



ELSEVIER

Computer Physics Communications 115 (1998) 45–68

Computer Physics  
Communications

## Solutions of Poisson's equation in channel-like geometries

Matthew Hoyles<sup>a,b</sup>, Serdar Kuyucak<sup>a</sup>, Shin-Ho Chung<sup>b,1</sup>

<sup>a</sup> *Department of Theoretical Physics, Research School of Physical Sciences, Australian National University, Canberra, ACT 0200, Australia*

<sup>b</sup> *Protein Dynamics Unit, Department of Chemistry, Australian National University, Canberra, ACT 0200, Australia*

Received 2 April 1998

### Abstract

Electric forces play a key role in the conductance of ions in biological channels. Therefore, their correct treatment is very important in making physical models of ion channels. Here, we present FORTRAN 90 codes for solution of Poisson's equation satisfying the Dirichlet boundary conditions in realistic channel geometries that can be used in studies of ion channels. For a general channel shape, we discuss a numerical solution of Poisson's equation based on an iterative technique. We also provide an analytical solution of Poisson's equation in toroidal coordinates and its numerical implementation. A torus shaped channel is closer to reality than a cylindrical one, hence it could serve as a useful test model. © 1998 Elsevier Science B.V.

PACS: 87.22.Bt; 87.22.Fy

Keywords: Poisson's equation; Ion channels

### PROGRAM SUMMARY

*Title of program:* bics

*Catalogue identifier:* ADIT

*Program Summary URL:*

<http://www.cpc.cs.qub.ac.uk/cpc/summaries/ADIT>

*Program obtainable from:* CPC Program Library, Queen's University of Belfast, N. Ireland

*Computers:* Fujitsu VPP300

*Operating systems or monitors under which the program has been tested:* Fujitsu UXP/V (System V UNIX)

*Programming language used:* FORTRAN 90

*Memory required to execute with typical data:* 24 Mb for the analytical method, 720 Mb for the iterative method

*No. of bits in a word:* The program uses single and double precision IEEE floating point numbers, and 32 bit signed integers

*Has the code been vectorised?:* Yes

*No. of bytes in distributed program, including test data, etc.:* 548269

*Distribution format:* uuencoded compressed tar file

*Keywords:* Poisson's equation, ion channels

<sup>1</sup> E-mail: shin-ho.chung@anu.edu.au

*Nature of physical problem*

The program calculates the potential energy of and force on an ion in the vicinity of a model ion channel. It can include the effects of an externally applied field and fixed charges other than the ion. It can also calculate the electric field and potential in the absence of an ion.

*Method of solution*

We treat ions and fixed charges as point charges, and treat water and channel protein as regions of constant dielectric strength. The electric potential at a point in the pore can be found by solving Poisson's equation with appropriate boundary conditions.

The program uses one of two methods: The iterative method distributes a grid of surface charges across an arbitrary cylindrically symmetric boundary, and modifies them iteratively until it finds a self-consistent solution. The analytical method computes a finite number of terms in an infinite series which is a solution to Poisson's equation for a toroidal boundary.

*Unusual features of the program*

The program takes advantage of the new features of FORTRAN 90. It is split into modules rather than subroutines, and uses the concept of abstract data types to regulate data structures.

## LONG WRITE-UP

### 1. Introduction

The flow of ions through membrane channels is a fundamental biological process that regulates most bodily functions at the cell level [1]. Despite the significance of the problem, the physical processes involved in transport of ions across a biological channel have not been well understood yet. A key component in the modeling of channels is the electric force acting on an ion in the channel, which is ultimately responsible for the channel's conductance. Channels are made of protein and most have vestibular openings similar to an hourglass shape. Because proteins have a low dielectric constant ( $\epsilon_p = 2$ ) compared to water ( $\epsilon_w = 80$ ) in which ions move, the channel boundary plays a significant role in determining the electric forces acting on ions. The basic problem involves solving Poisson's equation for a single ion in an appropriate channel geometry. The effect of an applied electric field usually poses a simpler problem and can be taken into account using the superposition principle. Similarly, an electric potential for the many-ions case can be obtained by superposing the potentials of individual ions.

There are many coordinate systems in which Poisson's equation separates, enabling analytical solutions. However, none of them can be used to simulate a boundary in the hourglass or catenary shape, which approximate biological channels best. Thus, a numerical solution of the problem is required to simulate realistic channel shapes. In the next section, we describe an iterative technique and its numerical implementation that can be used to solve Poisson's equation for an arbitrary, closed boundary [2].

