MCSD: A MATLAB Tool for Monte-Carlo Simulations of Diffusion in Biological Tissues

David N. Sousa
ISPA - Instituto Universitário

Hugo A. Ferreira
Universidade de Lisboa

Abstract

Understanding diffusion processes is particularly important for clinical imaging to distinguish between healthy and pathological tissues or between benign and malignant lesions. Computational models of diffusion allow us to predict specific architectural properties of biological tissues which correlate to the measured values of diffusion imaging, thus providing a deeper understanding of certain pathologies *in vivo*. Here we present a free and open-source MATLAB tool developed for that purpose.

Keywords: matlab, diffusion, monte-carlo, simulation.

1. Introduction

Diffusion is the random motion of particles in a fluid, statistically modeled by a probability density function (PDF) of the proportion of particles displaced in a given direction. Free diffusion occurs in homogeneous and isotropic media. Mathematically, this translates to a gaussian behavior described by the Einstein's Brownian motion equation (Einstein 1956). The density of particles P at the point x at time t is given by

$$P(x,t) = \frac{N}{\sqrt{4\pi Dt}} \cdot \exp\Big\{ - \frac{x^2}{4Dt} \Big\} \tag{1}$$

where N is the total number of particles and D is the coefficient of diffusion, dependent on the type of molecule and characteristics of the medium. The variance of the PDF is given by $\sigma^2 = 2Dt$. In two and three dimensions, the variance of the PDF is equal to $4Dt$ and $6Dt$ respectively. Therefore, it is possible to measure diffusion from the variance of the PDF. In complex heterogeneous environments such as biological tissues, the movement of water molecules is highly constrained by barriers such as those related to cellular components: membranes, organelles and large proteins. Nonetheless, in very small regions (e.g. intracellular compartments and the extracellular space) and in short time intervals, diffusion has a gaussian behavior and can still be described by eq. 1, taking different values in different compartments. Considering multiple compartments, each with its own PDF, the total diffusion value is obtained from the variance of the total PDF, which given the complexity of the environment has a non-gaussian behaviour. In this case, tissue micro-structural complexity is characterized by observing the deviation from a normal distribution, or in other words by calculating the kurtosis of diffusion $K = M_4 M_2^{-2} - 3$, where M_4 and M_2 are the 4th and 2nd central moments of the PDF. (Jensen and Helpern 2010).

Diffusion-Weighted Imaging (DWI) is a Magnetic Resonance Imaging (MRI) technique which measures the diffusion of water molecules in biological tissues *in vivo*, thereby enabling the characterization of tissue micro-structural complexity. Diffusion Kurtosis Imaging (DKI) is an extension of DWI which quantifies the non-gaussianity of water diffusion (Jensen, Helpern, Ramani, Lu, and Kaczynski 2005). This technique has shown to be very useful in quantifying the complexity of tissue barriers (Fieremans, Jensen, and Helpern 2011; Henriques, Correia, Nunes, and Ferreira 2015), and has been applied to the study of various pathologies including stroke, Parkinson's Disease, and prostate and breast cancer (Hui, Fieremans, Jensen, Tabesh, Feng, Bonilha, Spampinato, Adams, and Helpern 2012; Kamagata, Tomiyama, Motoi, Kano, Abe, Ito, Shimoji, Suzuki, Hori, Nakanishi *et al.* 2013; Tamura, Shinmoto, Soga, Okamura, Sato, Okuaki, Pang, Kosuda, and Kaji 2014; Nogueira, Brandão, Matos, Nunes, Loureiro, Ramos, and Ferreira 2014). Another important method for characterizing microstructural changes due to injury, treatment, disease or even normal physiological changes such as aging, is the diffusion tensor imaging (DTI). This technique can be used to map the three-dimensional diffusion of water as a function of spatial location and it is specially important in cases of anisotropic diffusion. In white matter for example, diffusion is essentially parallel to axonal membranes but highly restricted in the directions perpendicular to axons (Alexander, Lee, Lazar, and Field 2007). In such cases, diffusion is better described by a multivariate normal distribution (Basser, Mattiello, and LeBihan 1994a,b) as follows,

$$P(\boldsymbol{r},t) = \frac{1}{\sqrt{(4\pi t)^3|\boldsymbol{D}|}} \cdot \exp\Big\{ - \frac{-\boldsymbol{r}^T\boldsymbol{D}^{-1}\boldsymbol{r}}{4t} \Big\} \tag{2}$$

