Unveiling Cellular & Molecular Events of Cardiac Arrhythmias

Hoang-Trong Minh Tuan[1], George S. William[1], Greg D. Smith[2], M. Saleet Jafri[1,3,4]

1 - Department of Bioinformatics and Computational Biology George Mason University

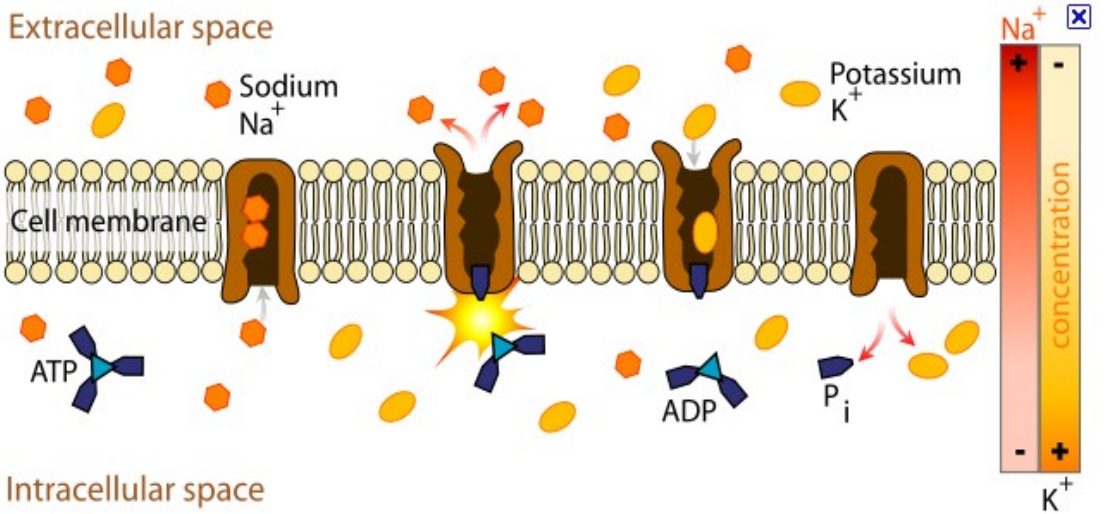
2- Department of Applied Science, William & Mary College

3 - Institute of Computational Medicine,The Johns Hopkins University

4 - Department of Biomedical Engineering and Technology The University of Maryland Baltimore

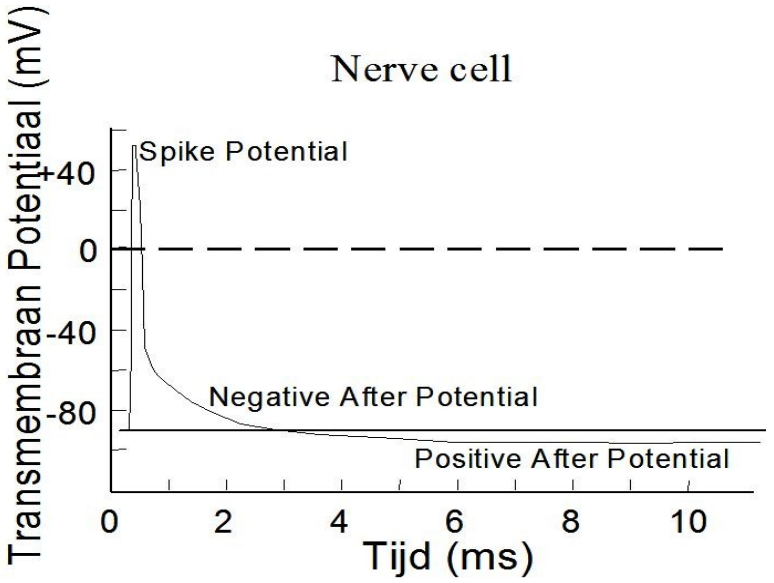
Overview “cellular electrophysiological modelling”

