Multiscale modeling of diffusion processes in dendrites and dendritic spines

Fredrik Eksaa Pettersen[1]

[1]Faculty of Mathematics and Natural Sciences
University of Oslo

June 26th, 2014

Outline

- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

Outline

- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

Physical models on different length scales

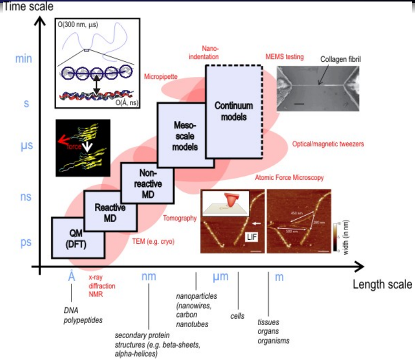


Figure: Illustration of physical models on different length scales, from Markus J. Buehler, MIT

Physical models on different length scales

- Systems on different length scales are characterized by different effects.

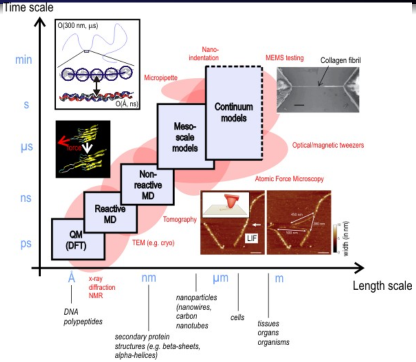


Figure: Illustration of physical models on different length scales, from Markus J. Buehler, MIT

Physical models on different length scales

- Systems on different length scales are characterized by different effects.
- Quantum Mechanics is the “typical” example in physics, but there are many more.

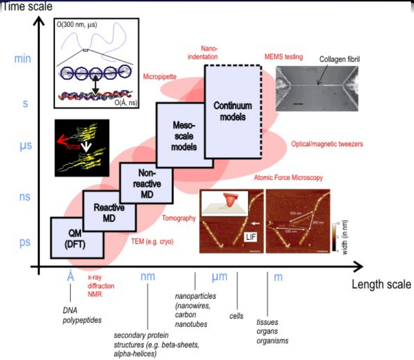


Figure: Illustration of physical models on different length scales, from Markus J. Buehler, MIT

Physical models on different length scales

- Systems on different length scales are characterized by different effects.
- Quantum Mechanics is the “typical” example in physics, but there are many more.
- Problems might arise between these length scales

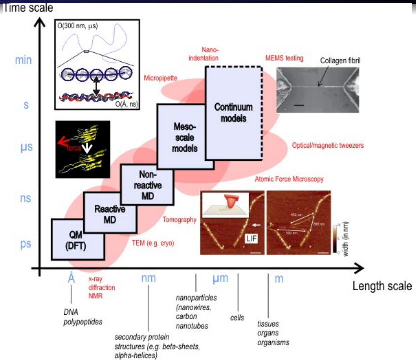


Figure: Illustration of physical models on different length scales, from Markus J. Buehler, MIT

Physical models on different length scales

Existing meso scale models

Some meso scale models exist already, but mostly these are aimed at specific problems and/or closed source.

- Dissipative Particle Dynamics
- Dendritc solidification modeling by
- Hybrid fluid flow models by

Aim of this project

Aim of this project

- Develop and implement a hybrid diffusion solver using Random Walk as a lower scale model.

Aim of this project

- Develop and implement a hybrid diffusion solver using Random Walk as a lower scale model.
- Make sure all parts of the theory are transparent.

Aim of this project

- Develop and implement a hybrid diffusion solver using Random Walk as a lower scale model.
- Make sure all parts of the theory are transparent.
- Test and verify implementation thoroughly.

Aim of this project

- Develop and implement a hybrid diffusion solver using Random Walk as a lower scale model.
- Make sure all parts of the theory are transparent.
- Test and verify implementation thoroughly.
- Apply developed software to physical problem in order to verify functionality.

Outline

- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

Random Walks

Figures/randomwalk-eps-conve

Random Walks

- Widely used in many applications.

Figures/randomwalk-eps-conve

Random Walks

- Widely used in many applications.
- Can be shown to fulfill the diffusion equation.