where

$$\boldsymbol{D} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{3}$$

is the diffusion tensor, a 3×3 covariance matrix of diffusion displacements normalized by the diffusion time interval, describing the magnitude of diffusion, the degree of anisotropy, and the orientation of diffusion anisotropy. The diffusion tensor is symmetric about the diagonal. Diagonalization yields the eigenvalues (λ_1, λ_2, λ_3) and corresponding eigenvectors off the diffusion tensor. If $\lambda_1 \sim \lambda_2 \sim \lambda_3$ diffusion is considered to be isotropic. Otherwise, it is considered to be anisotropic. White matter tractography and estimates of white matter connectivity patterns in the brain are possibly by measuring diffusion anisotropy (Conturo, Lori, Cull, Akbudak, Snyder, Shimony, McKinstry, Burton, and Raichle 1999; Mori, Crain, Chacko, and Van Zijl 1999; Basser, Pajevic, Pierpaoli, Duda, and Aldroubi 2000). The most widely used measure of anisotropy is the fractional anisotropy (FA) measure proposed by Koay, Chang, Carew, Pierpaoli, and Basser (2006):

$$FA = \sqrt{\frac{(\lambda_1 - \hat{\lambda})^2 + (\lambda_2 - \hat{\lambda})^2 + (\lambda_3 - \hat{\lambda})^2}{2(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)}} \tag{4}$$

where $\hat{\lambda} = (\lambda_1 + \lambda_2 + \lambda_3)/3$. FA takes values between 0 and 1. When diffusion is isotropic FA ~ 0. When diffusion is anisotropic FA ~ 1. Because of its sensitivity to both normal and abnormal microstructural changes in fibrous tissues, DTI is a promising technique in the field of clinical imaging.

Monte-Carlo simulations can be used to investigate the sensitivity of models of diffusion, kurtosis of diffusion, and anisotropy and further offer a deeper understanding on the various parameters defining the complexity of biological tissues, such as the cell radii distribution and type, permeability of membranes, intracellular volume fraction and values of intracellular and extracellular diffusion. (Lee, Bennett, and Debbins 2013; Sousa and Ferreira 2015; Szafer, Zhong, and Gore 1995; Meier, Dreher, and Leibfritz 2003; Fieremans, Novikov, Jensen, and Helpern 2010). In fact, by correlating theoretical and computational models insights with knowledge acquired from imaging techniques we can better understand various pathologies.

Although some theoretical and computational models were developed in this direction, no simple and free open-source tools were designed and made available for researchers in this field to test their basic predictions. The Monte-Carlo Simulations of Diffusion (**MCSD**) package and the tutorial we present here (section 3) were created for that purpose and their aim is to be simple and useful for both beginners and advanced users of MATLAB.

2. The MCSD tool

MCSD is a simple and free open-source MATLAB (The MathWorks Inc 2018) tool designed to simulate diffusion processes in complex environments such as biological tissues. With this tool it is possible to create 1, 2 or 3-D cell environments and simulate the brownian motion of particles subject to movement constraints, such as permeable or impermeable cell membranes or other types of barriers the user may want to define when designing his own complex environment. Additionally, **MCSD** offers a small set of functions to calculate particles' displacements, diffusion and kurtosis and skewness values of the particles' displacement distribution, as well as compartmental measures in all cartesian directions.

2.1. Installation and details

The MATLAB toolbox **MCSD** is available at `https://github.com/davidnsousa/mcsd`. In MATLAB the Statistics and Machine Learning Toolbox is required for a complete functioning of the package. **MCSD** is fully compatible with Octave too. To use it download the package and add the folder `mcsd/mcsd` to the MATLAB/Octave search path via the `addpath()` MATLAB/Octave built-in function. **MCSD** provides the following functions:

1. `cells()` designs 1, 2 and 3-D cell environments according to a user defined cell radii distribution and a specified region to pack the cells.

2. `rwalkfree()` generates the random walk of one or multiple particles in a free environment without any barriers.

3. `rwalk()` generates the random walk of one or multiple particles in the presence of barriers.

4. `displacement()` calculates the displacement of the particles in every orthogonal direction.

5. `cmeasures()` calculates a user-defined measure of the particles' displacement distribution in all cartesian directions, including compartmental components.

6. `where()` indicates what particles are inside or outside compartments at a specific step of the random walk trajectory.

