# Efficient numerical solution of the Bloch-Torrey equation for modeling multiple compartment diffusion

J. Li[1], D. Calhoun[2], C-H. Yeh[3], C. Poupon[4], and D. Le Bihan[4]
[1]INRIA-Saclay, Palaiseau Cedex, France, [2]CEA, Saclay, France, [3]National Yang-Ming University, Taiwan, [4]CEA Neurospin, Saclay, France

**Introduction**

Water diffusion in biological tissues is not free (Gaussian), as the signal attenuation is not monoexponential with diffusion-weighting (b value) [1]. Some groups have successfully characterized this attenuation with a biexponential model, which suggests the presence of 2 water pools in slow or intermediate exchange [1]. However, this model is still controversial [2] and the nature of the 2 pools (e.g., membrane-bound and intra/extracellular bulk water) remains elusive [3]. We propose a numerical method for solving the Bloch-Torrey partial differential equation in multiple diffusion compartments to compute the bulk magnetization of a sample under the influence of a diffusion gradient. We couple a mass-conserving finite element discretization in space with a stable time discretization using an explicit Runge-Kutta-Chebyshev method [4] . We are able to solve the Bloch-Torrey PDE in multiple compartments rapidly and accurately, making it a reasonable candidate as the forward solver in the inner iterative loop of an inverse problem solver going from signals to biological parameters.

**Method**

We consider the Bloch Torrey equation for the bulk magnetization in a sample under a diffusion gradient. In the sample Ω, we assume multiple compartments each with a diffusion coefficient $D^j$, hence we formulate the Bloch Torrey equation in each compartment $\Omega^j$, where the $\Omega^j$'s are the intra-cellular, extra-cellular, and membrane compartments.

$$\frac{\partial M^j(\vec{x},t)}{\partial t} = -i\gamma f(t)(\vec{G}\bullet\vec{x})M^j(\vec{x},t) + \nabla\cdot(D^j\nabla M^j(\vec{x},t)), \vec{x}\in\Omega^j, \qquad \Omega = \bigcup\Omega^j,$$

$$M^j(\vec{x},0) = M_0^j(\vec{x}), \vec{x}\in\Omega^j. \qquad M(\vec{x},t) = M^j(\vec{x},t), \vec{x}\in\Omega^j, \qquad j = \{'IC','MEM','EC'\}.$$

The initial condition $M_0^j(\vec{x})$ is the equilibrium density. We impose two interface conditions on the interfaces between the compartments:

$$D^j\frac{\partial M^j(\vec{x},t)}{\partial n} = D^k\frac{\partial M^k(\vec{x},t)}{\partial n}, \vec{x}\in\Gamma^{jk}, D^j\frac{\partial M^j(\vec{x},t)}{\partial n^j} = \alpha^{jk}\left(M^j(\vec{x},t) - M^k(\vec{x},t)\right), \vec{x}\in\Gamma^{jk}$$

The first enforces the conservation of mass and the second includes the permeability coefficient $\alpha^{jk}$ on the interface $\Gamma^{jk}$ across the compartments j and k. The signal is given by the integral of $M(\vec{x},t)$ over the entire sample: $S = \int_{\vec{x}\in\Omega} M(\vec{x},t)d\vec{x}$ . In the spatial discretization, we conserved numerical fluxes across the interfaces, hence there is no numerical mass loss associated with this method. In the time discretization, we used the Runge-Kutta Chebyshev method, which is an explicit time stepping method, meaning no large matrices need to be inverted at each time step, and the allowed time step dt is much larger than most commonly used methods (Euler or Crank-Nicolson time stepping, Monte-Carlo methods). In addition, the time step dt is adaptive: the code uses more time steps when problem needs it, namely, during the time when the gradient is turned on, i.e., when f(t) is not 0. This is all done automatically, based on the absolute and relative error tolerances on the computed residual, as requested by the user.