Figures/randomwalk-eps-conve

Random Walks

- Widely used in many applications.
- Can be shown to fulfill the diffusion equation.
- Used as lower scale model for this purpose

Figures/randomwalk-eps-conve

Finite difference methods

- For partial differential equations

Figures/FDM_stencils-

Finite difference methods
- For partial differential equations

- Approximate derivatives by finite differences using the definition of the derivative and omitting the limit:

$$\frac{du}{dt} = \lim_{\Delta t \to 0} \frac{u(t) - u(t+\Delta t)}{\Delta t}$$
$$\approx \frac{u(t) - u(t+\Delta t)}{\Delta t}$$

Figures/FDM_stencils-

Finite difference methods

- For partial differential equations

- Approximate derivatives by finite differences using the definition of the derivative and omitting the limit:

$$\frac{du}{dt} = \lim_{\Delta t \to 0} \frac{u(t) - u(t + \Delta t)}{\Delta t} \approx \frac{u(t) - u(t + \Delta t)}{\Delta t}$$

- Repeat for all derivatives in the PDE.

Figures/FDM_stencils-

Finite difference methods
- For partial differential equations

- Approximate derivatives by finite differences using the definition of the derivative and omitting the limit:

$$\frac{du}{dt} = \lim_{\Delta t \to 0} \frac{u(t) - u(t + \Delta t)}{\Delta t}$$
$$\approx \frac{u(t) - u(t + \Delta t)}{\Delta t}$$

- Repeat for all derivatives in the PDE.
- Solve equation on discrete mesh points.

Figures/FDM_stencils-

Finite difference methods

Finite difference methods

- The two discretizations used are summarized in the theta-rule description

$$\frac{u^{k+1} - u^k}{\Delta t} = \theta D \frac{\partial^2 u^{k+1}}{\partial x^2} + (1 - \theta) D \frac{\partial^2 u^k}{\partial x^2}. \qquad (1)$$

Finite difference methods

- The two discretizations used are summarized in the theta-rule description

$$\frac{u^{k+1}-u^k}{\Delta t} = \theta D \frac{\partial^2 u^{k+1}}{\partial x^2} + (1-\theta) D \frac{\partial^2 u^k}{\partial x^2}. \qquad (1)$$

- In order to accommodate a larger time step, the Backward Euler discretization ($\theta = 1$) must be implemented:

$$u_i^{k+1} = \frac{D\Delta t}{\Delta x^2}\left((u_{i+1}^{k+1} - u_i^{k+1}) - (u_i^{k+1} - u_{i-1}^{k+1})\right) + u_i^k.$$

Finite difference methods

- The two discretizations used are summarized in the theta-rule description

$$\frac{u^{k+1} - u^k}{\Delta t} = \theta D \frac{\partial^2 u^{k+1}}{\partial x^2} + (1 - \theta) D \frac{\partial^2 u^k}{\partial x^2}. \qquad (1)$$

- In order to accommodate a larger time step, the Backward Euler discretization ($\theta = 1$) must be implemented:

$$u_i^{k+1} = \frac{D\Delta t}{\Delta x^2} \left((u_{i+1}^{k+1} - u_i^{k+1}) - (u_i^{k+1} - u_{i-1}^{k+1}) \right) + u_i^k.$$

- By insertion the BE scheme results in a tridiagonal linear system in 1D.

Tridiagonal linear systems

Tridiagonal linear systems are efficiently solved by a specialized Gaussian elimination algorithm.

```
g[0] = up[0]/b[0];
H[0] = c[0]/b[0];
for(int i=1; i<n; i++){
    //forward substitution
    H[i]  = -c[i]/(b[i] + a[i]*H[i-1]);
    g[i] = (up[i] - a[i]*g[i-1])/(b[i] + a[i]*H[i
        -1]);
}
u[n-1] = g[n-1];
for(int i=(n-2);i>=0;i--){
    //Backward substitution
    u[i] = g[i] - H[i]*u[i+1];
}
```

Code 1 : The tridiag algoritm

Finite difference methods

Backward Euler in 2D

Finite difference methods

Backward Euler in 2D

- In 2D the solution to the discrete diffusion equation is a matrix.

Finite difference methods

Backward Euler in 2D

- In 2D the solution to the discrete diffusion equation is a matrix.
- Rewriting the matrix as a vector results in a banded linear system:

$$\left(\begin{array}{ccc|ccc|ccc}
\gamma & -2\beta & 0 & -2\alpha & 0 & 0 & 0 & 0 & 0 \\
-\beta & \gamma & -\beta & 0 & -2\alpha & 0 & 0 & 0 & 0 \\
0 & -2\beta & \gamma & 0 & 0 & -2\alpha & 0 & 0 & 0 \\
\hline
-\alpha & 0 & 0 & \gamma & -2\beta & 0 & -\alpha & 0 & 0 \\
0 & -\alpha & 0 & -\beta & \gamma & -\beta & 0 & -\alpha & 0 \\
0 & 0 & -\alpha & 0 & -2\beta & \gamma & 0 & 0 & -\alpha \\
\hline
0 & 0 & 0 & -2\alpha & 0 & 0 & \gamma & -2\beta & 0 \\
0 & 0 & 0 & 0 & -2\alpha & 0 & -\beta & \gamma & -\beta \\
0 & 0 & 0 & 0 & 0 & -2\alpha & 0 & -2\beta & \gamma
\end{array}\right)\mathbf{u} = \mathbf{u}_p$$