7. `fanisotropy()` calculates the fractional anisotropy of diffusion.

In the MATLAB/Octave command line type `help` followed by the name of each one of the functions above for further details about input and output parameters. Combining this small set of functions it is possible to simulate highly complex environments and diffusion processes.

A replication script (in MATLAB) for the present tutorial can also be found at the **MCSD** github repository.

3. Tutorial

3.1. The function `rwalkfree()`

The `rwalkfree()` function takes 3 input arguments by the following order: the particles' starting positions, the number of steps and step size. The following simulates a 1-dimensional random walk of one particle for 100 steps of size equal to 1, starting at the origin:

```
>> X = rwalkfree(0, 100, 1);
```

where `X` is the particle random walk trajectory. The return of `rwalkfree()` is an array of dimensions equal to 'number of steps' × 'number of walkers' × 'degrees of freedom'. That is, `X` contains the coordinates of every step of every particle. By manipulating the first parameter in `rwalkfree()` it is possible to change (1) the number of walkers, (2) their starting position and (3) the number of degrees of freedom. For example, the following repeats the simulation above for 3 walkers starting at positions -3, 1 and 2:

```
>> X = rwalkfree([-3 1 2], 100, 1);
```

For 1000 walkers starting at the origin with 2 degrees of freedom:

```
>> X = rwalkfree(zeros(2, 1000), 100, 1);
```

3.2. The function `displacement()`

In all cases, the following calculates the displacement of the particles in every orthogonal direction:

```
>> dx = displacement(X);
```

The return `dx` is an array with size equal to 'number of walker' × 'degrees of freedom'. To visualize the random walk trajectories of the particles and the particles' displacement distributions the user can use the `plot()` and `hist()` MATLAB/Octave built-in functions. Use the `help` function to learn more about this functions. The x_1 and x_2 coordinates of

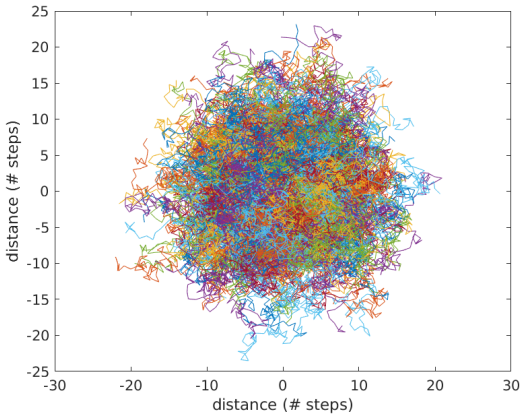

Figure 1: A 100-steps random walk of 1000 particles with 2 degrees of freedom, starting at the origin and with step size 1.

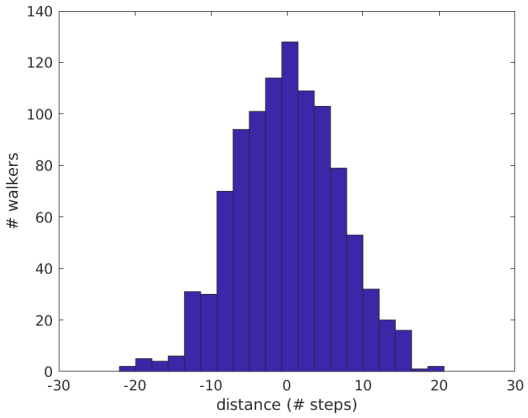

Figure 2: Displacement distribution of the horizontal component of displacement of 1000 particles with 2 degrees of freedom, starting at the origin and with step size 1.

the random walk trajectories can be accessed with `X(:,:,1)` and `X(:,:,2)` respectively. Displacement components are given by `dx(:,1)` and `dx(:,2)`. The walkers spread in every direction (Figure 1) and the histogram of the the particles' displacement distribution in the horizontal axis resembles the bell shape curve characteristic of Einstein's PDF (Figure 2), as it is expected in a process of unrestricted diffusion.