**Results and discussion**

We show the results of simulation for an infinite array (pseudo-periodic boundary conditions imposed on the computational domain) of spherical cells placed L=6 μm apart in x, y and z directions. There are two compartments: the intra-cellular compartment consists of spheres of radius 2.5 μm and $D^{IC} = 1.2\text{x}10^{-3}$ μm²/μs, the extra-cellular compartment has $D^{EC} = 1.5\text{x}10^{-3}$ μm²/μs. Assuming it is very thin the membrane-bound water compartment was modelled as an infinitely thin interface layer between between $\Omega^{IC}$ and $\Omega^{EC}$ with a finite permeability coefficient. The permeability between $\Omega^{IC}$ and $\Omega^{EC}$ was set to $\alpha = 5\text{x}10^{-5}$ μm²/μs. We simulated up to TE=Δ =30 ms and the orientation of $\vec{q}$ is in the direction [1,0,0]. In Fig 1a and 1b the sequence f(t) is a flat top lasting δ=10 ms (see Fig 1b for profile of f(t)). We simulated up to b=6000 μs /μm². In Fig 1a we plot the simulated signal at 4 different b values. We checked numerically that the mass at Δ =30 ms is indeed the same as at the start. The spatial discretization is 50 by 50 by 50. The requested time step given to the RKC algorithm is 500 μs and the requested absolute and relative errors is $10^{-4}$. At the beginning of the simulation many substeps were taken by RKC due to the sharp shape of the gradient, but during most of the simulation, there are only 2 to 4 substeps taken within each 500 μs requested step. In Fig 1b, we show the timing of the method during the 60 time steps (60x 500 μs = 30ms) by plotting the number of Laplacian evaluations against time step. At the beginning, 100 evaluations were needed to achieve relative and absolute error of $10^{-4}$, soon, only 40 evaluations were needed per time step. The computational time needed to compute each data point for a single b-value is about 85 seconds on a DELL laptop (Intel Core 2 Duo T7100 (1.80GHz)). For b=0 (pure diffusion case), computational time is the shortest, 58 seconds, but as b value increases, there is no significant increase in computational time (all around 80 seconds). In Figure 1c and 1d we show the simulated signal from three different oscillating gradients, with increasing frequency (sequence profiles are in Fig 1d). We see the signals are linear with $\vec{q}^{\,2}$, where $\vec{q} = \gamma\vec{G}$ , at high frequencies, hence probing short diffusion times (the top two curves), and it becomes nonlinear when the frequency is low (the lowest curve), hence indicating long diffusion times. The three curves are generated on a [20,20,20] spatial grid in 30 seconds of computational time each, on the same Dell laptop described above. In Fig 1e. we show the computational time required for freq = 2. Each bump of the sequence profile causes more Laplacian evaluations to be needed and this adaptivity is done automatically by the code. For G = 0, low number of Laplacian evaluations occurred at the tops of the bumps of f(t), where f(t) is the most smooth. Because our code is able to solve the Bloch Torrey equation in 3 dimensions in geometries with multiple compartments in very reasonable time, it would be very useful as the forward solver in the inner iterative loop of any inverse solver taking DMRI signals back to biological parameters.

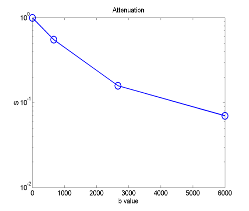

Fig 1a. Signal of flat top sequence profiled in Fig 1b.

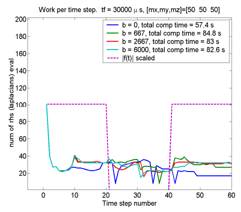

Fig 1b. Gradient profile and timing of flat top sequence.

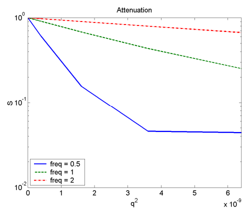

Fig 1c. Signal of oscillating sequence profiled in Fig 1d.

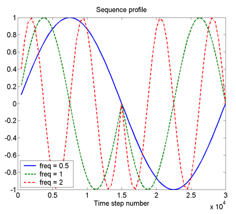

Fig 1d. Profile and timings of oscillating sequence.

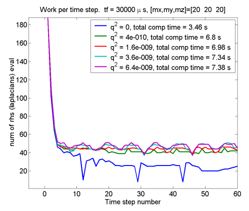

Fig 1e. Computational time for oscillating gradient freq = 2.

**References.** [1] Niendorf Th et al. MRM (1996) 36:847-857; Clark C, Le Bihan D. MRM (2000) 44:852-859; [2] Kiselev V. MRM (2007) 57: 464-469; [3] Le Bihan. PMB (2007) 52:57-90; [4] Sommeijer B. P. et al. JCAM (1998), 88(2):315–326;