Tridiagonal linear systems

Block tridiagonal solver

The tridiag algorithm can be rewritten to solve block tridiagonal systems by replacing divisions with matrix inverses:

$$
\begin{aligned}
H_0 &= -B_0^{-1} C_0 \\
\mathbf{g}_0 &= B_0^{-1} \mathbf{u}_{p0} \\
H_i &= -\left(B_i + A_i H_{i-1}\right)^{-1} C_i \\
\mathbf{g}_i &= \left(B_i + A_i H_{i-1}\right)^{-1} \left(\mathbf{u}_{pi} - A_i \mathbf{g}_{i-1}\right)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{u}_{n-1} &= \mathbf{g}_{n-1} \\
\mathbf{u}_i &= \mathbf{g}_i + H_i \mathbf{u}_{i+1}
\end{aligned}
$$

Tridiagonal linear systems

Performance of Block tridiagonal solver

- The 1D tridiagonal solver requires $\mathcal{O}(n)$ operations, comparable to the Forward Euler scheme.

Tridiagonal linear systems

Performance of Block tridiagonal solver

- The 1D tridiagonal solver requires $\mathcal{O}(n)$ operations, comparable to the Forward Euler scheme.
- The Block tridiagonal solver requires inversion of $2n$ matrices, but only once.

Tridiagonal linear systems

Performance of Block tridiagonal solver

- The 1D tridiagonal solver requires $\mathcal{O}(n)$ operations, comparable to the Forward Euler scheme.
- The Block tridiagonal solver requires inversion of $2n$ matrices, but only once.
- In total $\mathcal{O}(n^{2d-1})$ operations are required for a general system, one order less than e.g. LU decomposition.

Tridiagonal linear systems

Performance of Block tridiagonal solver

- The 1D tridiagonal solver requires $\mathcal{O}(n)$ operations, comparable to the Forward Euler scheme.
- The Block tridiagonal solver requires inversion of $2n$ matrices, but only once.
- In total $\mathcal{O}(n^{2d-1})$ operations are required for a general system, one order less than e.g. LU decomposition.
- Memory impact can also be reduced to $8 \cdot n^{2d-1}$ bytes, as opposed to $8 \cdot n^{2d}$ bytes.

Outline

- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

The algorithm

After the initial setup, the algorithm is as follows:

The algorithm

After the initial setup, the algorithm is as follows:

- The result from previous PDE time step, $\mathbf{u}_p$, is converted to a distribution of random walkers and sent to the RW solver.

The algorithm

After the initial setup, the algorithm is as follows:

- The result from previous PDE time step, $\mathbf{u}_p$, is converted to a distribution of random walkers and sent to the RW solver.
- The RW solver does a predefined number of micro scale time steps which correspond to one PDE time step.

The algorithm

After the initial setup, the algorithm is as follows:

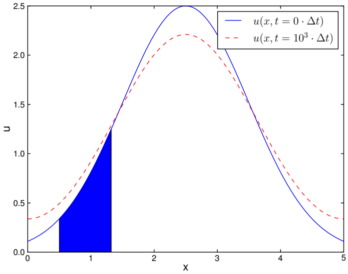
- The result from previous PDE time step, $\mathbf{u}_p$, is converted to a distribution of random walkers and sent to the RW solver.
- The RW solver does a predefined number of micro scale time steps which correspond to one PDE time step.
- The result from the RW solver is converted back to a concentration and this replaces the PDE solution, $\mathbf{u}_p$.

The algorithm

After the initial setup, the algorithm is as follows:

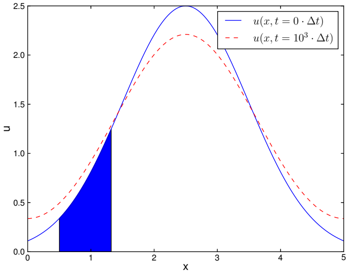
- The result from previous PDE time step, $\mathbf{u}_p$, is converted to a distribution of random walkers and sent to the RW solver.
- The RW solver does a predefined number of micro scale time steps which correspond to one PDE time step.
- The result from the RW solver is converted back to a concentration and this replaces the PDE solution, $\mathbf{u}_p$.
- $\mathbf{u}_p$ is then used as input to calculate the next time step.