3.3. The `rwalk()` function

The `rwalk()` function is similar to the `rwalkfree()` function but with two additional optional parameters: an anonymous function defining any barriers present in the random walk environment and a value between 0 and 1 specifying the probability that walkers are allowed to cross barriers. If these parameters are ignored, then the function is identical to `rwalkfree()`. The following repeats the experiment above but confining the random walk to an ellipse representing an elongated cell:

```
>> cell = @(x, y) sqrt((x / 10) ^ 2 + y ^ 2) < 3;
>> X = rwalk(zeros(2, 1000), 100, 1, cell);
```

Figure 3 shows the particles' random walk trajectories, and Figure 4 shows the particles' displacement distributions. Because diffusion is confined to an elliptical cell, it is more restricted along the cell short-axis than it is along the cell long-axis. Other examples are a 3-dimensional random walk of 1000 particles confined to a spherical cell of radius 5 (Fig. 5):

```
>> cell = @(x, y, z) sqrt(x ^ 2 + y ^ 2 + z ^ 2) < 5;
>> X = rwalk(zeros(3, 1000), 100, 1, cell);
```

and a 3-dimensional random walk of 1000 particles confined to a permeable cell in shape of a tube of radius 3 (Figure 6). In both cases the `plot3()` MATLAB/Octave built-in function is used for 3-dimensional plots.

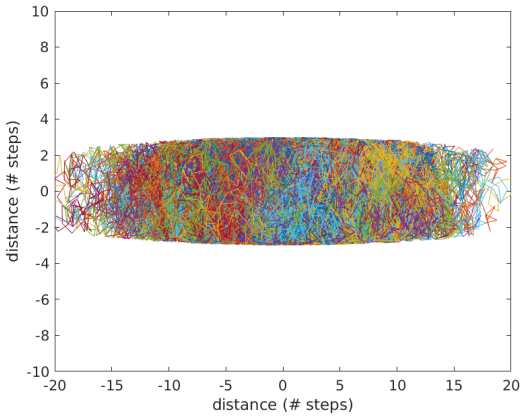

Figure 3: A 100-steps random walk of 1000 particles inside an elongated cell. Particles have 2 degrees of freedom, start at the origin and have step size 1.

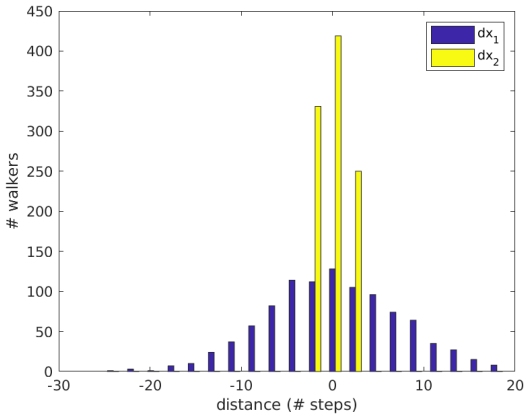

Figure 4: Displacement distributions of 1000 particles inside an elongated cell. Particles have 2 degrees of freedom, start at the origin and have step size 1. dx_1 and dx_2 are the horizontal (cell long-axis) and the vertical (cell short-axis) components of the displacement, respectively.

```
>> cell = @(x, y, z) sqrt(x ^ 2 + y ^ 2) < 3;
>> X = rwalk(zeros(3, 1000), 100, 1, cell, 0.0005);
```

By combining anonymous functions it is possible to design complex environments of multiple barriers and different types of cells shaped in different forms. In any case, the combined elements should always express logical conditions, as in the case of the example above, where the function evaluates to 1 or 0, whether (x_1, x_2), the position of the walker, falls inside or outside the ellipse. This is the idea behind the function `cells()` (Section 3.4): the combination of multiple conditions expressing circular regions where the function evaluates to a number different than 0, like in the following example:

```
>> c1 = @(x, y) sqrt(x ^ 2 + y ^ 2) < 3;
>> c2 = @(x, y) sqrt((x - 7) ^ 2 + (y - 9) ^ 2) < 2;
>> C = @(x, y) c1(x, y) .* 1 + c2(x, y) .* 2;
```

where '`.* 1`' and '`.* 2`' in the the last line make it possible to distinguish between different compartments. This might be necessary when designing a random walk environment to prevent the walkers from jumping between compartments when the step size is bigger than the distance between them.

