

Original citation:

Caudron, Quentin, Donnelly, Simon R., Brand, Samuel P. C. and Timofeeva, Yulia.
(2012) Computational convergence of the path integral for real dendritic morphologies.
The Journal of Mathematical Neuroscience, Volume 2 (Number 1). ISSN 2190-8567

Permanent WRAP url:

<http://wrap.warwick.ac.uk/54527>

Copyright and reuse:

The Warwick Research Archive Portal (WRAP) makes this work of researchers of the University of Warwick available open access under the following conditions.

This article is made available under the Creative Commons Attribution 2.0 Generic (CC BY 2.0) license and may be reused according to the conditions of the license. For more details see: <http://creativecommons.org/licenses/by/2.0/>

A note on versions:

The version presented in WRAP is the published version, or, version of record, and may be cited as it appears here.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

warwick**publications**wrap

highlight your research

<http://go.warwick.ac.uk/lib-publications>

Computational Convergence of the Path Integral for Real Dendritic Morphologies

Quentin Caudron · Simon R. Donnelly · Samuel P.C. Brand · Yulia Timofeeva

Received: 19 June 2012 / Accepted: 11 September 2012 / Published online: 22 November 2012

© 2012 Q. Caudron et al.; licensee Springer. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/2.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract Neurons are characterised by a morphological structure unique amongst biological cells, the core of which is the dendritic tree. The vast number of dendritic geometries, combined with heterogeneous properties of the cell membrane, continue to challenge scientists in predicting neuronal input-output relationships, even in the case of sub-threshold dendritic currents. The Green's function obtained for a given dendritic geometry provides this functional relationship for passive or quasi-active dendrites and can be constructed by a sum-over-trips approach based on a path integral formalism. In this paper, we introduce a number of efficient algorithms for realisation of the sum-over-trips framework and investigate the convergence of these algorithms on different dendritic geometries. We demonstrate that the convergence of the trip sampling methods strongly depends on dendritic morphology as well as the biophysical properties of the cell membrane. For real morphologies, the number of trips to guarantee a small convergence error might become very large and strongly affect computational efficiency. As an alternative, we introduce a highly-efficient matrix method which can be applied to arbitrary branching structures.

Keywords Dendrites · Path integral · Sum-over-trips · Morphology · Dendritic computation

Q. Caudron (✉) · S.P.C. Brand · Y. Timofeeva
Centre for Complexity Science, University of Warwick, Coventry, CV4 7AL, UK
e-mail: q.caudron@warwick.ac.uk

Q. Caudron · Y. Timofeeva
Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK

S.R. Donnelly
Doctoral Training Centre in Neuroinformatics and Computational Neuroscience, University of Edinburgh, Edinburgh, EH8 9AB, UK

S.P.C. Brand
Mathematics Institute, University of Warwick, Coventry, CV4 7AL, UK

1 Introduction

Discovered more than a century ago by Santiago Ramón y Cajal [1], dendrites form the vast majority of the surface area of a neuron, with the dendritic trees of some motoneurons representing up to 97% of total neuronal surface area and 75% of the total neuronal volume [2]. These complex branching structures are responsible for transferring electrical activity between synapses and the soma. As technology evolved, interest in dendrites began to gather momentum, with the invention of sharp micropipette electrodes in the early 1950s allowing intracellular recordings to be made. It was the breakthrough work of Wilfrid Rall [3] on the application of cable theory to dendritic modelling that provided significant insight into the role of dendrites in processing synaptic inputs, the historical perspective of which is summarised in a book by Segev, Rinzal and Shepherd [4]. Recent experimental and theoretical studies reinforce the fact that dendritic morphology and membrane properties play an important role in dendritic integration [5, 6]. We refer the reader to the book *Dendrites* [7], devoted exclusively to these formations and revealing their biological complexity at different scales.

It has also been known for some time that nonlinear voltage-gated ion channels are present in the dendrites of various types of neurons [8], and many recent dendritic models are constructed by combining the linear (passive) properties of dendrites together with nonlinear (active) dynamics of membrane channels. Although the nonlinear properties of ion channels contribute considerably to neuronal input-output relations, it is important to recognise that the passive properties of dendritic membranes provide the fundamental core for signal filtration and integration, and thus remain an essential component in understanding electrical signalling in dendrites [9].

When branched dendritic fibres are modelled by passive cable equations, the voltage response across the branching structure for any form of applied current can be calculated via a convolution operation, as long as the Green's function for the given dendritic tree is found. This approach provides an alternative to the compartmental method, based on the discrete spatial approximation of the potential [10, 11]. It is not always trivial to construct such a Green's function for realistic dendritic geometries. Arbitrarily-branching systems are inherently difficult to solve, a fact recognised early by Rall, who proposed a method of mapping the branching structure onto an equivalent cylinder provided that certain geometrical restrictions were satisfied [12]. The work of Koch and Poggio [13], based on the graphical calculus of Butz and Cowan [14], focused on the calculation of the response function for complete dendritic trees in the Laplace (frequency) domain. Later, Rall's method of equivalent cylinders was extended by releasing the constraints on diameters of individual branches and by constructing the Green's function, again in the Laplace domain [15, 16]. An alternative method for constructing the Green's function for a branching structure with a shunted soma was proposed by Evans and coauthors [17–19]. In this series of papers, the response function was found in the form of an eigenfunction expansion, which converges particularly rapidly for large times. For smaller times, a Laplace-domain series solution provides better accuracy, agreeing well with an earlier “sum-over-trips” method for constructing the Green's function directly in the time domain, proposed by Abbott et al. [20]. This sum-over-trips framework is built on a path integral formulation and enables the calculation of the Green's function on an arbitrary

dendritic geometry as a convergent infinite series solution. Cao and Abbott [21] presented an algorithm for a computational realisation of the sum-over-trips approach, based on the division of trips into four classes. They applied this algorithm to a number of sample dendritic trees, the largest of which had 22 branches, in contrast to real dendritic geometries, which might have more than 400 terminals alone [22], with a large variation in branch length. This complexity in neuronal morphologies across different types of neurons is expected to affect the convergence of computational implementations of the sum-over-trips framework.

In this paper, we introduce and investigate a number of efficient algorithms for calculating the Green’s function on dendritic trees using the sum-over-trips formalism. In Sect. 2, we review the theoretical framework and the four-classes algorithm of Cao and Abbott [21], and introduce alternative algorithms for the sum-over-trips method in Sect. 3. We begin with a modification on the four-classes algorithm aimed at improving its time complexity by developing a formal grammar to derive the trips. Then, a length-priority ordering of the trips using Eppstein’s algorithm [23] for finding the k shortest trips on a graph is proposed. We also derive a stochastic approach for sampling trips on the tree based on a Monte-Carlo approach. Finally, a highly-efficient deterministic method for discretised tree structures is described. We assess the convergence of the introduced algorithms on different dendritic geometries in Sect. 4, where we also compare the delay and attenuation of voltage spread on four reconstructed dendritic morphologies. Finally, in Sect. 5, we provide a discussion of our results, as well as possible extensions of this work.

2 The Sum-over-trips Framework

We consider a dendritic branching structure with the dynamics of the membrane voltage on a finite branch i described by the passive cable equation. An external current $I_j(t)$ is injected at a location y on branch j . The transmembrane voltage across the dendritic tree is then described by the following set of equations:

$$\pi a_i C \frac{\partial V_i}{\partial t} = \frac{\pi a_i^2}{4R_a} \frac{\partial^2 V_i}{\partial x^2} - \frac{\pi a_i}{R} V_i, \quad 0 \leq x \leq L_i, i \neq j \tag{1}$$

$$\pi a_j C \frac{\partial V_j}{\partial t} = \frac{\pi a_j^2}{4R_a} \frac{\partial^2 V_j}{\partial x^2} - \frac{\pi a_j}{R} V_j + \delta(x - y) I_j(t), \quad 0 \leq x \leq L_j. \tag{2}$$

Here, a_i is the diameter of branch i (measured in μm), R_a is the specific cytoplasmic resistivity (in $\Omega \text{ cm}$), C is the specific membrane capacitance (in $\mu\text{F cm}^{-2}$), and R is the resistance across one unit area of passive membrane (in $\Omega \text{ cm}^2$). Introducing the electrotonic space constant $\lambda_i = \sqrt{a_i R / (4R_a)}$, the membrane time constant $\tau = RC$ and the diffusion coefficient $D_i = \lambda_i^2 / \tau$, Eqs. (1) and (2) can be rewritten as

$$\frac{\partial V_i}{\partial t} = D_i \frac{\partial^2 V_i}{\partial x^2} - \frac{V_i}{\tau}, \quad 0 \leq x \leq L_i, i \neq j, \tag{3}$$

$$\frac{\partial V_j}{\partial t} = D_j \frac{\partial^2 V_j}{\partial x^2} - \frac{V_j}{\tau} + \frac{1}{\pi a_j C} \delta(x - y) I_j(t), \quad 0 \leq x \leq L_j. \tag{4}$$

In addition to these equations, the appropriate boundary conditions must be specified at all branching nodes and terminals: continuity of the potential across a node and Kirchoff’s law of conservation of current. Continuity of the potential requires that, for all pairs of branches m and n attached to a node,

$$V_m(L_m, t) = V_n(0, t),$$

where the distal end of branch m is connected to the proximal end of branch n . Conservation of current for the same node imposes

$$\sum_m \frac{1}{r_m} \frac{\partial V_m}{\partial x} \Big|_{x=L_m} = \sum_n \frac{1}{r_n} \frac{\partial V_n}{\partial x} \Big|_{x=0},$$

where $r_n = 4R_a/(\pi a_n^2)$ is the axial resistance on branch n (in $\Omega \text{ cm}^{-1}$), and each sum is over all branches connected to this node either with their distal or proximal ends. At individual terminals, we can either impose a closed-end boundary condition,

$$\frac{\partial V_k}{\partial x} \Big|_{x=L_k} = 0,$$

or an open-end boundary condition,

$$V_k(L_k, t) = 0,$$

where $x = L_k$ is a terminal on branch k .