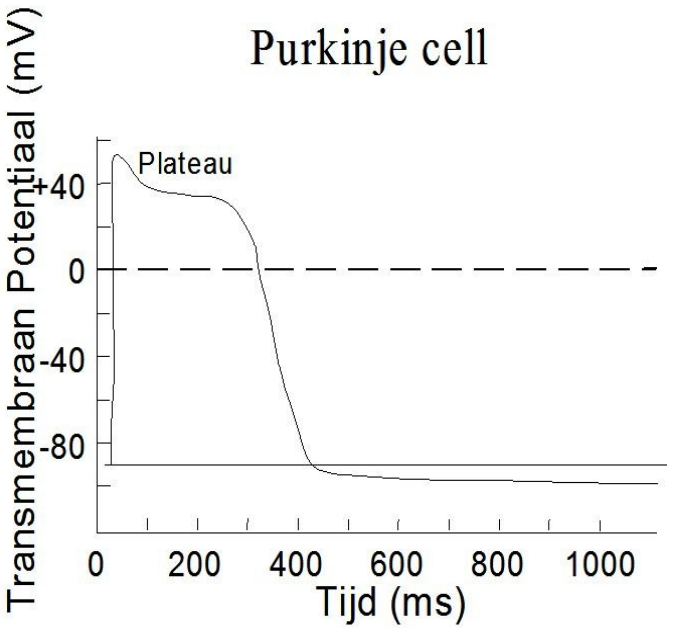
- Basic phenomenon in excitable cells is the propagation of **action potential** (AP)
- 1948-1952: Hodgkin-Huxley model (squid axon)
  - 3 ionic currents: Na+, K+, Cl-
- 1962: Noble model (Purkinje fibre)
  - 4 ionic currents: Na+, I_K1, I_K2



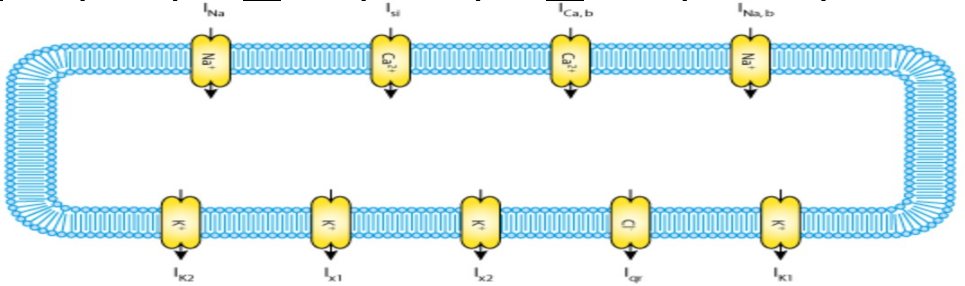
GEORGE MASON UNIVERSITY

Overview “cellular electrophysiological modelling”





- 1975: McAllister-Noble-Tsien model (Purkinje fibre)
  - 9 ionic currents: I_Na, I_K2, I_si (Ca), I_x1(fast), I_x2 (slow), I_qr, I_K1, I_Na.b, I_Cl.b



http://www.biomedicalphysics.org/PhysCircCourse/chapter_3.htm

Overview “cellular electrophysiological modelling”

[1975, 79, 83] Fabiato & Fabiato: confirm “Ca-induced Ca-release (CICR)

1977 — 1985 — 1991 — 1994

Beeler-Reuter model (ventricular myocyte)

DiFrancesco-Noble model (Purkinje fibre)

Luo-Rudy-1 model (ventricular myo.)

Luo-Rudy-2 model (vent. myo.)

[1972, Bassingthwaighte & Reuter]: ”influx of Ca from extracellular is not high enough to triger the contractile”

1993 Cheng et al. : [Ca] spark is the result of Ca release from SR due to the opening of one or more RyRs in a diadic subspace (local)

1998 — 1999 — 2001

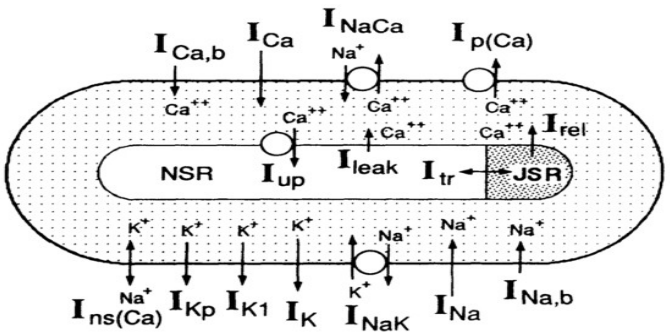
Jafri-Rice-Winslow model (vent. myo.)

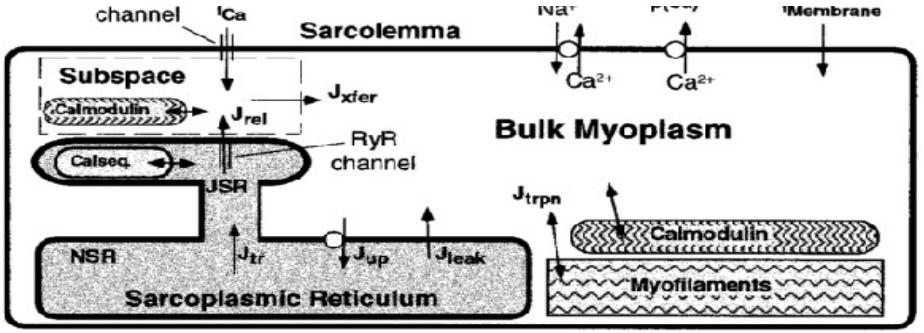
CICR mechanism

Rice-Jafri-Winslow model (vent. myo.)