3.4. The function `cells()`

The function `cells()` designs 1, 2 and 3-D cell environments, where cells are randomly positioned in space. It is based on the idea expressed in the end of the section above (Section 3.3), but in this case for non-overlaping circles. The input parameters are a vector of the cell radii distribution and a vector `[xmin xmax ymin ymax]` specifying the region to pack the

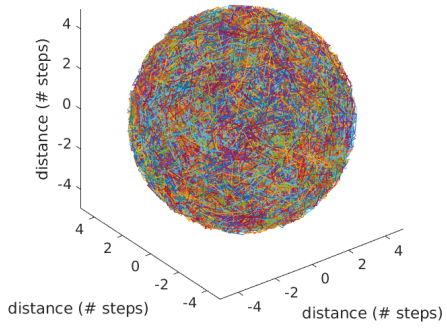

Figure 5: A 100-steps random walk of 1000 particles inside a spherical cell. Particles have 3 degrees of freedom, start at the origin and have step size 1.

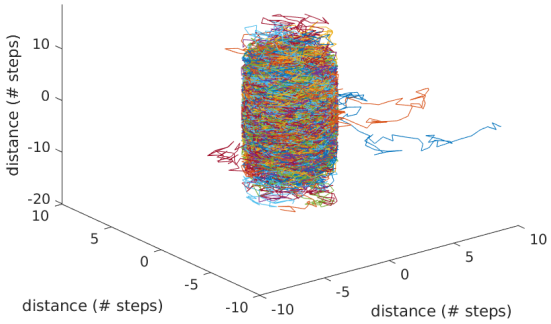

Figure 6: A 100-steps random walk of 1000 particles inside a permeable cell in shape of a tube. Particles have 3 degrees of freedom, start at the origin and have step size 1.

cells, or more specifically to select the intervals in which random numbers will be generated to choose the position of the circles inside the specified region. Consider for example three cells with radii 4, 3 and 2, all packed in a square region of side length 10:

```
>> [C, vf] = cells([4 3 2], [-5 5 -5 5]);
```

The output `C` is a combination of anonymous functions, each specifying one cell, where the combined function, given a position (x_1, x_2), evaluates to a particular integer. The second return of `cells()` is the intracellular volume fraction resulting from packing the cells. Note however that cells might not fit entirely inside the region. Therefore, `vf` is only an aproximation:

```
>> vf


vf =

    0.9111
```

The following simulates a 100-steps random walk of 1000 particles in the cell environment defined by `C`, but in this case the walkers start at random positions in the same region (the MATLAB/Octave built-in function `randi()` is used for that purpose):

```
>> X = rwalk(randi([-5 5], 2, 1000), 100, 1, C);
```

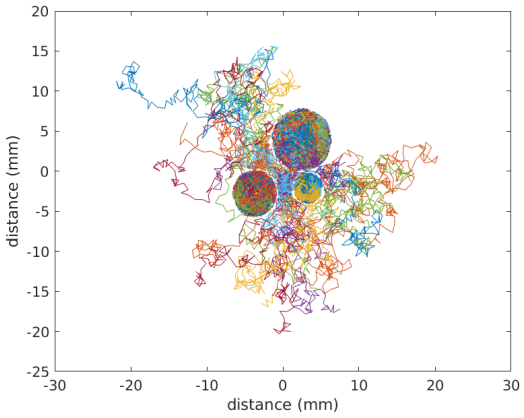

Figure 7: A 100-steps random walk of 100 of the 1000 particles in a 2-dimensional cell environment. Particles have a step size 1 and the starting position is random.

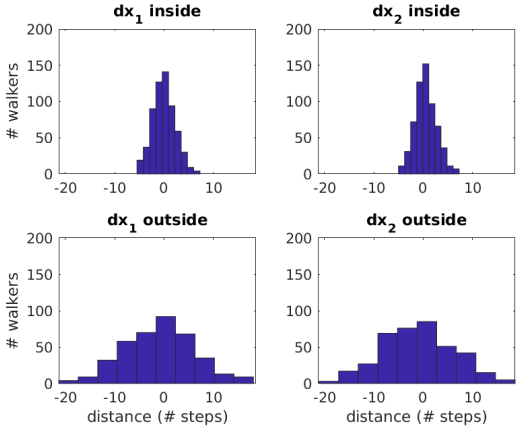

Figure 8: Inside and outside (of cells) displacement distributions of 1000 particles starting in random positions in a cell environment, with 2 degrees of freedom and step size 1. dx_1 and dx_2 are the horizontal and vertical components of displacement, respectively.

To investigate diffusion differences between different compartments, or any other measure of the particles' displacement distributions, it is necessary to discriminate between inside and outside particles.