When the injected current has the form of a delta pulse, that is, $I_j(t) = \delta(t)$, the solution to Eqs. (3) and (4) is the Green’s function $G_{ij}(x, y, t)$ which can be found as

$$G_{ij}(x, y, t) = \frac{1}{\pi a_j C} \sum_{\text{trips}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}, t), \tag{5}$$

where the sum is over all trips (more formally, graph-theoretic walks), starting at x and finishing at y , and describes the time-course of the membrane voltage at the location x on branch i in response to the injected current at the location y on branch j , where i can be taken to equal j if desired. The function G_{∞} takes the form

$$G_{\infty}(L_{\text{trip}}, t) = \frac{1}{\sqrt{4\pi t D_j}} e^{-(L_{\text{trip}})^2 \tau / (4t)} e^{-t/\tau}, \tag{6}$$

where $L_{\text{trip}} = L_{\text{trip}}(x/\lambda_i, y/\lambda_j)$ is the length of a trip along the tree that starts at point x/λ_i on branch i and ends at point y/λ_j on branch j . Note that the length of each branch needs to be scaled by its own electrotonic space constant before L_{trip} is calculated for Eq. (6). A constructed trip is allowed to reflect on or pass through any node on the tree an arbitrary number of times. The coefficients A_{trip} depend on the constructed trip and are determined according to the following rules [20]:

- From any starting point, $A_{\text{trip}} = 1$.

- For every node at which the trip passes from branch m to branch k where $m \neq k$, A_{trip} is multiplied by a factor $2p_k$.
- For every node at which the trip reflects along on a node back onto the same branch n , A_{trip} is multiplied by a factor $2p_n - 1$.
- For every terminal, A_{trip} is multiplied by $+1$ for the closed-end boundary condition or by -1 for the open-end boundary condition.

When the electrical properties of the cell membrane are identical for all branches, the factors p_k are defined as

$$p_k = \frac{a_k^{3/2}}{\sum_m a_m^{3/2}}, \tag{7}$$

where the sum is over all branches m connected to the node. When the parameters R and R_a vary from branch to branch, the expression (7) must be modified:

$$p_k = \frac{(\lambda_k r_k)^{-1}}{\sum_m (\lambda_m r_m)^{-1}}. \tag{8}$$

However, note that the sum-over-trips method for constructing the Green’s function in the time domain only works for uniform characteristic time constant τ across the entirety of the dendritic tree. The generalisation of this framework to support a quasi-active membrane, instead of a passive membrane, releases this restriction and different cell membrane properties can be chosen on each branch [24]. However, this means that the construction of the Green’s function as an infinite series solution can only be performed in the Laplace domain.

Knowing the Green’s function for a given dendritic structure allows one to find the voltage response along the entire tree. By finding $G_{ij}(x, y, t)$ for the ordered pair (x, y) , the Green’s function $G_{ji}(y, x, t)$ can be found using a simple reciprocity identity:

$$G_{ji}(y, x, t) = \frac{D_j r_j}{D_i r_i} G_{ij}(x, y, t). \tag{9}$$

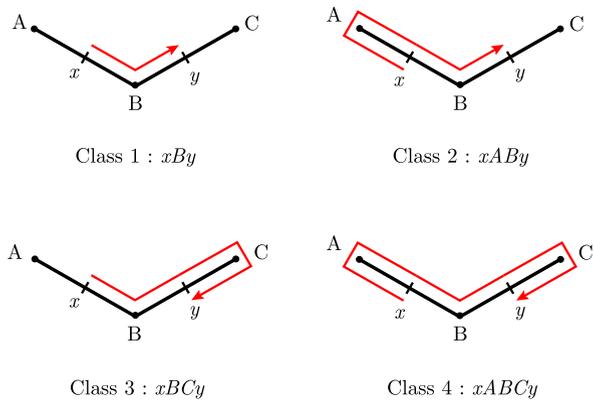
The voltage response can then be found for an arbitrary number of different discrete inputs as a sum of convolution integrals:

$$V_i(x, t) = \sum_j \int_0^t G_{ij}(x, x_j, t - s) I_j(s) ds, \tag{10}$$

where x_j is a location of a stimulus $I_j(t)$ on branch j .

The Green’s function calculated by Eq. (5) for any branching structure with finite length branches includes an infinite number of terms. It is possible to show that this infinite series solution converges faster than e^{-k} , for sufficiently-high k , the number of nodes visited by the trip. We demonstrate this in the Appendix for an arbitrary tree with nodes of degree $d = 3$ or less. This generalises Abbott’s convergence analysis [25], where it was shown that, for an infinite binary tree, the sum of coefficients A_{trip} is $\mathcal{O}(1)$ for trips visiting any number of nodes.

Fig. 1 Four classes of trips between two points. *Class 1* is the most direct trip, leaving x in the direction of y and not going past y before finishing (xBy). *Class 2* leaves x in the other direction, but finishes when it meets y ($xABy$). *Class 3* moves from x towards y , but goes past y and changes direction immediately after passing it before finishing (BCy). *Class 4* trips move from x first away from point y and pass y before reflecting on the next node and finishing ($ABCy$)



2.1 Four-Classes Algorithm

Cao and Abbott [21] introduced an algorithm for constructing the Green’s function using the sum-over-trips method. Their algorithm is based on finding the shortest trips between any two points of measurement x and current injection y on a tree. Starting from the most direct, shortest trip from x to y , passing through the minimum number of nodes, four classes of trips are defined by allowing a trip to leave the point x in either direction and approach y from either direction along their respective branches. These initial trips, therefore, form the first and shortest trips in their respective classes; longer trips are generated incrementally from these. New additional trips can pass the points x and y any number of times and are allowed to change direction at any node. We will refer to this method as the four-classes algorithm.

Figure 1 shows a model branching structure with two points x and y , and the four shortest classes of trips between them. Trips are represented as sequences of node identifiers, beginning and ending with x and y respectively. For example, we denote a trip from x to y via nodes A , B and C by its full description, $xABCy$.

From these main trips, the four-classes algorithm generates all $x \rightarrow y$ trips by inserting what are described as “excursions” into the trips. If A and B are adjacent nodes in a tree, then an excursion could be added to the trip xBy to generate the trip $xBABy$, representing a reflection on node B towards A , reflecting at the terminal A back towards B , passing through this node and finally onto point y . This process can be iterated indefinitely, generating a trip with two more nodes each time. If this process is applied to every node on every trip with n and $n + 1$ nodes, then every trip with $n + 2$ and $n + 3$ nodes will be generated. Thus, from the four shortest $x \rightarrow y$ trips on the tree, it is possible to construct all trips up to some threshold number of nodes in length explicitly. The lengths and coefficients of these trips can then be calculated from their full trip descriptions, allowing the Green’s function given by Eq. (5) to be approximated.

3 Algorithmic Realisations

Here, we suggest possible modifications to the four-classes algorithm of Cao and Abbott [21] as well as introduce novel alternative algorithms for the sum-over-trips formalism.

3.1 Formal Language Theory Approach

The four-classes algorithm generates duplicate trips [21], which must then be removed by a binary search through the list of existing trips for every new trip generated, which takes $\mathcal{O}(k \log k)$ time overall, where k is the number of trips constructed. There are two different mechanisms by which duplicate trips are generated, and both mechanisms can be eliminated by applying simple restrictions to the choice of excursions applicable to a trip. As an example of the first, for the tree in Fig. 1, it is possible to generate the trip $x B A B C B y$ in two different ways from the shortest Class 1 trip, $x B y$:

$$\begin{aligned}
 x B y &\xrightarrow{\text{Excursion } B \rightarrow B A B} x B A B y \xrightarrow{\text{Excursion } B \rightarrow B C B} x B A B C B y, \\
 x B y &\xrightarrow{\text{Excursion } B \rightarrow B C B} x B C B y \xrightarrow{\text{Excursion } B \rightarrow B A B} x B A B C B y.
 \end{aligned}$$

Due to the fact that the excursion may be added at any step in the trip (at the first or second B), the same trip may be generated multiple times. If we insist that excursions cannot be added at any step that precedes the excursion most recently added to the trip, this can be prevented. In the theory of context-free grammars, this is equivalent to requiring a leftmost derivation. We can represent this using a symbol to separate the mutable and immutable parts of the trip:

$$\begin{aligned}
 x|B y &\xrightarrow{\text{Excursion } B \rightarrow B|A B} x B|A B y \xrightarrow{\text{Excursion } B \rightarrow B|C B} x B A B|C B y, \\
 x|B y &\xrightarrow{\text{Excursion } B \rightarrow B|C B} x B|C B y \not\rightarrow x B A B C B y.
 \end{aligned}$$

The second mechanism by which duplicate trips are produced is the addition of excursions along the same branch, starting from either end. In the structure in Fig. 1, we have both $A \rightarrow A|B A$ and $B \rightarrow B|A B$. Hence, in spite of the leftmost derivation rule, we can generate $x A B A B y$ in two different ways (brackets added for clarity):

$$\begin{aligned}
 x|A B y &\rightarrow x(A|B A) B y, \\
 x|A B y &\rightarrow x A(B|A B) y.
 \end{aligned}$$

This problem can be avoided by assigning each branch a direction. If the branch $A B$ is given the direction $\overrightarrow{B A}$, then the excursion $A \rightarrow A|B A$ is disallowed. The choice of direction for each branch is unambiguous on acyclic structures: apart from the branch on which x is found, each branch must be directed away from x . The branch upon which x resides is directed away from y . This ensures that each node has a sequence

of excursions that allow the algorithm to generate trips including it. The allocation of direction to each branch can be performed before the process of generating trips and may coincide with finding the four main classes of trips. These modifications require that the graph be acyclic, since “away from a point” is not generally definable on a graph with cycles. There do exist cyclic graphs for which an unambiguous grammar can generate the language of $x \rightarrow y$ trips, but these are not relevant to the study of single dendritic trees.

The two presented modifications of the four-classes algorithm are sufficient to prevent the generation of any duplicate trips, without any trips being missed. Together, they provide an unambiguous context-free grammar generating the language of $x \rightarrow y$ trips.