graded-response

Sobie et al. model (calcium spark termination)





Review: Cheng-Lederer (2007)

Project: understanding Ca-dependent cardiac arrhythimias

- Cardiac arrhythmias:
  - Symptoms*: Patients with irregular pulse may lead to sudden death*
  - ***Reentry arrhythmias****: "from a single impulse, a single cardiac cell may give rises to two or more propagated response"*
  - Ancient times: diagnosis based on aertial pulse
  - Mid-20th century: ECG to record cardiac rate + rhythm → Knowledge on clincal symptoms
  - End-20th century: single channel kinetics, whole-cell recording → help understand arrhythmogenic mechanism
  - Early-21th century: structure and functional changes of specific ion channels → system biology approach
- Facts:
  - genetic defect of proteins link to disease phenotype
  - kinetic properties of ion channels related to protein structure (ion channels, buffers)
  - *defects in Ca dynamics is believed lead to cardiac arrhythmias*
  - *defects in Ca-regulating proteins have been linked to cardiac arrhythmias*
  - patients with these defects can live (for years/decades) and die suddenly

GEORGE MASON UNIVERSITY

Project: understanding Ca-dependent cardiac arrhythimias

- Challenge:
  - Rarity of the events (in weeks in physiological time) ~ months/years in simulation time
  - Detailed whole-cell model

    [Ca] spark --[triger/non-trigger]----> [Ca] waves ---> affect conduction system
  - To understand reentry arrythmias, we need to understand the basic electrochemical phenomena
    - [Ca] spark has small time-duration event → *underlying physiological events can be easily skipped by approximate methods*
    - It is necessary to look at networks of cells to study cardiac arrhythmias
  - [Ca] spark = generated by local cluster of RyR+DHPR → follow stochastic manner
  - Stochastic model (with Monte Carlo simulation) are very computationally expensive due to (1) very small time step, (2) large state space

Project: understanding Ca-dependent cardiac arrhythimias

- Motivation:
  - Adances in genetic engineering: point mutation (RyR2, CASQ2) → *kinetics model can be built (Markov-chain)*
  - Boom in computational power (multicore, many-core)
  - High-capacity and fast-access speed data storage
  - Availability of computational algorithm
    - Probability Density Method
    - Moment Closure Method
    - Ultra-fast Monte Carlo Simulation method – stochastic, exact method

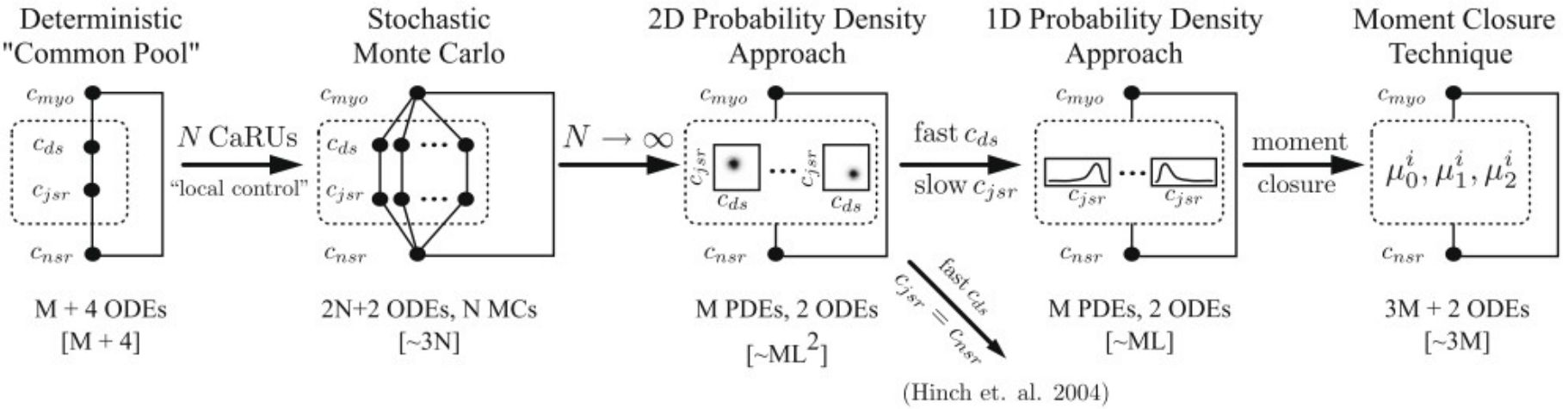
GEORGE MASON UNIVERSITY

Computational algorithm

- Probability-distribution model [Williams et al, 2007]
  - Deterministic formulasim
  - Diadic subspace [Ca] is NOT in quasi-equilibrium with myoplasm [Ca] or NSR [Ca]
  - A functional release unit (FSU): RyR + DHPR
  - $[Ca]_{ds}$ = f(FSU state)
  - $[Ca]_{jsr}$ = f(FSU state)
  - Valid when there are a large number of release units
  - 650x faster than original Monte Carlo