Conversion between length scales



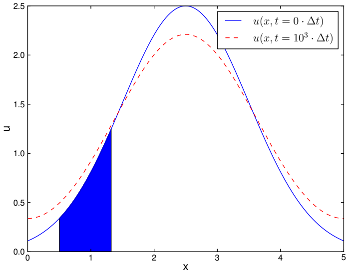
Conversion between length scales

- A single, real integer converts the concentration in one mesh point to a number of random walkers.



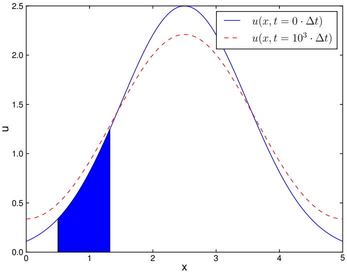
Conversion between length scales

- A single, real integer converts the concentration in one mesh point to a number of random walkers.
- $$C_{ij} = Hc \cdot u_{ij}$$



Conversion between length scales

- A single, real integer converts the concentration in one mesh point to a number of random walkers.
- $$C_{ij} = Hc \cdot u_{ij}$$
- The conversion must be done at each time step because the concentration over an area of the mesh might change.



Coupling the models through the step length

- The RW solver needs a constraint in order to make sure it models diffusion on the same time scale as the PDE model.

Coupling the models through the step length

- The RW solver needs a constraint in order to make sure it models diffusion on the same time scale as the PDE model.
- From an Einstein relation we find

$$\langle \tilde{\Delta x}^2 \rangle = 2Dd\tilde{\Delta t}$$

Coupling the models through the step length

- The RW solver needs a constraint in order to make sure it models diffusion on the same time scale as the PDE model.
- From an Einstein relation we find

$$\langle \tilde{\Delta x}^2 \rangle = 2Dd\tilde{\Delta t}$$

Coupling the models through the step length

- The RW solver needs a constraint in order to make sure it models diffusion on the same time scale as the PDE model.
- From an Einstein relation we find

$$\langle \tilde{\Delta x}^2 \rangle = 2Dd\tilde{\Delta t}$$

- Rewriting this results in the desired restriction which is placed on the step length:

$$l = \sqrt{2dD\frac{\Delta t}{\tau}}$$

Boundary conditions on the random walk

- Perfectly reflecting boundaries, equivalent to zero flux

$$\frac{\partial C}{\partial n} = 0$$

Boundary conditions on the random walk

- Perfectly reflecting boundaries, equivalent to zero flux

$$\frac{\partial C}{\partial n} = 0$$

- Updating concentration at each time step must also be considered a boundary condition.

Boundary conditions on the random walk

- Perfectly reflecting boundaries, equivalent to zero flux

$$\frac{\partial C}{\partial n} = 0$$

- Updating concentration at each time step must also be considered a boundary condition.
- Other boundary conditions might have been better, say perfect flux exchange:

$$\frac{\partial u}{\partial n} = \frac{\partial C}{\partial n}$$

Boundary conditions on the random walk

- Perfectly reflecting boundaries, equivalent to zero flux

$$\frac{\partial C}{\partial n} = 0$$

- Updating concentration at each time step must also be considered a boundary condition.
- Other boundary conditions might have been better, say perfect flux exchange:

$$\frac{\partial u}{\partial n} = \frac{\partial C}{\partial n}$$

- Requires some work on the PDE boundary conditions etc.

Outline

- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

Computation of the error

- Solving PDEs by FDMs results in errors which can tell a lot about the implementation.

Computation of the error

- Solving PDEs by FDMs results in errors which can tell a lot about the implementation.
- From the residuals, we know how the error should behave.

Computation of the error

- Solving PDEs by FDMs results in errors which can tell a lot about the implementation.
- From the residuals, we know how the error should behave.
- For a given exact solution, u_e, the error is defined by:

$$\varepsilon(t^k) = ||u(t^k) - u_e(t^k)||_2$$

$$\approx \sqrt{\Delta x \Delta y \sum_{i=0}^{k} \sum_{i=0}^{k} (u(t^k, x_i, y_j) - u_e(t^k, x_i, y_j))^2}.$$

Verification techniques

Method of manufactured solutions

- Make a solution by adapting the source term.

Verification techniques

Method of manufactured solutions

- Make a solution by adapting the source term.
- For example:

$$\begin{aligned} u(x,t) &= \frac{1}{x+1} \\ \implies s(x,t) &= \frac{2D}{(x+1)^3} \end{aligned}$$

Verification techniques

Method of manufactured solutions

- Make a solution by adapting the source term.
- For example:

$$u(x,t) = \frac{1}{x+1}$$

$$\implies s(x,t) = \frac{2D}{(x+1)^3}$$

- Many useful variations of this method.