3.2 Length-Priority Method

Since the coefficients A_{trip} decay at most with $e^{-L_{\text{trip}}}$ (although the number of trips increases with $e^{L_{\text{trip}}}$), the dominating term in the Green’s function (5) is the exponential decay $e^{-L_{\text{trip}}^2}$ in G_{∞} . The four-classes algorithm [21] does not generate trips in monotonic order in length, since trips are constructed by adding the same excursion to all four classes of trips. If, for example, a Class 2 trip is significantly longer than its Class 1 counterpart, due to x being along a long edge but close to a node, then a longer Class 2 trip will be generated before a potentially shorter Class 1 trip having an additional excursion on a shorter branch. In general, trips are likely to be disordered in length if the branches upon which x or y reside are substantially longer than at least one other branch on the tree, or if x or y are much closer to one of their adjacent nodes than to the other.

Here, we propose to realise the sum-over-trips framework by a length-priority method. In this implementation, trips are generated and the corresponding terms $A_{\text{trip}}G_{\infty}(L_{\text{trip}}, t)$ are added to the infinite series solution (5) in monotonic order in length L_{trip} . This is achieved by incorporating Eppstein’s algorithm [23] for finding the k shortest trips on a graph in $\mathcal{O}(m + n \log n + k)$ time, with n being the number of nodes and m the number of edges on the branching structure.

Both the four-classes algorithm and the improvements described in the language-theoretic approach rely on storing trips explicitly as sequences of nodes. This consumes $\mathcal{O}(kn)$ space and time for k trips with n nodes but allows on-the-fly calculation of coefficients A_{trip} . This is contrary to Eppstein’s algorithm [23], which stores trips using an implicit representation and allows us to find the k shortest trips implicitly using only $\mathcal{O}(1)$ space and time for each trip. The current implementation, based on Eppstein’s algorithm, requires $\mathcal{O}(kn)$ time to calculate coefficients despite the savings on space due to the implicit trip representation. However, Eppstein provides a method for computing any property that can be described by a monoid in $\mathcal{O}(1)$ time per trip. Such a description of coefficient calculation exists, and its use would supplement the current $\mathcal{O}(kn)$ to $\mathcal{O}(k)$ decrease in space requirements with an analogous decrease in time complexity. The savings in space already allow the length-priority method to scale better than the four-classes algorithm.

3.3 Monte-Carlo Method

The path integral formulation of the solution to the cable equation introduced by Abbott et al. [20] is derived via consideration of a Feynman–Kac representation of the solution in terms of random walkers on the dendritic geometry. Hence, it is natural to consider Monte-Carlo approaches to evaluating this path integral. Instead of a length-ordered series solution as provided by the length-priority approach, the Green’s function (5) can be constructed using a stochastic algorithm. The aim of this approach is to sample from trips $x \rightarrow y$ in such a way that the probabilistically more likely samples coincide with the trips that contribute most to the series solution (5).

To motivate this Monte-Carlo approach, let us consider a linear diffusion equation along an infinite one-dimensional cable,

$$\frac{\partial \mathcal{G}}{\partial t} = D \frac{\partial^2 \mathcal{G}}{\partial x^2}, \quad t \in [0, T], \tag{11}$$

satisfying the initial condition $\mathcal{G}(x, 0) = \delta(x - y)$. Analogously, a diffusion process for the state variable X_t can be defined by the stochastic equation

$$dX_t = \sqrt{2D} dW_t, \tag{12}$$

with the Wiener process W_t and the initial condition $X_0 = y$. It is well known that Eq. (11) is the Kolmogorov equation of the diffusion process (12), that is, the time evolution equation of the probability density for the state of the diffusion (12). On the one hand, solution of (11) via classical numerical or analytical methodology informs the probability density of X_t ; on the other hand, repeated sampling from (12) converges upon the solution $\mathcal{G}(x, t)$ of (11). This method of sampling from random walks can also be applied for arbitrary geometries by setting the appropriate boundary conditions at the branching nodes and terminals. Knowing $\mathcal{G}_{ij}(x, y, t)$ on a branching structure, we can easily find a solution of the cable equation on this geometry using the relation

$$G_{ij}(x, y, t) = \mathcal{G}_{ij}(x, y, t)e^{-t/\tau}.$$

Because the path integral form of the solution is equivalent to the expectation of a function on random walks upon the branching points of the dendritic tree, reduction of the random walk problem from the complete continuous space geometry of the neuron to the discrete topology of the branching points of the neuron gives a considerable efficiency saving to a Monte-Carlo solver. We introduce a parameter k_{\max} , the maximum number of discrete hops on nodes for which we wish to calculate the expectation. The maximum number is based upon the effective maximum range of diffusion during the interval $[0, t]$. Then, we generate a realisation of a random walk on the nodes,

$$\omega = (\omega_1, \omega_2, \dots, \omega_{k_{\max}}), \tag{13}$$

where each ω_k is a label identifying a particular node. For trips $x \rightarrow y$ we select ω_1 such that it is either of the two nodes adjacent to the branch containing x , with

equal probability. By indexing a branch between two nodes, ω_{k-1} and ω_k , as the k th branch, subsequent steps are performed with the transition probability

$$P(\omega_k|\omega_{k-1}) = p_k, \quad 2 \leq k \leq k_{\max}, \tag{14}$$

where p_k is given by (7). This connects the Monte-Carlo method to the earlier discussed path enumeration methods. Here, we introduce two auxiliary functions, ϕ and \tilde{a} , of subwalks of ω . The first is a function indicating whether a subwalk of k steps on a realisation ω is a valid trip, and is defined by

$$\phi(y, k, t, \omega) = \begin{cases} G_\infty(L(y, k, \omega), t), & \text{if } \omega_{k-1} \text{ and } \omega_k \text{ are the nodes adjacent to } y, \\ 0, & \text{otherwise,} \end{cases}$$

where k is the number of hops on nodes in the subwalk and $L(y, k, \omega)$ is the length of the subwalk. The other auxiliary function \tilde{a} is defined as

$$\tilde{a}(k, \omega) = \begin{cases} 1, & \text{if } k = 1, 2, \\ 2, & \text{if } \omega_{k-2} \neq \omega_k, \\ (2p_k - 1)/p_k, & \text{if } \omega_{k-2} = \omega_k, \\ 1, & \text{if at a closed terminal (this takes priority).} \end{cases}$$

The relevant function on paths can be defined as a composite of the auxiliary functions described above:

$$\tilde{A}(y, \omega, t) = \sum_{k=1}^{k_{\max}} 2\phi(y, k, t, \omega) \left(\prod_{i=1}^k \tilde{a}(i, \omega) \right).$$

The expectation of \tilde{A} with respect to the random walk (14) is equivalent to solving for the path integral, up to some value of k_{\max} at time t :

$$\begin{aligned} \mathbb{E}_P[\tilde{A}(y, \omega, t)] &= \sum_{\omega} P(\omega) \tilde{A}(y, \omega, t) \\ &= \sum_{\omega} \sum_{k=1}^{k_{\max}} 2P(\omega) \phi(y, k, \omega) \left(\prod_{i=1}^k \tilde{a}(i, \omega) \right) \\ &= \sum_{\substack{\omega: \\ x \rightarrow y \\ \text{at } k}} \sum_{k=1}^{k_{\max}} 2P(\omega) G_\infty(L(y, k, \omega), t) \left(\prod_{i=1}^k \tilde{a}(i, \omega) \right) \\ &= \sum_{\substack{\text{trips} \\ x \rightarrow y}} A_{\text{trip}} G_\infty(L_{\text{trip}}, t), \end{aligned}$$

where $P(\omega)$ is the probability of the realisation ω , and \mathbb{E} denotes the expectation operator. Therefore, the Monte-Carlo strategy is to sample, sequentially or in parallel, the random function \tilde{A} in order to construct this expectation.

3.4 Matrix Method

An alternative method of constructing the sum-over-trips series solution is by grouping trips by their lengths:

$$\sum_{\text{trips}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}, t) = \sum_l G_{\infty}(l, t) \sum_{\substack{\text{trips with} \\ L_{\text{trip}}=l}} A_{\text{trip}},$$

where the sum over l is over all possible trip lengths L_{trip} . On a dendritic tree, discretised as in compartmental models [10] or in a manner similar to the discretisation of the tree into *segments* in NEURON [26], grouping trips according to their lengths allows us to count the number of trips of a given length l without having to explicitly construct them.

This method uses a modified directed edge adjacency matrix of the discretised tree in order to compute the sum of coefficients of trips of a given length. It requires all compartments to have the same fixed length Δx , although this restriction can be relaxed in a generalisation presented at the end of this section. The extremities of compartments define the position of nodes; there is a directed edge in both directions between adjacent nodes.

We begin by defining \mathcal{V} as the set of nodes and \mathcal{E} as the set of directed edges in the discretised tree. Edges are ordered pairs of nodes: $e = (u, v) \in \mathcal{E}$ is a directed edge from u to v , with $u, v \in \mathcal{V}$. For any edge $e = (u, v)$, we denote the reverse edge by $e' = (v, u)$. Trips are taken to begin from a point x along a starting edge $s = (s_1, s_2)$ and end at a point y along a goal edge $g = (g_1, g_2)$ for $s, g \in \mathcal{E}$. We say that $x \in s$ or $x \in (s_1, s_2)$ if x resides along edge $s = (s_1, s_2)$. Based on the locations of $x \in s$ and $y \in g$, the orientations of s and g are defined such that the shortest $x \rightarrow y$ trip satisfies $x \rightarrow s_2 \rightarrow \dots \rightarrow g_1 \rightarrow y$. Therefore, the shortest $x \rightarrow y$ trip always starts on edge s , that is, in the $s_1 \rightarrow s_2$ direction, and approaches y along the edge g , in the $g_1 \rightarrow g_2$ direction. This is equivalent to a Class 1 trip; Class 2 trips leave x along the $s' = (s_2, s_1)$ edge, arrive at $y \in g$; Class 3 trips go from $x \in s$ to $y \in g'$; Class 4 trips, finally, go from $x \in s'$ to $y \in g'$. The locations of the points $x \in s$ and $y \in g$ along their respective edges are given as a fraction of the branch length such that $x\Delta x$ denotes the distance from x to s_2 and $y\Delta x$ is the distance between node g_1 and point y . We distinguish between k , the number of edges travelled in a particular trip, from the length of the trip L_{trip} . Because x and y reside along their respective edges, the total length of a trip that travels along k edges is less than if the full distance along k edges had been travelled. That is, $L_{\text{trip}} < k\Delta x$ for any combination of x, y and for all k .