Computational algorithm

- Moment-Closure [Williams et al. 2008]
  - Approximated probability densities by a beta-distribution
  - Describe the probability density by its first two moments.
  - Close the distribution by estimating the third moment is a function of the first 2 moments
  - 1000x faster than original Monte Carlo



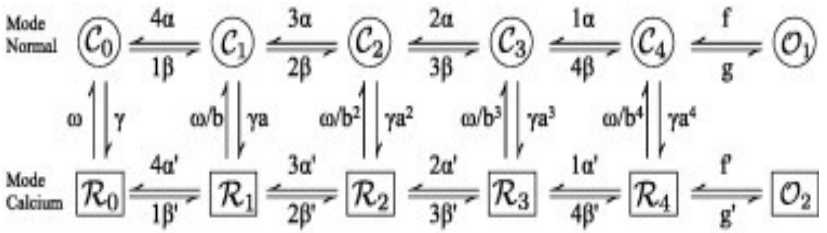
Review: Williams et al. (2010)

Ultra-fast Markov-Chain Monte-Carlo algorithm

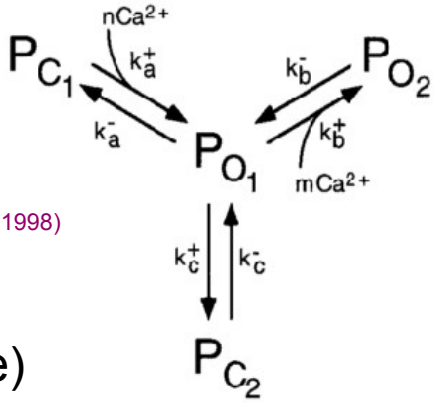
- [Unpublished]
- Property:
  - Stochastic
  - Exact method
  - Low memory usage
- How GPU fit to our problem & algorithm
  - Highly independent of release site computation
  - Low memory demands makes it fit to the limited device memory (4GB in Tesla 1060, 3GB in Fermi)

Ultra-fast Markov-Chain Monte-Carlo algorithm

- Kinetic of a single ion channel:
  - Model as a Markov-chain with a number of states



Jafri et. al (1998)

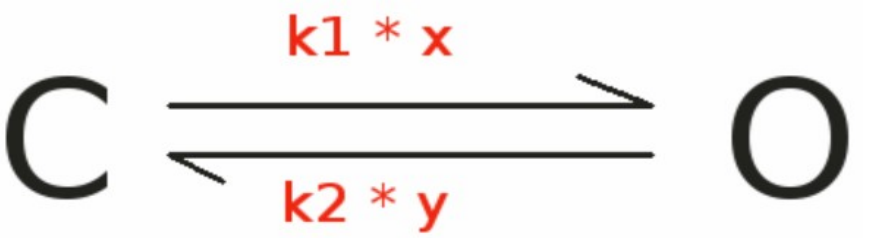


Smith-Keizer (1998)

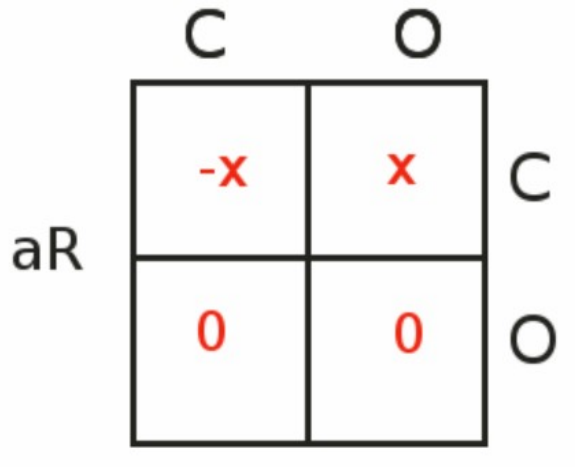
  - rate transition = constant (can multiply with a gating variable)
  - continuous-time discrete state → infinitesimal generating matrix a(i,j), b(i,j)
- A cluster of ion channels:
  - A rate transition matrix A(i,j), B(i,j)
  - How???