Verification techniques

Method of manufactured solutions

- Make a solution by adapting the source term.
- For example:

$$u(x,t) = \frac{1}{x+1}$$
$$\implies s(x,t) = \frac{2D}{(x+1)^3}$$

- Many useful variations of this method.
- Tests are done using

$$u(x,y,t) = e^{-\pi^2 t}\cos(\pi x)\cos(\pi y) + 1,$$

which fulfills boundary conditions and has $s(x,t) = 0$.

Verification techniques

Convergence tests

- Error term is on the form

$$\varepsilon = C_x \Delta x^2 + C_t \Delta t^1,$$

for the schemes that are implemented.

Verification techniques

Convergence tests

- Error term is on the form

$$\varepsilon = C_x \Delta x^2 + C_t \Delta t^1,$$

for the schemes that are implemented.

- Measuring the exponents gives the convergence of the scheme.

Verification techniques

Convergence tests

- Error term is on the form

$$\varepsilon = C_x \Delta x^2 + C_t \Delta t^1,$$

for the schemes that are implemented.

- Measuring the exponents gives the convergence of the scheme.
- Do several simulations and calculate

$$r \simeq \frac{\log\left(\varepsilon_1/\varepsilon_2\right)}{\log\left(\Delta t_1/\Delta t_2\right)}.$$

Verification techniques

Convergence tests

- Error term is on the form

$$\varepsilon = C_x \Delta x^2 + C_t \Delta t^1,$$

for the schemes that are implemented.

- Measuring the exponents gives the convergence of the scheme.
- Do several simulations and calculate

$$r \simeq \frac{\log\left(\varepsilon_1/\varepsilon_2\right)}{\log\left(\Delta t_1/\Delta t_2\right)}.$$

- Often difficult tests.

Verification techniques

Numerical exact solutions

- Discretization is a reformulation of a PDE as a difference equation.

Verification techniques

Numerical exact solutions

- Discretization is a reformulation of a PDE as a difference equation.
- New exact solution can be found - theoretically zero error!

Verification techniques

Numerical exact solutions

- Discretization is a reformulation of a PDE as a difference equation.
- New exact solution can be found - theoretically zero error!
- Forward Euler solution (1D):

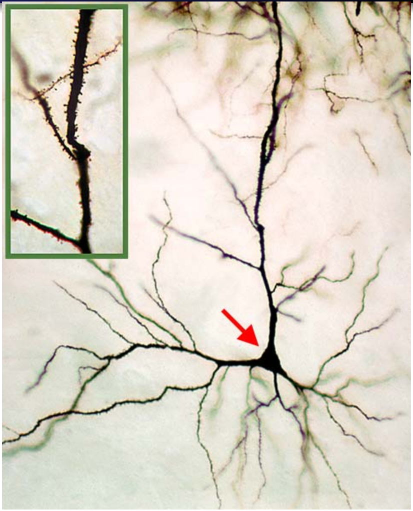
$$u^{k+1} = \sum_{i=0}^{k} \binom{k}{i} (D\Delta t)^i \frac{2^i}{\Delta x^{2i}} (\cos(\pi \Delta x) - 1)^i \cos(\pi x).$$

- Backward Euler solution:

