoothing (we recommend to start with 0.9, evaluate the carpets, and then increase the value if necessary); and (*iii*) indicate the number of columns to be processed. *e.g.,* if you performed pair correlation analysis in Step 8 using only 10 columns, then you need to indicate this number.

   Remember to indicate which correlation carpets you want to visualize:

   `"avgData_S = plot_ACF_carpets_Smoothing_v9(`**`ACF_carpet`**`);"` to visualize ACF carpets.

   `"avgData_S = plot_ACF_carpets_Smoothing_v9(`**`PCF_carpet`**`);"` to visualize pCF carpets.

c) The code will also give you the option to plot the average smoothed correlation carpets by running this line:

   `"plot_avgData_S(avgData_S, corrBins)"`

9. Run Step 9 to calculate the s.e.m. for each ACF carpet. Run this subsection in both cases whether you want to perform a weighted fit or not. This data is required as input for the fitting functions.

10. Run Step 10 to visualize the ACF before fitting to review the quality of the data.
11. Run Step 11 to fit the ACF to model functions. By running this section, a new window will appear. Select the theoretical model to be used. After selecting the theoretical model, you will be asked to indicate if the fitting will be performed using a weighting method or not; answer yes or no in the command window. If no answer is provided, the fitting will be performed without using weights.

    NOTE: Step 11 assumes ω = 260 nm and z = 780 nm as fixed parameters. If after performing the confocal calibration, different values for the PSF's dimensions are obtained, change the values for ω and z in the following files "Fitting_Anomalous_G0.m", "Fitting_G0.m", "Fitting_pCF.m", and "Fitting_two_components.m" placed inside the folder called "Fittting pCF codes". Within these codes, the initial values for the diffusion coefficient and amplitude can also be modified, though we recommend using the default values, evaluating the fitted curves, and then modifying those values if needed.

12. Run Step 11.1 to get fitted data for a one-component model. Run this subsection only if a **one**-component model was chosen.
13. Run Step 11.2. to get fitted data for a two-component model. Run this subsection only if a **two**-component model is chosen.
14. Run Step 12 to fit the ACF to an anomalous diffusion model. After running this subsection, indicate if a weighting procedure will be used as indicated before.
15. Run Step 12.1 to get fitted data for anomalous diffusion. Run this subsection only if an anomalous diffusion model is chosen.

    NOTE: Post-processing fit results. After executing the fitting algorithm, the code will generate a table containing all fitted parameters ("*parameterTable*"): diffusion coefficient, amplitude, and anomalous factor if a model including an index of heterogeneity was used ("*parameterTable2*"). Evaluate the results and discard those values with no biological relevance (*e.g.*, negative amplitudes and extremely high/low diffusion coefficients).

16. After evaluating the ACF, return to *Step 8* and change the input parameters to calculate the pCF. The main change should be changing "*radius*" to a value different than 0, and silencing "%A_avg = cell2mat(XCF_av);" and activating "pCF_mean = cell2mat(XCF_av);". After making those modifications, run *Step 8* again.
17. Run Step 13 to fit the pCF to a model function. After running this subsection, the user will need to indicate the distance in nm in the command window. *e.g.*, if pCF12 is being calculated and a pixel size of 80 nm is being considered, then the value to be entered should be 960 (80 nm x 12).
18. Run Step 13.1 to get the fitted data after fitting the pCF to a model function.
19. If the model function does not explain the pCF, run Step 14 to fit the pCF to a Gaussian. This subsection will fit the pCF to a Gaussian model using the *fit* function in MATLAB. Consider that if two molecular species differ in diffusion coefficient by more than a decade, they will appear as separate peaks in the pCF profile. Hence, indicate if the fit will be performed using either one or two-term Gaussian models by answering gauss1 or gauss2 in the command window. If no

answer is provided, the code will fit the pCF to a one-term Gaussian model. We recommend using one Gaussian model, visualizing the fitting, and deciding if the data requires a two-term Gaussian model.

20. Run Step 15 to plot the raw and fitted pCF data, and get the peak values.
21. Run Step 16 to create a new table with the peak values in a linear scale.
22. If fitting the pCF to a Gaussian using default starting values and constraint bounds results in inaccurate results (*i.e.*, the peak is not determined accurately), use optimized starting values using the MATLAB Curve Fitting Toolbox as indicated in the code (Steps 17-20). To open the MATLAB Curve Fitting Toolbox write "curveFitter" in the command window. A complete user guide can be found at *https://au.mathworks.com/help/curvefit/*. Note that when entering the data to the Curve Fitting toolbox, the time corresponding to the *x*-axis should be in a logarithmic scale (log(corrBins)).
23. Run Step 19 if the fitting was performed using optimized starting values to get the additional fitted data. Be careful to indicate the number of fitted curves we want to obtain. For example, if we fitted three pCFs using optimized starting values, then change the number for "numDatasets" to 3.

```
"numDatasets = 3;"
```

24. Run Step 20 to plot the new fitting and get the new peak values. Again, be careful to indicate the number of fitted curves you want to obtain. For example, if we fitted three pCFs, then change the number for "numDatasets" to 3.