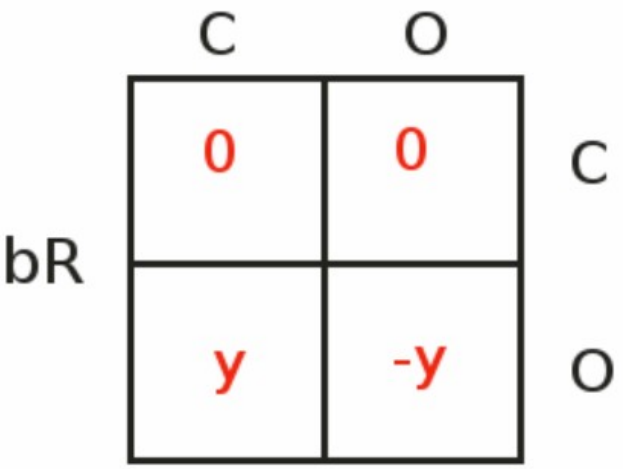
Ultra-fast Markov-Chain Monte-Carlo algorithm

- RyR
  - M=2 states minimal model



  - k1 = f(Ca, RyR_{open}) : Ca-dependent: C → O
  - k2 = f(RyR_{open}) : Ca-independent: O → C
  - Single-channel rate-transition matrix





GEORGE MASON UNIVERSITY

Ultra-fast Markov-Chain Monte-Carlo algorithm

- A cluster of RyR
  - State: (x1, x2) with
    - x1 = number of RyR in state 1
    - x2 = number of RyR in state 2
  - E.g: N=5 RyR

$$\frac{(N+M-1)!}{(N-1)! \times M!}$$

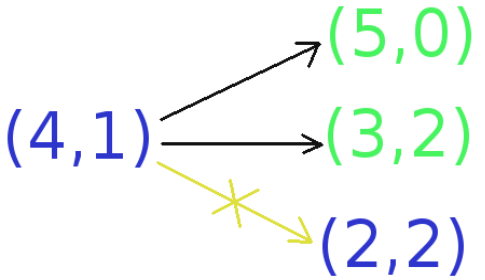
	C	O
1	5	0
2	4	1
3	3	2
4	2	3
5	1	4
6	0	5

  - Cluster rate-transition matrix
    - AR(:,:), BR(:,:) of size 6x6

GEORGE MASON UNIVERSITY

Ultra-fast Markov-Chain Monte-Carlo algorithm

- Exact simulation:
  - In a small time-step, only a SINGLE channel can change state



  - Law of conversation: rate out + rate in = υ

AR

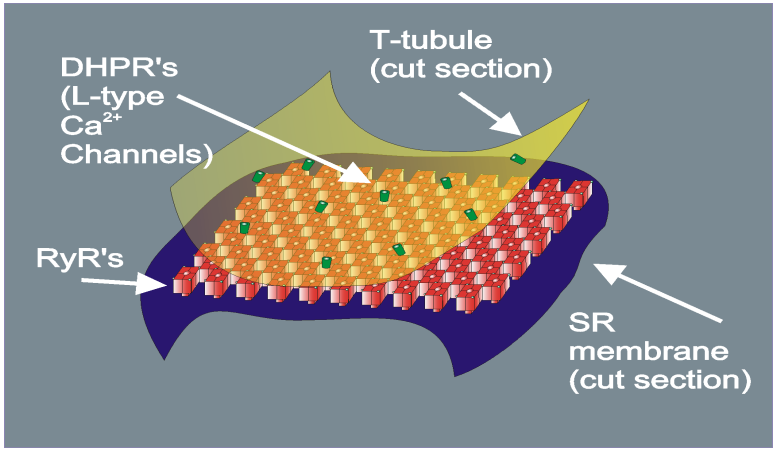
(5,0)	(4,1)	(3,2)			(0,5)
-q	q	0	0	0	0
r1	-(r1+r2)	r2			

$$q = 5 \times \mathrm{aR(C,O)}$$
$$r1 = 1 \times \mathrm{aR(O,C)}$$
$$r2 = 4 \times \mathrm{aR(C,O)}$$

GEORGE MASON UNIVERSITY

Ultra-fast Markov-Chain Monte-Carlo algorithm

- A Calcium release site:
  - Each with 50-300 stochastically gating Ca^{2+} channels
  - *[Bers & Stiffel, 1993]* RyR:DHPR=7.3:1
  - Species to species: RyR:LCC ~ 4:1 → 8:1
- Single cell:
  - Calcium release sites: 10,000-20,000



GEORGE MASON UNIVERSITY

Ultra-fast Markov-Chain Monte-Carlo algorithm

- Model a release site:
  - Kronecker sum of matrices from 2 clusters: RyR & DHPR

$$\mathbf{K} = \mathbf{A} \oplus \mathbf{B} = \mathbf{A}_n \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{B}_m$$

  - RyR:
    - 2 matrices: $[Ca]_{ds}$-dependent, $[Ca]_{ds}$-independent
  - DHPR:
    - 4 matrices: V1-dependent, V2-dependent, $[Ca]_{ds}$-dependent, (V, $[Ca]_{ds}$)-independent