The aim of the matrix method is to group all trips starting on a given edge and finishing on a target edge by their lengths, L_{trip} , and calculate the sum of the coefficients A_{trip} of those trips for each particular group instead of calculating coefficients individually for each trip. The sums of coefficients are computed simultaneously for trips ending on all edges, starting at a point $x \in (s_1, s_2)$, allowing us to compute G_{ij} for all j , for all x on edge i and for all y on any edge, in a single run of the calculation.

We define the coefficients function $c_k^s : \mathcal{E} \rightarrow \mathbb{R}$, $s \in \mathcal{E}$, as the sum of all coefficients A_{trip} which begin at point $x \in s$ and travel over k edges, finishing on a given edge g :

$$c_k^s(g) = \sum_{\substack{\text{trips} \\ x \rightarrow \dots \rightarrow y \\ \text{in } k \text{ jumps}}} A_{\text{trip}}, \quad x \in s, y \in g.$$

Because the set of edges \mathcal{E} is both ordered and finite, then $\mathbf{c}_k^s = (c_k^s(e_1), \dots, c_k^s(e_{|\mathcal{E}|})) \in \mathbb{R}^{|\mathcal{E}|}$ can be thought of as a vector, where $e_i \in \mathcal{E}$ for $i = 1, \dots, |\mathcal{E}|$. The i th element of the vector \mathbf{c}_k^s corresponds to the sum of coefficients A_{trip} for all trips originating at x on s and ending along the i th edge e_i , having travelled over k edges. The vector \mathbf{c}_1^s consists mostly of zeros, with a one only in the entry corresponding to the edge s , as the coefficient of moving in this direction remains 1, while all other moves are invalid by travelling over only one edge, and hence have coefficient 0.

We can now define a matrix $Q \in \mathbb{R}^{\mathcal{E} \times \mathcal{E}}$ such that

$$Q^k c_1 = c_{k+1}. \tag{15}$$

Q is a modified form of the edge-adjacency matrix, where instead of containing ones to denote edge adjacency and zero otherwise, it contains the coefficient taken in moving from one edge to another. The entries of Q can be computed based on the morphology of the graph. If the j th entry corresponds to edge (u, v) and the i th entry to edge (v, w) , then the entry Q_{ji} is the coefficient taken when moving from branch (u, v) to (v, w) . In the general case, these numerical values must be determined for each entry. However, in the simplified case where the radii on all branches are equal and all nodes have degree $d = 1$ or $d = 3$, the matrix Q can be constructed according to

$$Q_{ji} = \begin{cases} -\frac{1}{3}, & \text{if } j = (u, v) \text{ and } i = (v, u), \text{ where } v \text{ is a node of degree } d = 3, \\ 1, & \text{if } j = (u, v) \text{ and } i = (v, u), \text{ where } v \text{ is a closed terminal } (d = 1), \\ \frac{2}{3}, & \text{if } j = (u, v) \text{ and } i = (v, w), \text{ where } u \neq w, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Note that the above rules apply to the transpose of Q_{ij} .

Thus, knowing the matrix Q from the dendritic geometry and the vector \mathbf{c}_1^s from the starting edge s , it is possible to construct the sum of $c_k^s(g)$ terms, for all $k < k_{\text{max}}$, equal to the sum of coefficients for all trips travelling up to k_{max} edges, from $x \in s$ to $y \in g$. However, by considering trips moving from x in one direction only and arriving at y from only one direction, we have calculated the coefficients of just Class 1 trips. In order to find coefficients for the remaining three classes, we must also compute $c_k^s(g)$, $c_k^s(g')$ and $c_k^s(g'')$. These can be found in the same way as above. Using (15), the Green's function in (5) can therefore be written as

$$G_{ij}(x, y, t) = \sum_{\substack{\text{trips} \\ x \text{ to } y}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}, t)$$

$$\begin{aligned}
 &= \sum_{k=1}^{k_{\max}} [(Q^{k-1}c_1^s)_g G_\infty(L_1(k), t) \\
 &\quad + (Q^{k-1}c_1^{s'})_g G_\infty(L_2(k), t) \\
 &\quad + (Q^{k-1}c_1^s)_{g'} G_\infty(L_3(k), t) \\
 &\quad + (Q^{k-1}c_1^{s'})_{g'} G_\infty(L_4(k), t)], \tag{17}
 \end{aligned}$$

where $(Qc_1^s)_g$ is the g th element of the matrix-vector product of Q and c_1^s . Lengths L_1, \dots, L_4 are the lengths of Class 1 to Class 4 trips, respectively, and are defined as

$$\begin{aligned}
 L_1(k) &= \Delta x(2(k - 1) + x + y), \\
 L_2(k) &= \Delta x(2k - x + y), \\
 L_3(k) &= \Delta x(2k + x - y), \\
 L_4(k) &= \Delta x(2(k + 1) - x - y).
 \end{aligned}$$

By selecting a small Δx , branches may be approximated by a discretisation using an integer number of edges of length Δx . As in compartmental models, this allows the full morphology of the dendritic tree to be approximated, in a trade-off between high speed (large Δx) and accuracy (small Δx). As $\Delta x \rightarrow 0$, however, this approach tends to the computational complexity of naively integrating the cable equation using numerical methods. As in numerical simulations, where reducing Δx in order to increase accuracy brings about a necessary and associated change in Δt , the same is true of the matrix method: selecting a small Δx and hence increasing $|\mathcal{E}|$, implies that k_{\max} must be increased.

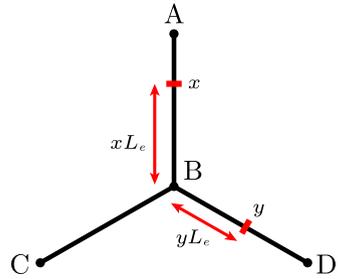
This algorithm can be generalised to accept several discrete edge lengths $\Delta x_1, \dots, \Delta x_n$, at an exponential cost in the number of different lengths n , allowing ‘‘caricature’’ neurons to be constructed from a small number of different edge lengths. Our description of this method is focused on the case where x and y are located on different branches. For computations where x and y are required to exist on the same edge, the edges can be discretised such that x and y appear on different segments. In all cases with bounded node degree, Q is a sparse matrix with only a few entries per row, and $\mathcal{O}(|\mathcal{E}|)$ entries altogether, making the complexity for the calculation of all coefficients $\mathcal{O}(|\mathcal{E}|k_{\max})$ by using highly-efficient sparse linear algebra algorithms.

3.4.1 Example calculation

Here, we demonstrate an example realisation of the matrix method for a dendritic structure of three branches of equal length Δx , shown in Fig. 2. In this symmetrical case, the matrix Q is very small and can be constructed by hand. We place the point of measurement x along edge $s = (A, B)$ and the point of current injection y along $g = (B, D)$.

We begin by ordering the edge pairs as follows: $(A, B), (B, A), (B, C), (C, B), (B, D), (D, B)$. Based on this ordered set, we can obtain two coefficients vectors $c_1,$

Fig. 2 A dendritic structure used in the example calculation for the matrix method



one for trips that begin at x and move towards B , denoted $c_1^s = (1\ 0\ 0\ 0\ 0\ 0)^T$, and another for trips moving from x towards node A , denoted $c_1^{s'} = (0\ 1\ 0\ 0\ 0\ 0)^T$. Using the rules described in (16), we can construct the matrix Q for our dendritic structure as follows:

$$Q = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{1}{3} & 0 & 0 & \frac{2}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & 0 & 0 & -\frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{2}{3} & 0 & 0 & \frac{2}{3} & 0 & -\frac{1}{3} \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Note that all rows and columns sum to 1. Knowing this matrix Q and breaking the trips into four main classes, it is straightforward to find the complete Green's function:

$$\begin{aligned} G_{sg}(x, y, t) &= \sum_{k=1}^{k_{\max}} (Q^{k-1} c_1^s)_g G_{\infty}(\Delta x(2n + x + y), t) \\ &+ \sum_{k=1}^{k_{\max}} (Q^{k-1} c_1^{s'})_g G_{\infty}(\Delta x(2n + 2 - x + y), t) \\ &+ \sum_{k=1}^{k_{\max}} (Q^{k-1} c_1^s)_{g'} G_{\infty}(\Delta x(2n + 2 + x - y), t) \\ &+ \sum_{k=1}^{k_{\max}} (Q^{k-1} c_1^{s'})_{g'} G_{\infty}(\Delta x(2n + 4 - x - y), t). \end{aligned} \tag{18}$$

4 Convergence of Methods

We first validate the computational implementations of our algorithms by constructing the Green's function $G_{ij}(x, y, t)$ on a small binary tree. Two profiles of the response function obtained by the length-priority, the Monte-Carlo and the matrix methods are shown in Fig. 3 and compared to a numerical simulation computed by

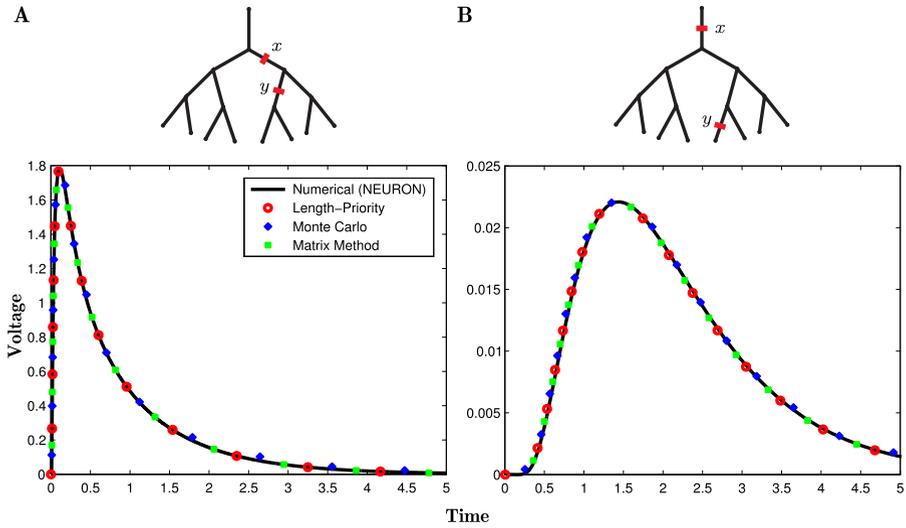


Fig. 3 The Green’s function constructed by a number of methods. The voltage traces show the analytical solutions $G_{ij}(x, y, t)$ for fixed x and y on a binary tree for the length-priority method (red circles), the Monte-Carlo method (blue diamonds) and the matrix method with $k_{max} = 100$ (green squares) superimposed on NEURON’s numerical solution (black line). Parameter set A (Table 1) is used in these computations

the software package NEURON [26]. These plots demonstrate an excellent agreement between a number of approaches for obtaining the Green’s function and the numerical solution, with a slightly worse performance of the Monte-Carlo method for larger times.

