Discretization for Radial Concentrations is central difference forward time explicit Euler integration

radial discretization in a nutshell

The continuous partial differential equation we consider is

$$\frac{dc}{dt} = \frac{1}{r}\frac{\partial}{\partial r}\left(rD\frac{\partial}{\partial r}c(r,t)\right) - k_{\text{PDE}}c(r,t)pde(r) + F_0(t)r_0\delta(r-r_0).$$

Denoting $c_i(t) = c(r_i, t)$ and $pde_i = pde(r_i)$, we used the following discretization

$$\frac{c_i(t+\Delta t) - c_i(t)}{\Delta t} = \frac{D}{r_i \Delta r}(c_{i+1/2} - c_{i-1/2}) + \frac{D}{\Delta r^2}(c_{i+1} - 2c_i + c_{i-1}) - k_{\text{PDE}}pde_i c_i + F_0\delta_{0i}$$

Where we have taken $c_{-1} = c_{-1/2} = c_0$, and $c_{N+1} = c_{N+1/2} = c_N$, and $c_{i+1/2} = \frac{1}{2}(c_{i+1} + c_i)$, and $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

For each time step, we then explicitely computed $c(r_i, t + \Delta t)$ with a time step of $\Delta t = 0.005$ seconds and radial step $\Delta r = 1$ micron.

more detailed description of radial discretization

- Let $\oplus$ denote the concatenation function.
- Let $\mathbf{rmesh} = \bigoplus_{r=r_0=50\mu m}^{r_{\max}=1000\mu m} r \equiv \bigoplus_{i=0}^{N-1} r_i$ denote the 1D array of radial distances from the aggregate. Note that rmesh is an array of length N.
- Let $c(r,t)$ denote the cAMP concentration at radial distance r at time t in nM.
- Let $pde(r,t)$ denote the PDE concentration at radial distance r at time t in nM.
- Let $F_0(t) \in [0, 200]$ nM/s be the rate of change in cAMP concentration at the aggregate at time t.
- Let $dt = 0.005s$ be the size of one time step and $dr = 1\mu m$ be the spatial resolution of **rmesh**.

Then, we may compute $c(r, t + dt) = c(r,t) + dt \cdot \frac{dc}{dt}\Big|_t$, where $\frac{dc}{dt}\Big|_t$ is returned by the function `time_step`.

`time_step` is effectively defined from the following steps:

step 1: compute the face terms

- $\mathbf{dface}(t) = 0 \oplus \left(\bigoplus_{i=0}^{N-2} c(r_{i+1}, t) - c(r_i, t)\right) \oplus 0$. Note **dface** has length N+1
- $\mathbf{cp}(t) = c(r_0, t) \oplus \left(\bigoplus_{i=0}^{N-1} c(r_i, t)\right) \oplus c(r_{\max}, t)$. Note **cp** has length N+2
- $\mathbf{face}(t) = \frac{1}{2}\bigoplus_{i=0}^{N} \text{cp}(r_{i+1}, t) + \text{cp}(r_i, t)$. Note face has length N+1

Let $\text{cp}(r_i, t)$, $\text{face}(r_i, t)$, and $\text{dface}(r_i, t)$ denote the i^{th} entry of $\mathbf{cp}(t)$, $\mathbf{face}(t)$, and $\mathbf{dface}(t)$, respectively.

step 2: compute the transient terms

- $\textbf{term1}(t) = \frac{D}{dr} \bigoplus_{i=0}^{N-1} \frac{1}{r_i} \Big(\text{face}(r_{i+1}, t) - \text{face}(r_i, t) \Big)$
- $\textbf{term2}(t) = \frac{D}{dr^2} \bigoplus_{i=0}^{N-1} \text{dface}(r_{i+1}, t) - \text{dface}(r_i, t)$
- $\textbf{termDEG}(t) = -k_{PDE} \bigoplus_{i=0}^{N-1} c(r_i, t) pde(r_i, t)$
- $\textbf{termBC}(t) = F_0 \oplus \bigoplus_{i=1}^{N-1} 0$

step 3: add the transient terms Then,

$$\frac{dc}{dt}\Big| = \textbf{term1}(t) + \textbf{term2}(t) + \textbf{termDEG}(t) + \textbf{termBC}(t)$$

the numpy implementation fo the time step

In [1]:
```
import numpy as np
```
executed in 272ms, finished 12:14:31 2020-07-15

In [2]:
```
def time_step(c, pde, rmesh, D, kPDE, dr, fluxLeft, fluxRight=0):
    '''returns d[cAMP]/dt = the rate of change of the cAMP field c
    rmesh is the 1D radial mesh,
    c = 1D array of cAMP concentrations
    pde = 1D array of PDE concentrations
    D = diffusion coefficient of cAMP
    kPDE = decay constant of cAMP due to PDE
    dr = spatial resolution, for example 1 (microns)
    fluxLeft is the rate cAMP is being added at the "left" boundary, closest t
    fluxRight is he rate cAMP is being added at the "right" boundary, furthest
    explicitely using only current state.'''
    #step one: compute the face terms
    dface = np.hstack([ 0. , np.diff(c) , 0. ])
    cp    = np.hstack([c[0], c, c[-1]])
    face  = cp[1:]*0.5 + cp[:-1]*0.5

    #step two: compute transient terms
    term1 = D*np.diff(face)/rmesh/dr
    term2 = D*np.diff(dface)/dr**2
    termDEG = -1*kPDE*c*pde
    #calculate boundary term
    termBC = np.hstack([ fluxLeft , 0*c[1:-1] , fluxRight ])

    #step three: add transient terms
    dcdt  = term1 + term2 + termBC + termDEG
    return dcdt
```
executed in 9ms, finished 12:14:31 2020-07-15

In [ ]:
```

```