3.5. The function `where()`

The function `where()` allows the user to identify which particles are inside and outside compartments at any step of the random walk trajectory. The input arguments are `X` and `C` computed above, and an integer value specifying the step. In this case, because cells do not have permeable membranes, every walker will always remain in its original compartment and therefore it is indifferent what step is considered as an argument for `where()`:

```
>> [in, out] = where(X, C, 1);
```

The first return of `where()` is a vector of the indexes of the particles inside compartments, while the second is a vector of the indexes of particles outside compartments. The vectors `in` and `out` can be used to access trajectory coordinates or displacement components of inside and outside particles (e.g. `X(:,in,1)`, `X(:,out,2)`, `dx(in,:)`, `dx(out,2)`). This function can be very useful for plots and measures. Figure 7 shows the random walk trajectories of 100 particles of the previous example, and Figure 8 shows the particles' displacement distributions in intracellular and extracellular spaces.

3.6. The function `cmeasures()`

`cmeasures()` calculates a user-defined measure of the particles' displacement distribution in all cartesian directions, including compartmental components. The function takes takes a

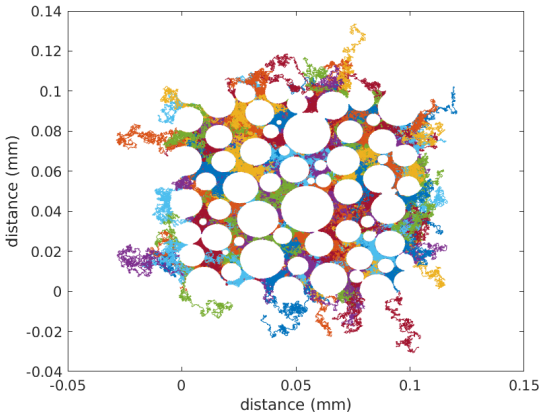

Figure 9: Simulation of a diffusion process in a cell environment characterized by a normal distribution of the cell radii, with $\mu = 0.005$ and $\sigma = 0.0025$. The coefficient of diffusion is set to 0.003 mm^2/s and the diffusion time interval is 0.026 ms

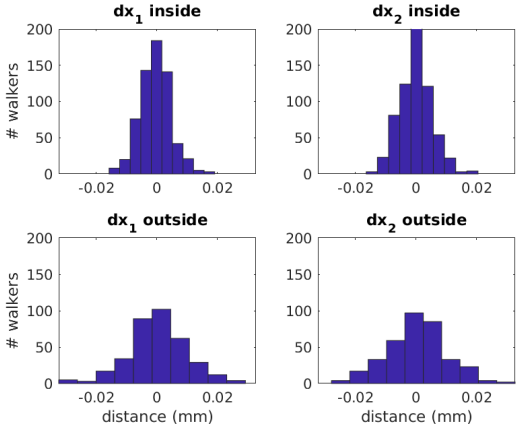

Figure 10: Inside and outside (of cells) displacement distributions of 1000 particles starting in random positions at the cell environment shown in Figure 9. dx_1 and dx_2 are the horizontal and vertical components of displacement, respectively.

function handle for a specific measure and `X` and `C`, as computed above. The last is optional, for those cases where barriers are present in the random walk environment.

Here we present a more realistic and useful example. Consider a normal distribution of the cell raddi with mean $\mu = 0.005$ mm and standard deviation $\sigma = 0.0025$ mm. We will use the MATLAB/Octave built-in function `normrnd()` to generate the radii distribution. For details about this function type '`help normrnd`' in the command line. The cells are packed in a square region of side length l = 0.1 mm. At 37° temperature the coefficient of free diffusion in water is D = 0.003 mm/s^2 (for simplicity we consider the same diffusion value in every compartment). In this case, instead of specifying the step size and number of steps explicitly, we derive them from the empirical value of diffusion. Consider also the time interval t = 0.026 s, typical of an MRI scan, and the random walk time-step dt = 0.000025 s. The random walk is t/dt steps long, and the step size $\sqrt{4Ddt}$ is calculated from the mean square displacement formula derived from the Einstein's PDF in two dimensions. The following code simulates the random walk of 1000 particles in the conditions described, and calculates every component of diffusion in intracellular and extracellular compartments:

```
>> D = 0.003;
>> t = 0.026;
>> dt = 0.000025;
>> l = 0.1;
>> [C, vf] = cells(normrnd(0.005, 0.0025, 1, 70), [0 l 0 l]);
>> vf
>> X = rwalk(rand(2, 1000) * l, t / dt, sqrt(4 * D * dt), C);
>> diffusion = @(dx) var(dx) / (2 * t);
>> Diff = cmeasures(diffusion, X, C)
```

```
vf =

    0.7181


Diff =

    0.0010    0.0010
    0.0005    0.0005
    0.0018    0.0018
```