GEORGE MASON UNIVERSITY

Ultra-fast MCMC

- Complexity
  - 50 2-state RyR: cluster of 51 states
  - 7 6-state DHPR: cluster of 924 states
  - Release site: 47,124 states
  - Memory demand: 16GB

- K matrix:
  - Highly sparse
- Question???
  - *Use an existing package for sparse matrix?* - still need a full matrix first
  - *How to handle computation with such sparse matrix in GPU?*

GEORGE MASON UNIVERSITY

Ultra-fast MCMC

- Compact form for K:
  - Use two separate matrices:
    - Kcomp(i,j) = keep ONLY non-zero rate transition
    - Kidx(i,j) = true column index of Kcomp(i,j)

0	12	0	1
0	0	0	2

- A

$$\mathbf{K}_{comp} = f(\mathbf{A}_{comp}, \mathbf{B}_{comp})$$
$$\mathbf{K}_{idx} = g(\mathbf{A}_{idx}, \mathbf{B}_{idx})$$

- Acomp

	12	1
	2	x

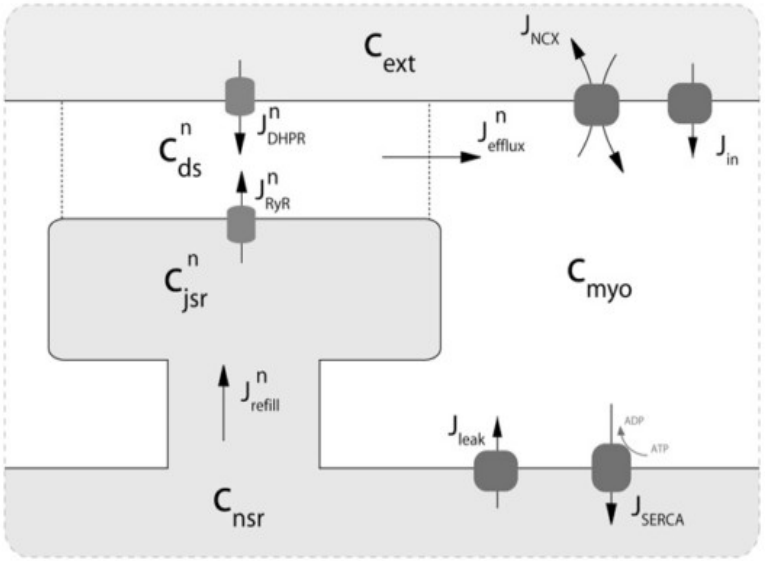
- Aidx

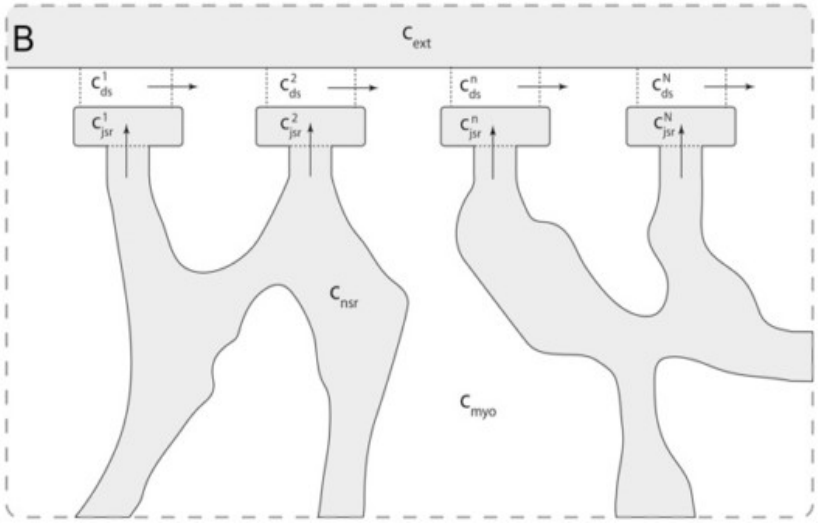
2	2	4
1	4	x

GEORGE MASON UNIVERSITY

Case study: computational model to study Ca leak

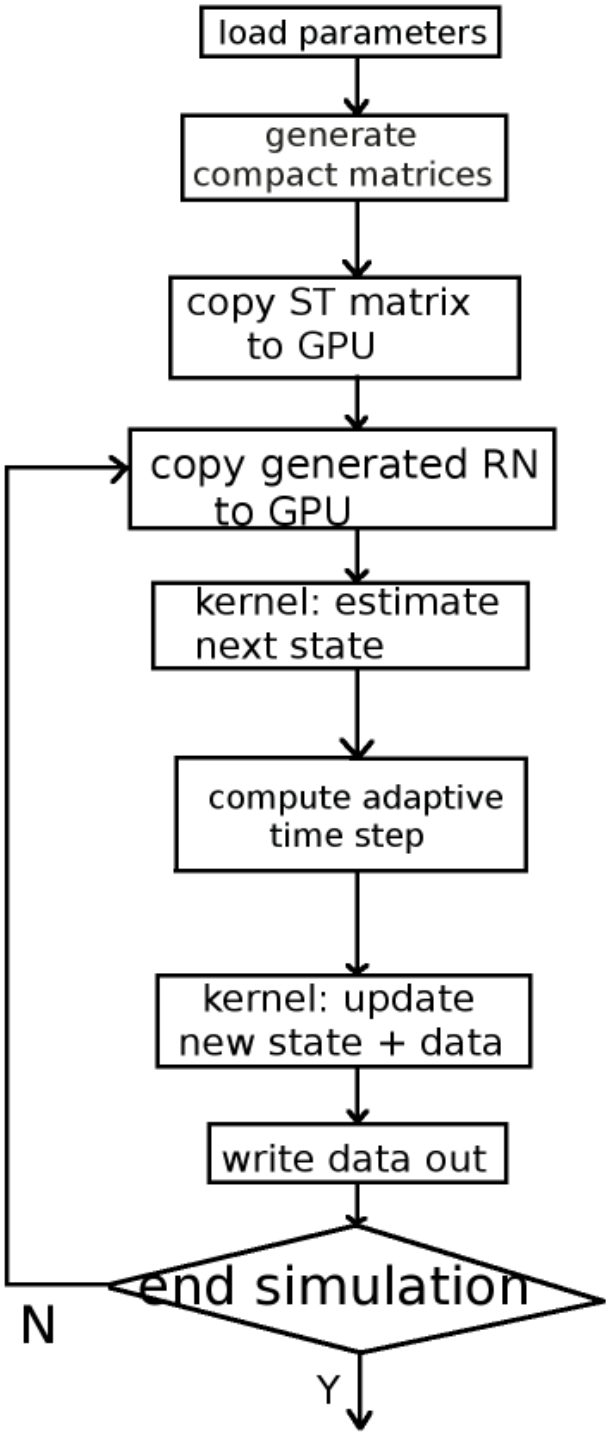
- RyR
  - Minimal model with 2 states
- Release site:
  - 10,000
  - Each site has 50 RyR
- Whole-cell model





Algorithm on GPU

- Each release site → single core (SP)



Algorithm on GPU

```
value     description
# NOTE: Only a single digit before decimal point
# First 14 columns are for numbers

# release site configuration
5.0d2          N  (number of release site)
5.0d1          nR (number of RyR/release site)
7.0d0          nL (number of DHPR/release site)
6.0d0          mL (number of DHPR state)
2.0d0          mR (number of RyR state)