The computational convergence of any algorithm impacts both the accuracy of its results and the speed at which these results are obtained. While the series solution to the Green’s function (5) is proven to converge for a sufficiently-high number of terms (see the Appendix), this assumes optimal ordering of terms in the solution. Because the coefficients A_{trip} are impossible to compute until a trip is constructed, generating terms for the series solution in descending order of magnitude is inherently difficult. The four-classes, length-priority and matrix methods generate trips in order of increasing L_{trip} , with the aim of ordering trips by their $G_{\infty}(L_{trip}, t)$ terms, which decreases monotonically in the length of the trip. The Monte-Carlo method uses a stochastic method to order trips by their probabilities, with more likely trips contributing more to (5). However, none of these approaches order the trips optimally, and hence their accuracy relies not on the theoretical convergence of the mathematical method, but on the computational convergence of the algorithm that implements it.

To assess the computational convergence of the algorithms, Green’s function solutions were constructed on a number of branching geometries shown in Fig. 4. In addition to a binary tree with the topology in Fig. 4A, the algorithms were also applied to four real neuronal reconstructions obtained from the NeuroMorpho Database [27] in .swc format and shown in Figs. 4B–E. These files use a sequence of nodes with precise radii and three-dimensional locations to describe the location of the soma and the paths taken by the axon and each dendrite. A dendrite’s path can be described

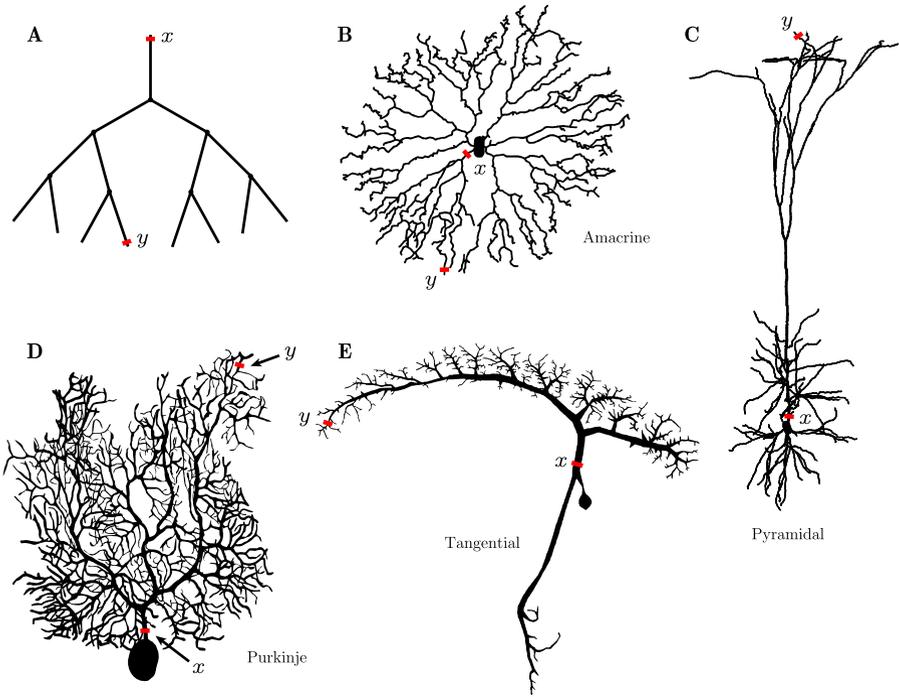


Fig. 4 Neuronal structures used in construction of the Green’s function. **A:** a binary tree, **B:** a rabbit amacrine cell [28], **C:** a rat pyramidal cell [29], **D:** a rat Purkinje cell [30], and **E:** a blowfly tangential cell [31]

using as many nodes as necessary to accurately reflect the spatial jitter and variation in radius of its path. Because radii are described at nodes, edges between two nodes of different radius taper. The sum-over-trips formalism requires constant diameter along edges, but allows discontinuous jumps in the diameters at nodes. Hence, edge diameter was defined as the average of the diameters of adjacent nodes. This allows full dendritic branches to be represented as a sequence of uniform cylinders of arbitrary length and with abrupt changes in diameters at nodes.

We used the following normalised L^1 error as a measure of convergence:

$$\varepsilon = \frac{1}{V_N} \int_0^T |G_{ij}(x, y, t) - V^*(x, y, t)| dt,$$

where T is the final simulation time, $V^*(x, y, t)$ is NEURON’s numerical solution to very high accuracy and $V_N = \int_0^T V^*(x, y, t) dt$ is the integral of the accurate NEURON solution. This convergence measure is therefore relative to the amplitude of the “real” solution, and thus errors ε are comparable between different neuronal types.

Figure 5 shows the convergence of the four-classes and of the length-priority methods as a function of the number of trips on five geometries from Fig. 4. Three sets of parameters given in Table 1 were considered for the binary tree, and the relative errors ε for each case are demonstrated in Figs. 5A–C. These plots illustrate fairly

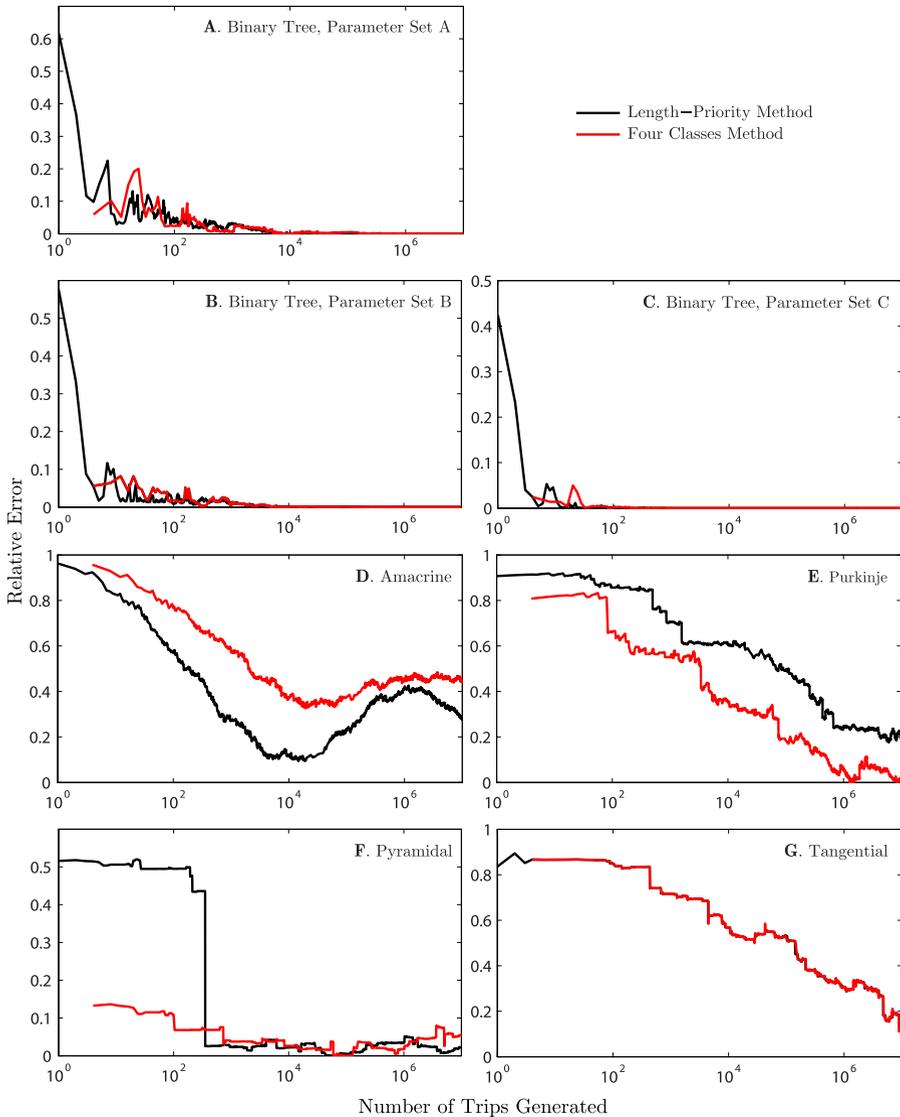


Fig. 5 Convergence of the four-classes and length-priority methods for a number of dendritic morphologies. The relative error ε of the approximation of $G_{ij}(x, y, t)$ is shown as a function of the number of trips in the sum-over-trips framework for injection at y and measurement at x on the dendritic trees in Fig. 4. Membrane parameters for real dendritic morphologies: $C = 1 \mu\text{F cm}^{-2}$, $R = 3,000 \Omega \text{ cm}^2$ and $R_a = 100 \Omega \text{ cm}$. Note that the four-classes method always begins with four trips, and each step in the algorithm adds a further trip of each class

uniform convergence, in which both methods offer similar accuracies and rates of convergence. Of the two trees with biophysically realistic parameters, the binary tree with the longer branches (parameter set C) converges faster, as is expected on structures with longer trips in each Class. This is reflected in Table 2, which shows the

Table 1 Parameter sets of the binary tree in Fig. 4A

	Parameter set A	Parameter set B	Parameter set C
Branch length L	0.3	50 μm	100 μm
Branch diameter a	0.05	1 μm	1 μm
Diffusion coefficient D	1	$2.5 \times 10^4 \mu\text{m}^2 \text{ms}^{-1}$	$2.5 \times 10^4 \mu\text{m}^2 \text{ms}^{-1}$
Membrane time constant τ	1	3.3 ms	3.3 ms
Membrane capacitance C	1	1 $\mu\text{F cm}^{-2}$	1 $\mu\text{F cm}^{-2}$

Table 2 Length-priority method on a binary tree: number of trips required for a given accuracy

Binary tree	Relative error threshold ε			
	0.1	0.05	0.01	0.001
Parameter set A	3,240	8,750	1,820,000	$>5 \times 10^7$
Parameter set B	825	2,600	129,000	$>5 \times 10^7$
Parameter set C	22	65	815	7,700

number of trips required on the binary tree to remain under a given error threshold for different parameter sets. Moreover, a binary tree with non-dimensionalised parameters (parameter set A) requires noticeably many more trips for desired accuracy in comparison to the same tree with biophysically realistic parameters.