where each column of the output arrays shows the measures in each displacement direction. The first line is the total value of the measure, the second is the measure inside compartments and the third is the measure outside compartments. The results show very clearly that diffusion, althought restricted, is isotropic due to the uniform distribution of the cells. To calculate kurtosis or skewness of the particles' displacement distributions, the user should substitute the function handle for diffusion in the first argument of `cmeasures()` with `@kurtosis` or `@skewness`. Depending on the processing power of the computer running this program, this simulation might take some minutes. To discriminate between intracellular and extracellular space more easily, Figure 9 shows only the extracellular walkers' trajectories. The particles' displacement distributions in intracellular and extracellular spaces are shown in Figure 10.

3.7. The function `fanisotropy()`

Given a diffusion tensor, `fanisotropy()` calculates the fractional anisotropy of diffusion (eq. 4). To demonstrate this, repeat the previous example but allow the walkers to also move freely upwards to simulate diffusion along axonal membranes (the axons are represented by `A` below). In this case the step size $\sqrt{6Ddt}$ is calculated from the mean square displacement formula derived from the Einstein's PDF in three dimensions. The diffusion tensor is calculated from the covariance matrix of displacements normalized by the diffusion time (eq. 3).

```
>> A = @(x, y, z) C(x, y);
>> X = rwalk(rand(3, 1000) * l, t / dt, sqrt(6 * D * dt), A);
>> Diff = cmeasures(diffusion, X, A)
>> dx = displacement(X);
>> DT = cov(dx) / (2 * t)
>> FA = fanisotropy(DT)


Diff =

    0.0010    0.0010    0.0032
    0.0005    0.0005    0.0032
    0.0019    0.0019    0.0033
```

```
DT =

    0.0010   -0.0001    0.0001
   -0.0001    0.0010    0.0000
    0.0001    0.0000    0.0032


FA =

    0.6336
```

As expected, because the same cell configuration was used to design the axons, radial diffusion is isotropic, but axial diffusion is unrestricted. Its value inside and outside axons is approximately the value of the coefficient of free diffusion in water at 37° temperature. As such, diffusion is anisotropic, as demonstrated by the value of FA calculated from the diffusion tensor DT.

4. Conclusions

Combining theoretical and computational insights about diffusion processes with the knowledge acquired from imaging techniques has proved to be an important research direction for understanding the micro-structural complexity of biological tissues. But, as yet, no simple and free open-source tools are available for researchers in this field to test their basic predictions. **MCSD** offers such possibility. As it was shown in the present tutorial, the functions provided are highly flexible and useful for the design of complex random walk environments such as biological tissues. In fact, although **MCSD** was developed specifically to simulate diffusion processes in such environments, researchers from other fields might find this package useful as well.

Acknowledgments

The authors acknowledge the support of Fundação para a Ciência e Tecnologia (UID/PSI/0481 0/2013 and UID/BIO/00645/2013)

References

Alexander AL, Lee JE, Lazar M, Field AS (2007). "Diffusion Tensor Imaging of the Brain." *Neurotherapeutics*, **4**(3), 316–329.

Basser PJ, Mattiello J, LeBihan D (1994a). "Estimation of the Effective Self-Diffusion Tensor from the NMR Spin Echo." *Journal of Magnetic Resonance, Series B*, **103**(3), 247–254.

Basser PJ, Mattiello J, LeBihan D (1994b). "MR Diffusion Tensor Spectroscopy and Imaging." *Biophysical journal*, **66**(1), 259–267.

Basser PJ, Pajevic S, Pierpaoli C, Duda J, Aldroubi A (2000). "In Vivo Fiber Tractography Using DT-MRI Data." *Magnetic resonance in medicine*, **44**(4), 625–632.

Conturo TE, Lori NF, Cull TS, Akbudak E, Snyder AZ, Shimony JS, McKinstry RC, Burton H, Raichle ME (1999). "Tracking Neuronal Fiber Pathways in the Living Human Brain." *Proceedings of the National Academy of Sciences*, **96**(18), 10422–10427.

Einstein A (1956). *Investigations on the Theory of the Brownian Movement*. Courier Corporation.