$$\vec{u}^k = \left(\mathbf{M}^{-1}\right)^k \vec{u}^0.$$

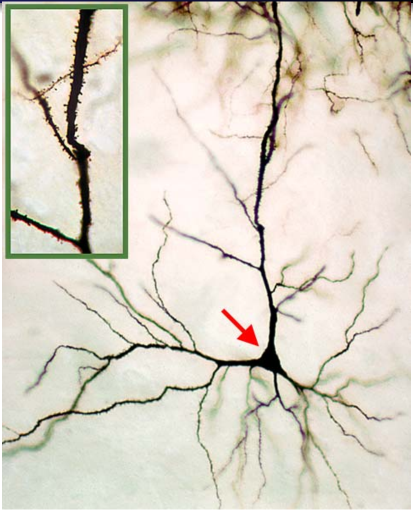
Outline

The problem



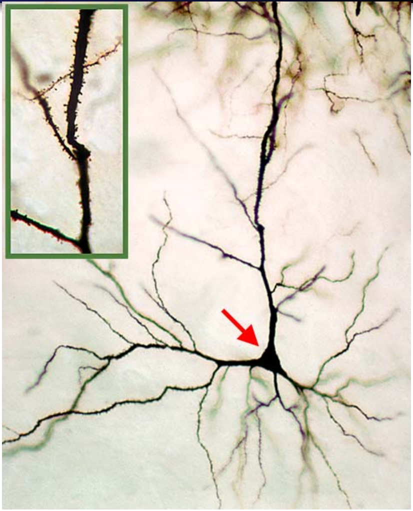
The problem

- Pyramidal neuron



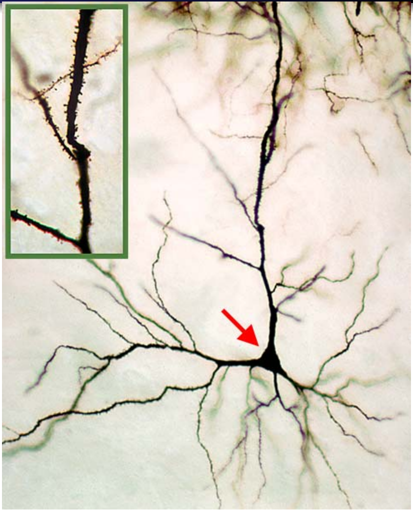
The problem

- Pyramidal neuron
- Protein Kinase Cγ



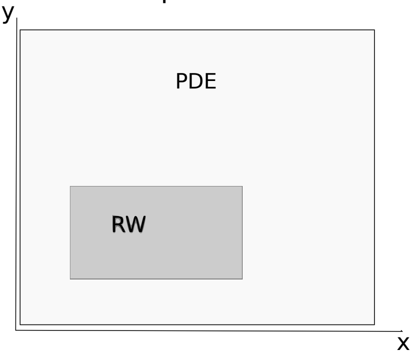
The problem

- Pyramidal neuron
- Protein Kinase Cγ
- Dendritic spines

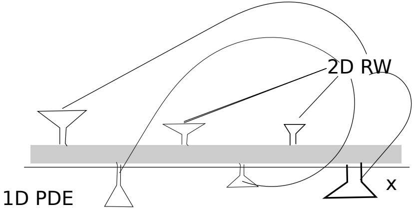


Computational model

Default setup



Modified setup



Parameters and other details

- Craske et.al. report increases of $5\frac{\text{nMol}}{\text{L}}$ in spines.

Parameters and other details

- Craske et.al. report increases of $5\frac{\text{nMol}}{\text{L}}$ in spines.
- Corresponds to 1-2 walkers, using values from Arellano et.al.

Parameters and other details

- Craske et.al. report increases of $5\frac{\text{nMol}}{\text{L}}$ in spines.
- Corresponds to 1-2 walkers, using values from Arellano et.al.
- Spine geometries chosen at random to correspond with Arellano et.al.

Parameters and other details

- Craske et.al. report increases of $5\frac{\mathrm{nMol}}{\mathrm{L}}$ in spines.
- Corresponds to 1-2 walkers, using values from Arellano et.al.
- Spine geometries chosen at random to correspond with Arellano et.al.
- Initial condition is modified to accommodate absorption effects (`http://jcb.rupress.org/content/170/7/1147/suppl/DC1`).

Outline

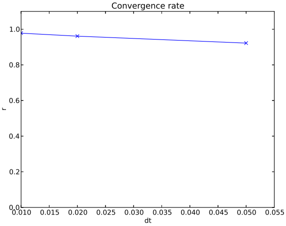
- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

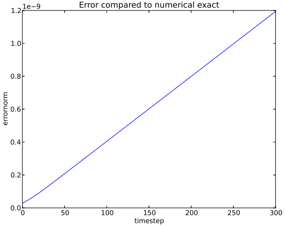
What we are looking for

- Successful tests of PDE solvers.
- Successful tests of RW solver.
- Successful test of hybrid solver given sufficient number of walkers.