Cross-pCF workflow

The cross-pCF workflow ("*crossPCF_workflow_v2.m*") follows the same steps as the pCF workflow. The main difference is that you will need to open TIFF files for two channels.

1. Run Step 2.1. and Step 2.2. within the code to open TIFF files for the first and second channels. Then, follow the instructions given for the pCF workflow.

MSD workflow

NOTE: This code will need the "*cropped_images*" cell array as input data obtained using the *"PCF_workflow.m"*. Importantly, this code will calculate the MSD based on the following pCF distances: pCF0, pCF4, pCF8, pCF12, pCF16, pCF20, pCF24, pCF28, and pCF32.

1. Open the code called "*Calculate_and_Fit_MSD_v2.m*" and define the folder with all MATLAB functions by running Step 1 within the code.
2. Run Step 2 to calculate the pCF at a varied distance and store the data in a structure array. As indicated for the *PCF_workflow*, change the input parameters according to your experiment. Set 'mfirstCol' equal to 1 and 'mlastCol ' to the last column present in the intensity carpet. *e.g.*, If the acquisition was performed using a scan size of 64 pixels, then, set 'mlastCol' to 64. Change the 'sampleFreq' accordingly. Leave 'ReverseOrder = false' so the calculation is performed in the direction of the acquisition.

3. Run Step 3 to get the diffusion coefficient from the ACF.
4. Run Steps 4 fit the pCF to a Gaussian and get the peak values for each distance. Note that you will need to create cell arrays for each pCF distance that was calculated in Step 3. You will also need to modify the output “pCF_values” accordingly. Follow the example below.

   To fit the pCFs obtained for a distance of 4 pixels, the code in Step 4 should look like:

   ```
   pCF_distance = pCF_average_data.pCF4;

   pCF4_values = Fit_pCF_for_MSD_toGaussian2(pCF_distance, corrBins, MatrixSize);
   ```

   To fit the pCFs obtained for a distance of 8 pixels, then the code in Step 4 should look like:

   ```
   pCF_distance = pCF_average_data.pCF8;
   MatrixSize   = 18;

   pCF8_values = Fit_pCF_for_MSD_toGaussian2(pCF_distance, corrBins, MatrixSize);
   ```

   and so on.

   The values in “pCF_average_data.pCFx” (input) and “pCFx_values” (output) should be modified until we obtain the peaks values for “pCF4, pCF8, pCF12, pCF16, pCF20, pCF24, pCF28, and pCF32”. The next steps will not work until we obtain the matrices for each pCF distance as indicated.

5. Run Step 5 to create a table with the peak’s values.
6. Run Step 6 to calculate the MSD.
7. Export the MSD and time tables (“MSD” and “All_peaks_transformed” matrices) into an editable spreadsheet such as Excel. Remove those values with unreasonable time values (*i.e.*, MSD with peak values with long times (e.g., 0.5 or 1 s)). Take care of removing the MSD and corresponding time together. Also remove NaN values. Subsequently, assign each peak to its respective MSD to create a two-column matrix (time, MSD). Save the file as .csv.

8. Run Step 7 to import the manually curated MSD data (.csv). Remember that the code assumes time is stored in the first column and MSD is stored in the second column.

9. **Optional. Process the MSD and time without the export/import step.** Here, by using the “processMatrix” function, we apply a strong cutoff to eliminate unreasonable time values (e.g., removing values where time is greater than 0.23 s).

   Optional. Visualize the data before fitting to eliminate outliers.

10. Run Step 8 to fit the MSD data. After running this step, a new window will appear. Select the model function that will be used to fit the MSD: (*i*) free diffusion (MSD_isotropic), (*ii*) confined diffusion (MSD_ confined), or (*iii*) transient confined diffusion (MSD_ transt_confined).

11. Run Step 9 to plot the raw and fitted MSD data.

MSD workflow for cross-PCF data

The cross-pCF workflow for MSD ("*Calculate_fit_cross_MSD_v2.m*") follows the same steps as the MSD workflow. The main difference is that you will need to have the cell arrays called "*cropped_images_CH1*" and "*cropped_images_CH2*" as input generated using the "*crossPCF_workflow*" code. Then, the instructions given for the MSD workflow should be followed.