Fieremans E, Jensen JH, Helpern JA (2011). "White Matter Characterization with Diffusional Kurtosis Imaging." *Neuroimage*, **58**(1), 177–188.

Fieremans E, Novikov DS, Jensen JH, Helpern JA (2010). "Monte Carlo Study of a Two-Compartment Exchange Model of Diffusion." *NMR in Biomedicine*, **23**(7), 711–724.

Henriques RN, Correia MM, Nunes RG, Ferreira HA (2015). "Exploring the 3D Geometry of the Diffusion Kurtosis Tensor-Impact on the Development of Robust Tractography Procedures and Novel Biomarkers." *Neuroimage*, **111**, 85–99.

Hui ES, Fieremans E, Jensen JH, Tabesh A, Feng W, Bonilha L, Spampinato MV, Adams R, Helpern JA (2012). "Stroke Assessment with Diffusional Kurtosis Imaging." *Stroke*, **43**(11), 2968–2973.

Jensen JH, Helpern JA (2010). "MRI Quantification of Non-Gaussian Water Diffusion by Kurtosis Analysis." *NMR in Biomedicine*, **23**(7), 698–710.

Jensen JH, Helpern JA, Ramani A, Lu H, Kaczynski K (2005). "Diffusional kurtosis Imaging: The Quantification of Non-Gaussian Water Diffusion by Means of Magnetic Resonance Imaging." *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, **53**(6), 1432–1440.

Kamagata K, Tomiyama H, Motoi Y, Kano M, Abe O, Ito K, Shimoji K, Suzuki M, Hori M, Nakanishi A, *et al.* (2013). "Diffusional Kurtosis Imaging of Cingulate Fibers in Parkinson Disease: Comparison with Conventional Diffusion Tensor Imaging." *Magnetic resonance imaging*, **31**(9), 1501–1506.

Koay CG, Chang LC, Carew JD, Pierpaoli C, Basser PJ (2006). "A Unifying Theoretical and Algorithmic Framework for Least Squares Methods of Estimation in Diffusion Tensor Imaging." *Journal of Magnetic Resonance*, **182**(1), 115–125.

Lee CY, Bennett KM, Debbins JP (2013). "Sensitivities of Statistical Distribution Model and Diffusion Kurtosis Model in Varying Microstructural Environments: a Monte Carlo Study." *Journal of Magnetic Resonance*, **230**, 19–26.

Meier C, Dreher W, Leibfritz D (2003). "Diffusion in Compartmental Systems. I. A Comparison of an Analytical Model with Simulations." *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, **50**(3), 500–509.

Mori S, Crain BJ, Chacko VP, Van Zijl PC (1999). "Three-Dimensional Fracking of Axonal Projections in the Brain by Magnetic Resonance Imaging." *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, **45**(2), 265–269.

Nogueira L, Brandão S, Matos E, Nunes RG, Loureiro J, Ramos I, Ferreira HA (2014). "Application of the Diffusion Kurtosis Model for the Study of Breast Lesions." *European radiology*, **24**(6), 1197–1203.

Sousa DN, Ferreira HA (2015). "Diffusion Kurtosis Imaging: Monte Carlo Simulation of Diffusion Processes Using Crowdprocess." In *Bioengineering (ENBENG), 2015 IEEE 4th Portuguese Meeting on*, pp. 1–4. IEEE.

Szafer A, Zhong J, Gore JC (1995). "Theoretical Model for Water Diffusion in Tissues." *Magnetic resonance in medicine*, **33**(5), 697–712.

Tamura C, Shinmoto H, Soga S, Okamura T, Sato H, Okuaki T, Pang Y, Kosuda S, Kaji T (2014). "Diffusion kurtosis Imaging Study of Prostate Cancer: Preliminary Findings." *Journal of magnetic resonance imaging*, **40**(3), 723–729.

The MathWorks Inc (2018). *MATLAB - The Language of Technical Computing, Version 9.4 (R2018a)*. The MathWorks, Inc., Natick, Massachusetts.

Affiliation:

David N. Sousa
William James Center
ISPA- Instituto Universitário
1149-041 Lisboa, Portugal
E-mail: davidnsousa@gmail.com

Hugo A. Ferreira
Instituto de Biofísica e Engenharia Biomédica
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749-016 Lisboa, Portugal
E-mail: hugoferreira@campus.ul.pt