# simulation settings
#-8.7d1         V_rest
-8.0d1         V_rest
-1.0d1         V_step (-20, 0 (0.0001), 20) mV (triggered voltage)
0.0d0          t_start (time start)
0.5d0          t_end   (time end)
1.0d-7         t_step  (time step)
0.3d0          t_start_V_clamp
0.4d0          t_end_V_clamp
1.0d1          output_interval (how often to write data out)

# initial condition
1.4d5          Na_o (ion concentration)
1.02d4         Na_i
1.8d3          Ca_o
1.0d0          Cm (membrane capacitance) uF/cm^2

# constant
9.6485d4       F (Faraday)  C/mol
8.314d3        R (universal gas) mJ mol-1 K-1
3.10d2         T (temperature) K

2.2d0          eta (hill coefficient)
0.07d0         Ej
-0.89d0        Ecc
-0.89d0        Eoo
```

GEORGE MASON UNIVERSITY

Algorithm on GPU

```
!!       Compute random transition number.
CALL DRANDUNIFORM(noutincr * NSFU, LB, UB,STATE,X,INFO)
X_dev = X

DO iinner=1,noutincr  ! inner loop
   CALL getCompPdt<<<NSFU/blocksize, blocksize>>> &
        (NSFU, N_R, maxnklm, maxnklp, irow_R, &
        Ej, Ecc, Eoo, eta)
   cuErr = cudaThreadSynchronize()
```

```
CALL  updateCalcium<<<NSFU/blocksize, blocksize>>> &
     (NSFU, dt, maxnklm, maxnklp, &
     X_dev(NSFU*(iinner-1)+1:NSFU*iinner), &
     lambda2, v_ryr, v_refill, v_efflux, Ca_nsr, Ca_myo)
cuErr = cudaThreadSynchronize()
```

GEORGE MASON UNIVERSITY

Benchmark

- System:
  - PGI Fortran 10.5 + CUDA Fortran
  - NVIDIA CUDA 2.3 SDK
  - Tesla C1060
  - Intel Nehalem i7
  - 12GB RAM
  - Double-precision computing