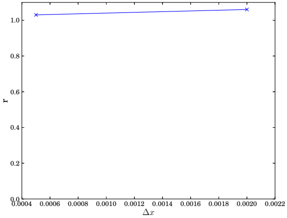
PDE solvers

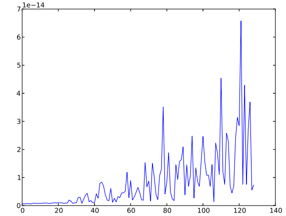
BE



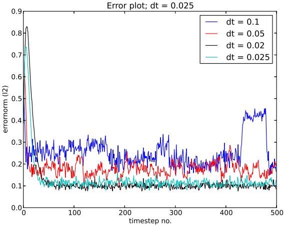


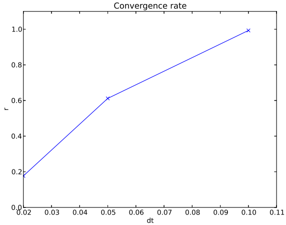
FE





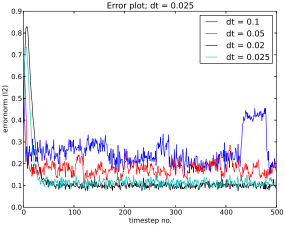
RW solver

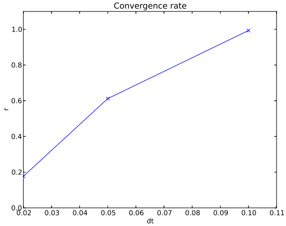

Error plot; dt = 0.025
dt = 0.1
dt = 0.05
dt = 0.02
dt = 0.025
errornorm (l2)
timestep no.
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
100
200
300
400
500


Convergence rate
r
dt
0.0
0.2
0.4
0.6
0.8
1.0
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
0.11

RW solver

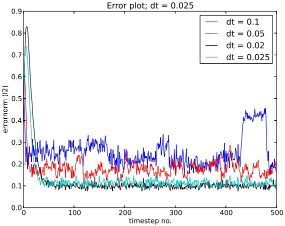
- Needs a new initial condition, otherwise similar.

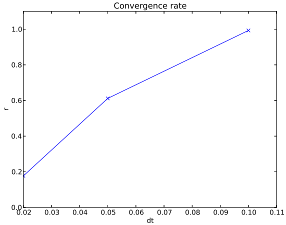




RW solver

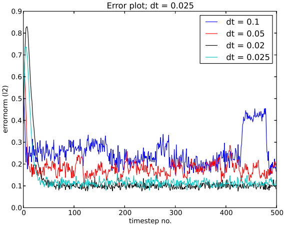
- Needs a new initial condition, otherwise similar.
- No numerical exact solution.

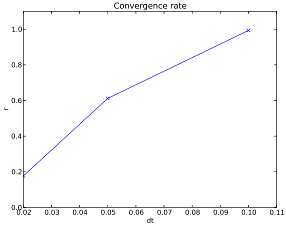




RW solver

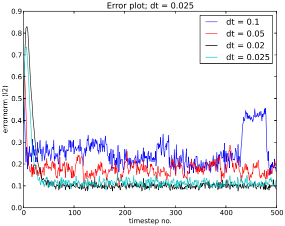
- Needs a new initial condition, otherwise similar.
- No numerical exact solution.
- Expected convergence rate of 0.5.

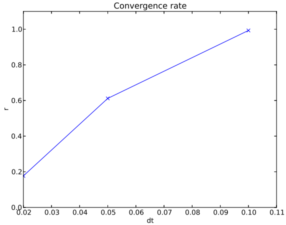




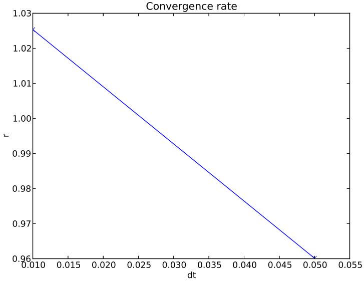
RW solver

- Needs a new initial condition, otherwise similar.
- No numerical exact solution.
- Expected convergence rate of 0.5.
- Verifies that the coupling works.





Hybrid solver



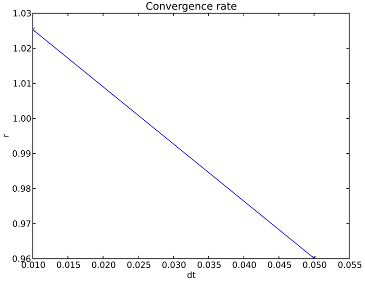
Hybrid solver

- New error term:

$$\varepsilon(t) = C_t \Delta t + C_x \Delta x^2 + \frac{C_{RW}}{\sqrt{Hc}}$$

- A lot of walkers are required:

$$Hc \geq \frac{1}{\Delta t^2}$$



Hybrid solver

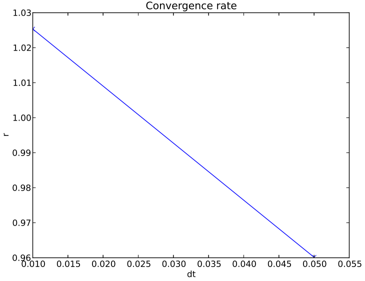
- New error term:

$$\varepsilon(t) = C_t \Delta t + C_x \Delta x^2 + \frac{C_{RW}}{\sqrt{Hc}}$$

- A lot of walkers are required:

$$Hc \geq \frac{1}{\Delta t^2}$$

- Expect first order convergence.