Figures 5D–G show ε for the structures in Figs. 4B–E respectively. They demonstrate that convergence is non-trivial on complex branching structures. Figure 5D shows that the length-priority method makes consistently less error on the amacrine cell geometry, in contrast to the convergence of the Purkinje cell, shown in Fig. 5E, where the four-classes method generates less error for all numbers of trips. Both of these show strongly irregular convergence and high-amplitude oscillation in the errors ε in the amacrine cell. For both methods, the Purkinje cell shows a plateau in error for Green's functions with few trips, indicating that either these trips are of small magnitude or that their voltage traces alternate between undershooting or overshooting the correct solution between subsequent trips. This indicates that neither the length-priority or the four-classes methods are good heuristics for ordering terms in the Green's function. This is further hinted at by the oscillating property of the error, which implies that there are regions where trips that increase the error are more frequent than trips that reduce it.

The pyramidal cell's convergence shows very discontinuous behaviour (Fig. 5F), particularly in the length-priority method. The large jump in error when approximately 350 trips are included in the Green's function was found to be caused by the first and shortest Class 2 trip included thus far, with all prior trips belonging to Class 1. This behaviour is likely to arise if there exist very short branches along the shortest and most direct $x \rightarrow y$ trip, and thus many Class 1 trips are generated first, being shorter than the first Class 2 trip. Whilst one of the motivating reasons for considering a length-priority approach was to generate trips fully by length order, this heuristic makes no attempt to include the coefficient A_{trip} in its ordering. This is an example of a pathologically large change in the coefficients value for a Class 2 trip

which contributes a very significant amount to the Green's function. The four-classes approach, which enforces generation of trips of all four classes at every added excursion, does not show such a drastic drop in error. However, the error plot is still very discontinuous, and this may be a characteristic of situations as we have just described, where points x and y are placed on branches having a very different length to those on the most direct $x \rightarrow y$ trip, or when these points are placed very close to a node. Whether injection and measurement points are located on branches that are significantly longer or shorter than those along the shortest $x \rightarrow y$ trip, both the four-classes and the length-priority methods will generate trips in an "unnatural" order, subsampling the trips where current will spread the most, but oversampling in areas of the tree with very short branches. This pathological feature may not be inherently present in the real neuronal morphology, but may have been created during digital reconstruction from slice image data if, for example, a change of radius were found along the branch. Therefore, this pathology may not be representative of the neuronal geometry, but becomes a function of the reconstruction.

The tangential cell's convergence, shown in Fig. 5G, shows almost identical errors for both the four-classes and the length-priority methods, indicating that trips are generated in a similar order regardless of method. Contrary to the example with the pyramidal cell, this behaviour is likely to occur when x and y are placed on branches that are significantly shorter than those that arise on the shortest $x \rightarrow y$ path, such that the length-priority method returns trips of Class 1, 2, 3 and 4 in sequential order, as these increases in length are shorter than adding an excursion along the direct $x \rightarrow y$ trip.

Our results clearly indicate that the convergence of the realisation of the sum-over-trips framework by either the four-classes or the length-priority method strongly depends on a dendritic geometry. For real morphologies, the number of trips required quickly becomes very large to the point where guaranteeing convergence to within some small error threshold may become computationally expensive.

The convergence of the Monte-Carlo method is shown in Fig. 6 for the binary tree in Fig. 4A with parameter set A and for a larger binary tree of depth 16 with the same parameters. Boundary effects can be seen on the smaller tree, where the convergence rate is slightly faster than that of a typical Monte-Carlo integration observed here for a larger tree. It is worth noting that the x -axis on this plot shows the number of random walks generated; however, due to the number of sub-trips extracted from each random walk, the number of terms contributing to the Green's function can potentially be significantly different. The graphs show that the Monte-Carlo method is very slow to converge, although the method is much more predictable in its convergence despite the noise. As expected, therefore, the Monte-Carlo method generates trips that are more "naturally" ordered, and hence convergence is much more monotonic. Despite this improved ordering of terms in the series solution, the Monte-Carlo approach remains computationally intensive and very slow to converge with increased number of trips.

Finally, the convergence of the matrix method on a binary tree is shown in Fig. 7. This algorithm converges extremely quickly to within very small error tolerances as a function of k_{\max} , the maximum number of edges covered by trips generated. The values of the product of A_{trip} coefficients obtained by this method remain $\mathcal{O}(1)$ for all

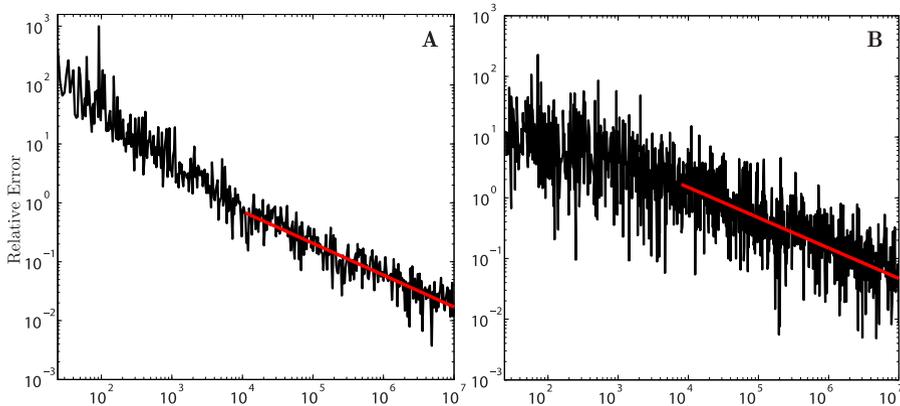


Fig. 6 Convergence of the Monte-Carlo method for a binary tree. The relative error ϵ is shown as a function of the number of random walk realisations generated, k . **A:** error generated on the binary tree in Fig. 4A with parameter set A. The red line shows a fit for $\epsilon \sim k^{-0.54}$. **B:** the convergence error on a binary tree of depth 16 (65,536 nodes). The red line demonstrates a fit for $\epsilon \sim k^{-0.5}$, the typical rate of convergence of a Monte-Carlo integration

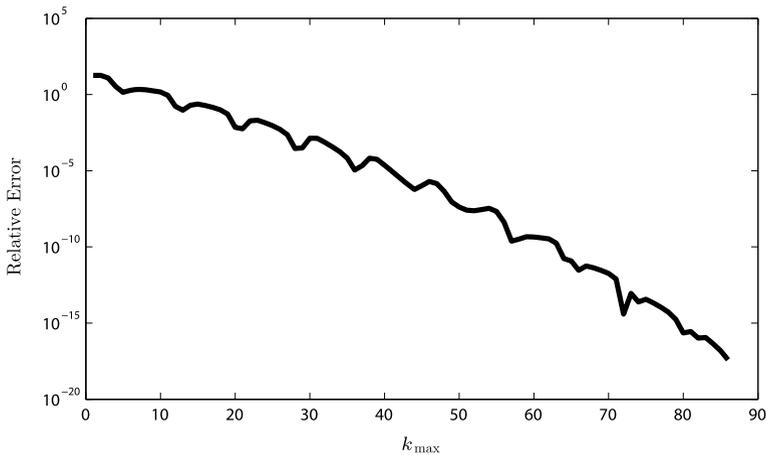


Fig. 7 Convergence of the matrix method. The relative error ϵ is shown as a function of the maximal number of edges travelled in the trips, k_{\max} , for a binary tree in Fig. 4A

k_{\max} which agrees with the proof in [25]. Because the algorithm is based on simple matrix-vector multiplication, where the matrix is $|\mathcal{E}| \times |\mathcal{E}|$ in size, the computation of the Green’s function for small trees such as the binary trees used here to within $\epsilon = 10^{-15}$ only takes a fraction of a second. On more complex trees, such as Purkinje cells, this becomes more expensive, although computing $G_{ij}(x, y, t)$ for the whole tree remains computationally preferable to the use of brute-force simulators. Using the reciprocal rule (9), this is possible in $|\mathcal{E}|$ applications of the algorithm. This compares favourably with the length-priority and the four-classes methods, which require $|\mathcal{E}|(|\mathcal{E}| + 1)/2$ applications of the algorithm, and with NEURON, which would re-

quire $|\mathcal{E}|^2$ simulations, and this would only provide solutions for a single point y on each edge. In addition, the sparseness of the matrix Q means that coefficient calculation up to k_{\max} only takes $\mathcal{O}(|\mathcal{E}|k_{\max})$ time, and so the method scales linearly with the number of branches on the tree.