Benchmark

- Why GPU?
  - Large amount of computation can be done in parallel
  - State space is reused at every computational step → compact form make it fit to GPU device space
- Programming issue:
  - Kernel configuration
    - Block size = 32x??? (128, 160, 192, 256, 512)
    - Grid = 1D
    - Occupancy ???
  - Data alignment
    - 10112 or 10028 or 10240 release sites
  - Memory access latency:
    - Registers (16K)
    - Shared memory
    - Global device memory

Benchmark

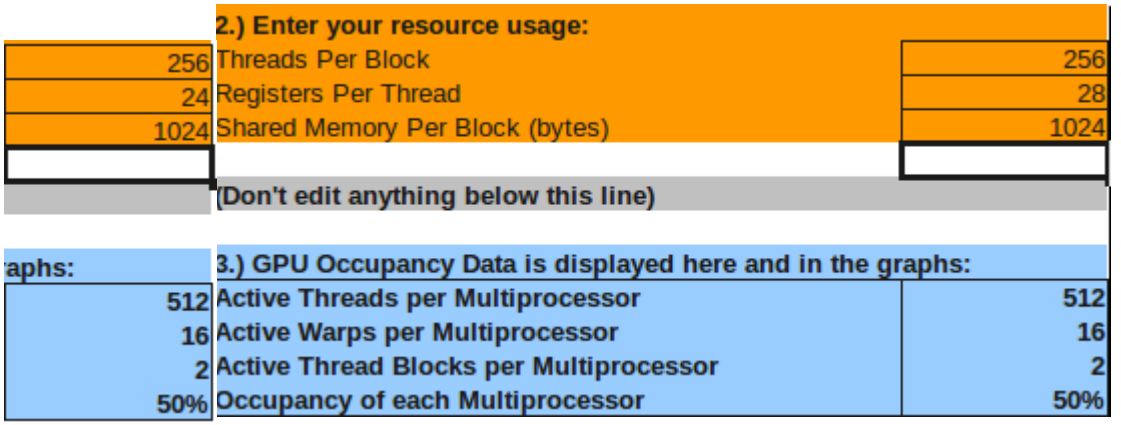
Method	Original	MATLAB	Fortran	GPU
Runtime	11000 min	110 min	20 min	45 sec
Speedup	1x	100x	550x	14,667x

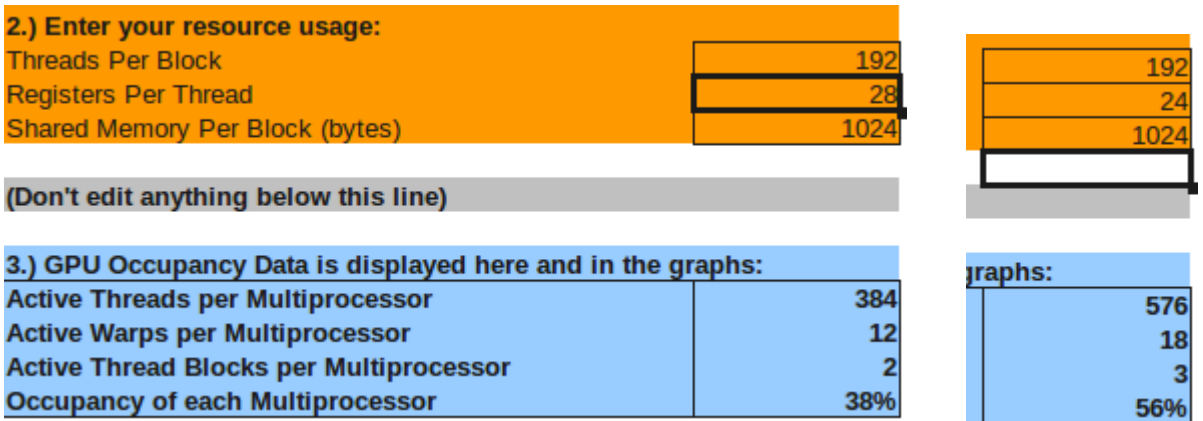
- 10,000 release units
- 50 RyRs
- 1 second simulation time

Benchmark

- GPU benchmark

128-10112	160-10080	192-10176	256-10240	512-10240	configuration
179.93	45.65	40.56	44.38	44.21	time (sec)





Conclusion

- Simulations suggest a mechanism for the basis of calcium leak from the SR in cardiac myocytes.
- Our Ultrafast Monte Carlo Method make stochastic simulation of calcium dynamics possible.
- The efficiency is such that these methods can be applied to (1) whole-cell detailed model, (2) networks of cardiac myocytes to study calcium dysfunction leading to arrythmia.

Ongoing research

- Temporospatial whole-cell model on GPU
  - 1D, 2D and 3D
- Tissue level (multiple GPU)
  - A network of multiple cells
  - Studying the pathogenesis of cardiac arrhythmias using this model, and compare with experimental data