Hybrid solver

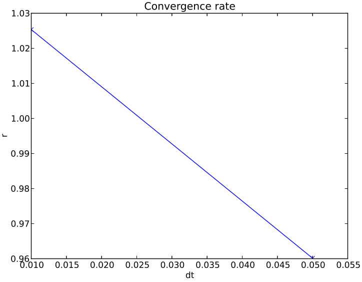
- New error term:

$$\varepsilon(t) = C_t \Delta t + C_x \Delta x^2 + \frac{C_{RW}}{\sqrt{Hc}}$$

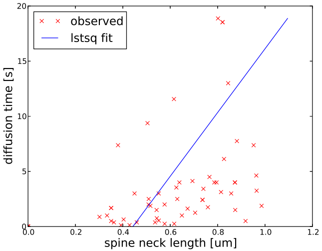
- A lot of walkers are required:

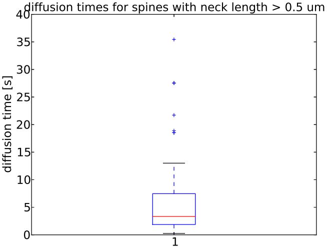
$$Hc \geq \frac{1}{\Delta t^2}$$

- Expect first order convergence.
- The extra error can be controlled (at a large cost).



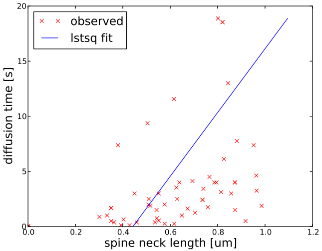
Results from application

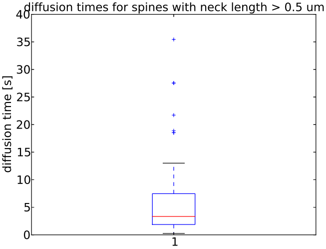




Results from application

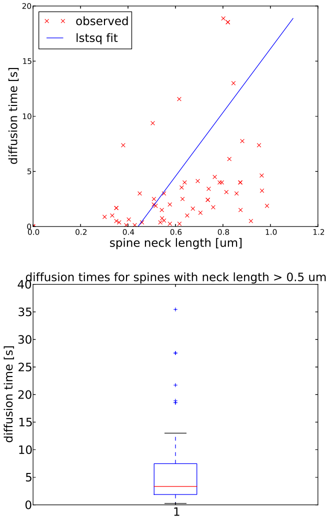
- Consistently towards lower range of results by Craske et.al.





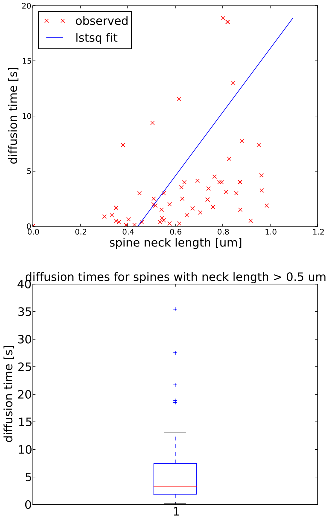
Results from application

- Consistently towards lower range of results by Craske et.al.
- Otherwise relatively good.



Results from application

- Consistently towards lower range of results by Craske et.al.
- Otherwise relatively good.
- $\sim$ 2-3 seconds can be added.



Outline

- Motivation
- Theory
- Details of the coupling
- Verification
- Application
- Results
- Concluding remarks

Conclusions

Conclusions

- Both PDE, RW and hybrid solvers are implemented correctly.

Conclusions

- Both PDE, RW and hybrid solvers are implemented correctly.
- Software can be modified to model real applications.

Conclusions

- Both PDE, RW and hybrid solvers are implemented correctly.
- Software can be modified to model real applications.
- Proof of concept.

Further work

Further work

- Implement flux exchange boundary conditions

Further work

- Implement flux exchange boundary conditions
- Find other physical applications - anisotropy.

Further work

- Implement flux exchange boundary conditions
- Find other physical applications - anisotropy.
- Implement Finite element PDE solver.

Thank you for your attention!

References I

Diane M. Mueller, ND RN and John J. Oro', MD.

Prospective Analysis of Presenting Symptoms Among 265 Patients With Radiographic Evidence of Chiari Malformation Type I With or Without Syringomyelia.

Journal of the American Academy of Nurse Practitioners, 16(3), 2004.

References II

Støverud, Karen H. and Alnæs, Martin and Langtangen, Hans Petter and Haughton, Victor and Mardal, Kent-André.

Effect of pia mater, central canal, and geometry on wave propagation and fluid movement in the cervical spinal cord.

Manuscript submitted for publication, 2014.