4.1 Structural-Electrotonic Properties

The Green’s function $G_{ij}(x, y, t)$ constructed for a given dendritic geometry provides a measure of the transfer impedance between the input location y on branch j and the point of measurement x on branch i . Assessing whether the neuronal geometry significantly impacts this input-output relation is a step towards answering questions regarding the structure-function relationship behind the enormous natural variation in dendritic morphologies. Using the measures introduced by Zador et al. [32], the propagation delay and the log-attenuation, we analyse the transfer of the response signal in four reconstructed cells in Fig. 4. For a pair of points (x, y) along the tree, we reintroduce the propagation delay \mathcal{P}_{xy} as a measure of the impact of the tree’s electrotonic structure on the timing of signals, defined as

$$\mathcal{P}_{xy} = \hat{t}_x - \hat{t}_y.$$

Here, \hat{t}_x and \hat{t}_y are the centroids of the two corresponding voltage transients $G_x(t) = G_{ij}(x, y, t)$ and $G_y(t) = G_{jj}(y, y, t)$ respectively:

$$\hat{t}_x = \frac{\int_0^\infty tG_x(t) dt}{\int_0^\infty G_x(t) dt} \quad \text{and} \quad \hat{t}_y = \frac{\int_0^\infty tG_y(t) dt}{\int_0^\infty G_y(t) dt}.$$

This delay measure admits an additive property such that $\mathcal{P}_{xy} = \mathcal{P}_{xz} + \mathcal{P}_{zy}$ for a point z between x and y . The log-attenuation of the response signal between a pair of points (x, y) is computed as $\mathcal{L}_{xy} = \log \mathcal{A}_{xy} > 0$, where

$$\mathcal{A}_{xy} = \frac{\int_0^\infty G_y(t) dt}{\int_0^\infty G_x(t) dt} \geq 1. \tag{19}$$

It acts as a measure of the amount a transient signal’s amplitude diminishes as it travels between two points. \mathcal{L}_{xy} is also additive for a point z between x and y , that is, $\mathcal{L}_{xy} = \mathcal{L}_{xz} + \mathcal{L}_{zy}$.

Figure 8 demonstrates the propagation delay and the log-attenuation as a function of distance y away from x for the reconstructed dendritic morphologies in Figs. 4B–E. The point x was placed near the soma as shown in Fig. 4 and the position of y was moved away from x to the distal dendrites along a single path. As expected with an additive property, both the delay and the log-attenuation are linear in the distance between y and x . The curves in Fig. 8 show a noisy linear trend, which could be smoothed to better demonstrate this linearity by sampling the data from multiple points located along different branches, but at the same fixed distance away from x , in the manner of an expanding sphere of radius y , with origin at x . To compare how the response signal is transferred in four neuronal types, we plot the rate of change of the delay and log-attenuation for individual cells in Fig. 9, computed by a linear regression and imposing that the line passes through the origin. It succinctly illustrates that the input signal will spread differently in these four cells, with the tan-

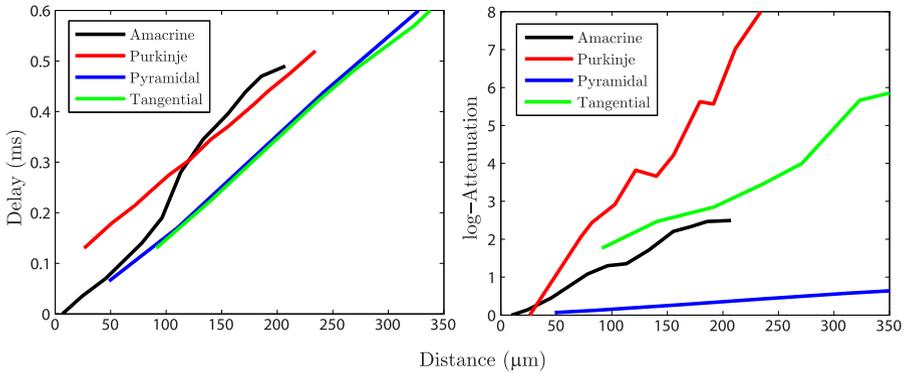
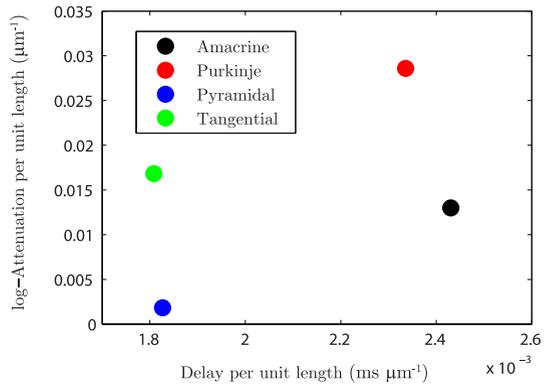


Fig. 8 Propagation delay and log-attenuation for reconstructed geometries

Fig. 9 Transfer properties of the response signal for reconstructed geometries



gential cell having a similar rate of delay with the pyramidal cell and a similar rate of log-attenuation with the amacrine cell. The signal in the Purkinje cell is shown to be attenuated most, whereas the pyramidal cell transfers it very effectively. Similar conclusions, but about the propagation of the dendritic action potential, were made by Vetter et al. [30]. The activation of voltage-gated channels is expected to be less robust in the case of strong attenuation of the passive spread of voltage which might explain the results in [30].

5 Discussion

In this paper, we introduced a number of efficient algorithms for the computational realisation of the sum-over-trips framework and assessed their convergence. We started with some modifications of the four-classes algorithm of Cao and Abbott [21] to avoid constructing duplicate trips. An unambiguous context-free grammar was derived, which is able to generate all trips uniquely and in monotonic order of length. We then developed the length-priority method, in which trips are constructed purely in length order rather than in classes. Both methods were found to demonstrate very

nonuniform convergence which was highly-dependent on the dendritic morphology, as well as the biophysical properties of the cell membrane. Oscillations of the convergence error make it difficult to predict the number of trips required for constructing the Green's function on a particular geometry. Dendritic structures with longer branches of uniform diameters will converge faster, that is, for a smaller number of trips in the series solution. Instead of sampling the trips in some well-defined order, we also derived a stochastic method of sampling the trips based on a Monte-Carlo approach. Finally, we proposed an extremely efficient matrix method which computes the trip coefficients, A_{trip} , for trees where all branches are integer-multiples in length to some base length, Δx .

Although we considered dendrites to be passive in this study, the proposed algorithms can be easily generalised to support quasi-active (resonant) dendrites with a calculation of the Green's function in the Laplace domain [24]. Moreover, it is straightforward to include an isopotential soma in the Laplace-domain series solution. A soma can be considered as a special node with the factors $p_k(\omega)$ on the branches connected to the soma defined as $p_k(\omega) = r_k^{-1} \sqrt{(\omega + \tau^{-1})D_k^{-1}} / (\widehat{C}\omega + \widehat{R}^{-1} + \sum_m r_m^{-1} \sqrt{(\omega + \tau^{-1})D_m^{-1}})$, where \widehat{C} and \widehat{R} are the capacitance and the resistance of the somatic membrane and ω is the Laplace transform's frequency variable. Similar factors can also be found for when the leaky-end boundary condition, referred to as *natural termination* by Tuckwell [33], is imposed at the terminals. A knowledge of the Green's function for a given dendritic structure allows one to efficiently find the sub-threshold voltage response along the entire tree for any number of various inputs, either analytically or via a computation of the convolution integral. This obviates the need for the brute-force numerical simulations of an underlining set of PDEs. Such simulations may be computationally expensive, particularly since they have to be re-initiated each time a new stimulus is introduced. In the case of supra-threshold inputs, which can activate voltage-gated channels known to be present in dendrites of many neurons, the Spike-Diffuse-Spike (SDS) type model [34, 35] can be utilised for analysing the propagations of dendritic action potentials. Although the voltage-gated channels in the SDS framework are modelled by piecewise linear instead of nonlinear dynamics, it has been shown that the speed of a wave propagation in the SDS model is in excellent agreement with a more biophysically realistic nonlinear model [36]. However, an analytically tractable SDS model combined with a fast algorithm for constructing the Green's function on real geometries provides a computationally efficient framework for studying wave scattering in dendrites.

Although networks of spatially-extended neural cells can be numerically simulated, there are currently few mathematical studies of such networks. A natural extension might be to consider a network of branched neurons coupled by gap-junctions. The sum-over-trips formalism can then be generalised to support a presence of new boundary conditions. Recent results of Harris and Timofeeva [37] can be applied to the case of tip-to-tip coupling of the dendritic branches. The proposed algorithms can then be modified by including additional sum-over-trips rules. It is worth mentioning that the computational schemes presented in this paper are able to handle cyclic graphs, which may form as a result of gap-junction coupling across several neurons. While the matrix method is expected to be the most efficient for a network

of symmetric or regular structures, realistic reconstructions of discretised trees remain within computational reach. For example, given a sparse random matrix, of correct density and of size 20,000, equivalent to a dendritic tree with 10,000 edges, the calculation of $G_{ij}(x, y, t)$ for all j up to $k_{\max} = 10^5$ only takes fifteen seconds on a desktop computer. For comparison, the Purkinje cell reconstruction in Fig. 4D has just under 5,000 branches. For the case of very large, complex irregular structures it might be possible to employ a recently developed technique of reducing the complexity of large dendrites [38] before applying the sum-over-trips methodology.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

QC was directly involved in developing and implementing the algorithms, carried out all analysis, and drafted the manuscript. SRD participated in developing and implementing the algorithms. SPBC developed the Monte-Carlo algorithm. YT conceived and guided the study. All authors contributed improvements to the final manuscript, which they have read and approved.

Acknowledgements QC and SPCB would like to acknowledge the Complexity Science Doctoral Training Centre at the University of Warwick along with the funding provided by the EPSRC (EP/E501311). SRD acknowledges funding from the EPSRC and the MRC through the Doctoral Training Centre in Neuroinformatics at the University of Edinburgh. YT would like to acknowledge the support provided by the BBSRC (BB/H011900) and the RCUK.

Appendix: Mathematical Convergence of the Sum-over-trips Series Solution

Here, we consider an identical diffusion coefficient D for all branches, although it is possible to generalise this proof to support different diffusion coefficients. Fixing t throughout, we let

$$G_{ij}(x, y) = \sum_{\text{trips}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}) = \sum_{k=0}^{\infty} \sum_{\substack{\text{paths with} \\ k \text{ nodes}}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}),$$

where

$$G_{\infty}(L_{\text{trip}}) = \frac{1}{\sqrt{4\pi t D}} e^{-L_{\text{trip}}^2/(4Dt)} e^{-t/\tau}. \tag{20}$$

A_{trip} is a product of k factors $2p \in (0, 2)$ and $2p - 1 \in (-1, 1)$, where k is the number of nodes visited by the trip, for any branch diameters. Then, for a trip touching k nodes,

$$|A_{\text{trip}}| \leq 2^k. \tag{21}$$

There exists a constant $B > 0$ such that every trip touching k nodes satisfies

$$L_{\text{trip}} \geq Bk. \tag{22}$$

This makes B the coefficient of the lower bound on trip length in terms of the number of nodes in a trip. Intuitively, Bk equals the minimum distance between any two nodes where, for this purpose, we count x and y as nodes.

Let F_k be the number of trips with k nodes. Since each node has degree $d \leq 3$, then

$$F_k \leq 3^k. \tag{23}$$

We introduce

$$\Gamma_k = \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}).$$

Then

$$\begin{aligned} |\Gamma_k| &= \left| \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}) \right| \\ &\leq \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} |A_{\text{trip}} G_{\infty}(L_{\text{trip}})| \\ &= \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} |A_{\text{trip}}| |G_{\infty}(L_{\text{trip}})|. \end{aligned} \tag{24}$$

For simplicity, we rewrite the Green’s function (20) as $G_{\infty}(L_{\text{trip}}) = Ce^{-EL_{\text{trip}}^2}$, where

$$C = \frac{e^{-t/\tau}}{\sqrt{4\pi Dt}} \quad \text{and} \quad E = \frac{1}{4Dt}.$$

Then using (21)–(24) we get

$$\begin{aligned} |\Gamma_k| &\leq \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} 2^k Ce^{-EL_{\text{trip}}^2} \\ &\leq \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} 2^k Ce^{-EB^2k^2} \\ &\leq F_k 2^k Ce^{-EB^2k^2} \\ &\leq 3^k 2^k Ce^{-EB^2k^2} \\ &= 6^k Ce^{-EB^2k^2} \\ &= Ce^{-k(EB^2k - \ln(6))}. \end{aligned} \tag{25}$$

We define $N = \lfloor \ln(6)/(EB^2) \rfloor$ such that $EB^2k - \ln(6) > 0$ for $\forall k > N$. Then

$$\begin{aligned} \left| \sum_{k=0}^{\infty} \Gamma_k \right| &\leq \sum_{k=0}^{\infty} |\Gamma_k| \\ &\leq \sum_{k=0}^{\infty} C e^{-k(EB^2k - \ln(6))} \\ &= \sum_{k=0}^N C e^{-k(EB^2k - \ln(6))} + \sum_{k=N+1}^{\infty} C e^{-k(EB^2k - \ln(6))}. \end{aligned} \tag{26}$$

The first sum in (26) is a finite sum of finite terms, and is hence finite. We will now show that the second sum is also finite using d’Alembert’s ratio criterion for convergent series. The ratio ρ_k of the consecutive terms in the series, k and $k + 1$, is

$$\begin{aligned} \rho_k &= \left| \frac{C e^{-(k+1)(EB^2(k+1) - \ln(6))}}{C e^{-k(EB^2k - \ln(6))}} \right| \\ &= e^{-(EB^2(2k+1) - \ln(6))}. \end{aligned}$$

Letting $k \rightarrow \infty$, we obtain

$$\begin{aligned} \rho_{\infty} &= \lim_{k \rightarrow \infty} \rho_k \\ &= \lim_{k \rightarrow \infty} e^{-(EB^2(2k+1) - \ln(6))} \\ &= 0. \end{aligned}$$

With $\rho_{\infty} < 1$, the second sum in (26) converges absolutely for all constants $B, C, E > 0$. Therefore, the series in (26) is absolutely convergent for sufficiently-high k .

If we define

$$G_{ij}^M(x, y, t) = \sum_{k=0}^M \sum_{\substack{\text{trips with} \\ k \text{ nodes}}} A_{\text{trip}} G_{\infty}(L_{\text{trip}}, t),$$

then

$$\left| G_{ij} - G_{ij}^M \right| \leq \sum_{k=M+1}^{\infty} C e^{-k(EB^2k - \ln(6))}$$

and the path integral converges faster than e^{-k} in the worst case, with the number of nodes k visited by the trips.

References

- Cajal R: *Histology of the Nervous System of Man and Vertebrates*. New York: Oxford University Press; 1995 (trans. N Swanson and LW Swanson, first published 1899).
- Ulfhake B, Kellerth JO: **A quantitative light microscopic study of the dendrites of cat spinal alpha-motoneurons after intracellular staining with horseradish peroxidase**. *J Comp Neurol* 1981, **202**:571-583.
- Rall W: **Core conductor theory and cable properties of neurons**. In *Handbook of Physiology—The Nervous System (I)*; 1977:39-97.
- Segev I, Rinzel J, Shepherd GM (Eds): *The Theoretical Foundation of Dendritic Function: Selected Papers of Wilfrid Rall with Commentaries*. Cambridge: MIT Press; 1995.
- Spruston N, Stuart G, Häusser M: **Dendritic integration**. In *Dendrites*. Edited by Spruston N, Stuart G, Häusser M. New York: Oxford University Press; 2008.
- van Ooyen A, Duijnhouwer J, Remme MWH, van Pelt J: **The effect of dendritic topology on firing patterns in model neurons**. *Netw Comput Neural Syst* 2002, **13**(3):311-325.
- Spruston N, Stuart G, Häusser M (Eds): *Dendrites*. New York: Oxford University Press; 2008.
- Johnston D, Narayanan R: **Active dendrites: colorful wings of the mysterious butterflies**. *Trends Neurosci* 2008, **31**(6):309-316.
- London M, Häusser M: **Dendritic computation**. *Annu Rev Neurosci* 2005, **28**:503-532.
- Rall W: **Theoretical significance of dendritic trees for neuronal input-output relations**. In *Neural Theory and Modeling*. Edited by Reiss RF. Stanford: Stanford University Press; 1964:73-97.
- Segev I, Fleshmann IJ, Burke RE: **Compartmental models of complex neurons**. In *Methods in Neuronal Modeling*. Cambridge: MIT Press; 1989.
- Rall W: **Theory of physiological properties of dendrites**. *Ann NY Acad Sci* 1962, **96**(2):1071-1092.
- Koch C, Poggio T: **A simple algorithm for solving the cable equation in dendritic trees of arbitrary geometry**. *J Neurosci Methods* 1985, **12**(4):303-315.
- Butz EG, Cowan JD: **Transient potentials in dendritic systems of arbitrary geometry**. *Biophys J* 1974, **14**(9):661-689.
- Whitehead RR, Rosenberg JR: **On trees as equivalent cables**. *Proc R Soc Lond B, Biol Sci* 1993, **252**(1334):103-108.
- Lindsay K: **Analytical and numerical construction of equivalent cables**. *Math Biosci* 2003, **184**(2):137-164.
- Evans JD, Kember GC, Major G: **Techniques for obtaining analytical solutions to the multicylinder somatic shunt cable model for passive neurones**. *Biophys J* 1992, **63**:350-365.
- Major G, Evans JD, Jack JJB: **Solutions for transients in arbitrary branching cables: I. Voltage recording with a somatic shunt**. *Biophys J* 1993, **65**:423-449.
- Evans JD, Major G: **Techniques for the application of the analytical solution to the multicylinder somatic shunt cable model for passive neurones**. *Math Biosci* 1995, **125**:1-50.
- Abbott L, Farhi E, Gutmann S: **The path integral for dendritic trees**. *Biol Cybern* 1991, **66**:49-60.
- Cao BJ, Abbott LF: **A new computational method for cable theory problems**. *Biophys J* 1993, **64**(2):303-313.
- Rapp M, Segev I, Yarom Y: **Physiology, morphology and detailed passive models of guinea-pig cerebellar Purkinje cells**. *J Physiol* 1994, **474**:101-118.
- Eppstein D: **Finding the k shortest paths**. *SIAM J Comput* 1999, **28**(2):652-673.
- Coomes S, Timofeeva Y, Svensson CM, Lord GJ, Josić K, Cox SJ, Colbert CM: **Branching dendrites with resonant membrane: a “sum-over-trips” approach**. *Biol Cybern* 2007, **97**(2):137-149.
- Abbott LF: **Simple diagrammatic rules for solving dendritic cable problems**. *Physica A* 1992, **185**(1-4):343-356.
- Carnevale N, Hines M: *The NEURON Book*. Cambridge: Cambridge University Press; 2006.
- Ascoli GA, Donohue DE, Halavi M: **NeuroMorpho.Org: a central resource for neuronal morphologies**. *J Neurosci* 2007, **27**(35):9247-9251.
- Bloomfield A, Miller F: **A functional organization of ON and OFF pathways in the rabbit retina**. *J Neurosci* 1986, **6**:1-13.
- Radman T, Ramos RL, Brumberg JC, Bikson M: **Role of cortical cell type and morphology in sub- and suprathreshold uniform electric field stimulation**. *Brain Stimul* 2009, **2**(4):215-228.
- Vetter P, Roth A, Häusser M: **Propagation of action potentials in dendrites depends on dendritic morphology**. *J Neurophysiol* 2001, **85**:926-937.
- Cuntz H, Forstner F, Haag J, Borst A: **The morphological identity of insect dendrites**. *PLoS Comput Biol* 2008, **4**:e1000251.

32. Zador AM, Agmon-Snir H, Segev I: **The morphoelectrotonic transform: a graphical approach to dendritic function.** *J Neurosci* 1995, **15**(3):1669-1682.
33. Tuckwell HC: *Introduction to Theoretical Neurobiology: Volume 1. Linear Cable Theory and Dendritic Structure.* Cambridge: Cambridge University Press; 1988.
34. Coombes S, Bressloff PC: **Saltatory waves in the spike-diffuse-spike model of active dendritic spines.** *Phys Rev Lett* 2003, **91**:028102.
35. Timofeeva Y: **Travelling waves in a model of quasi-active dendrites with active spines.** *Physica D, Nonlinear Phenom* 2010, **239**(9):494-503.
36. Timofeeva Y, Lord GJ, Coombes S: **Spatio-temporal filtering properties of a dendritic cable with active spines: a modeling study in the spike-diffuse-spike framework.** *J Comput Neurosci* 2006, **21**(3):293-306.
37. Harris J, Timofeeva Y: **Intercellular calcium waves in the fire-diffuse-fire framework: Green's function for gap-junctional coupling.** *Phys Rev E* 2010, **82**:051910.
38. Yan B, Li P: **Reduced order modeling of passive and quasi-active dendrites for nervous system simulation.** *J Comput Neurosci* 2011, **31**:247-271.