Of all the coordinate systems in which Poisson's equation separates, the toroidal coordinates come closest to forming a realistic channel in a torus shape [3]. While this is not a very close approximation to an actual channel (the curvature is opposite), there are many advantages to having analytical solutions. For example, in a Brownian dynamics simulation [4], the electric potential needs to be calculated many times, and such a simulation is simply not feasible if one has to calculate the potential numerically at every time step. With such applications in mind, we present a solution of Poisson's equation in toroidal coordinates. The solution is rather complicated requiring careful programming, therefore it is included here as a second program.

### 2. Numerical solution

In this section, we discuss a numerical solution of Poisson's equation using an iterative technique.

### 2.1. Poisson's equation in dielectric media

The electric potential  $\varphi$  in dielectric media satisfies Poisson's equation,

$$\nabla^2 \varphi = -\frac{\rho}{\epsilon_0 \epsilon}, \quad (1)$$

where  $\rho$  and  $\epsilon$  refer to the charge density and the relative dielectric constant, and  $\epsilon_0$  is the permittivity of free space. We wish to solve Eq. (1) for general channel geometries, without imposing any symmetries either on the channel shape or the position of ions. A typical boundary representative of ion channels is shown in Fig. 6. Denoting outside the boundary (i.e. channel) by subscript 1 and inside by 2, the potentials should satisfy the usual continuity conditions at the boundary,

$$\varphi_1 = \varphi_2, \quad \epsilon_1 \nabla \varphi_1 \cdot \hat{\mathbf{n}} = \epsilon_2 \nabla \varphi_2 \cdot \hat{\mathbf{n}}, \quad (2)$$

where  $\hat{\mathbf{n}}$  is the unit normal to the surface. Eq. (2) can be expressed in terms of the electric fields  $\mathbf{E}$  as

$$\epsilon_1 \mathbf{E}_1 \cdot \hat{\mathbf{n}} = \epsilon_2 \mathbf{E}_2 \cdot \hat{\mathbf{n}}. \quad (3)$$

In general, this problem cannot be solved using analytical methods and one has to resort to iterative numerical techniques. Because the solution of Poisson's equation is unique for a closed boundary, convergence of results ensures that the solution found is the correct one.

Following Ref. [5], we first reduce the three-dimensional boundary value problem into an equivalent two-dimensional problem by replacing this system with an equivalent system of charges in a vacuum which produces the same electrical potential throughout space. The charge densities  $\rho_i$  in the two regions are replaced by reduced charge densities  $\rho_i/\epsilon_i$ ,  $i = 1, 2$ . The discontinuity in the electric fields across the boundary can be represented by polarization charge density,  $\sigma$ , induced at the surface. Using an infinitesimal Gaussian pill-box across a surface area  $\Delta S$  at position  $\mathbf{r}$ , the two are related by

$$(\mathbf{E}_1 - \mathbf{E}_2) \cdot \hat{\mathbf{n}} = \frac{\sigma}{\epsilon_0}. \quad (4)$$

Thus the electric fields can be written as

$$\mathbf{E}_1 = \mathbf{E}_{\text{ex}} + \frac{\sigma}{2\epsilon_0} \hat{\mathbf{n}}, \quad \mathbf{E}_2 = \mathbf{E}_{\text{ex}} - \frac{\sigma}{2\epsilon_0} \hat{\mathbf{n}}, \quad (5)$$

where  $\mathbf{E}_{\text{ex}}$  is the part due to all the charges except those in  $\Delta S$ . Eliminating  $\mathbf{E}_2$  from Eqs. (3), (4) and substituting  $\mathbf{E}_1$  from Eq. (5), we obtain a relationship between the surface charge density and the external field,

$$\sigma = P \mathbf{E}_{\text{ex}} \cdot \hat{\mathbf{n}}, \quad (6)$$

where

$$P = 2\epsilon_0 \frac{\epsilon_2 - \epsilon_1}{\epsilon_2 + \epsilon_1} \quad (7)$$

is the polarizability of the boundary. In Eq. (6),  $\mathbf{E}_{\text{ex}} \cdot \hat{\mathbf{n}}$  is determined from the normal derivative of the external potential,

$$\varphi_{\text{ex}}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \left[ \sum_i \int \frac{\rho_i(\mathbf{r}')}{\epsilon_i |\mathbf{r} - \mathbf{r}'|} dV' + \int_{\mathbf{r}' \neq \mathbf{r}} \frac{\sigma(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dS' \right]. \quad (8)$$

Starting with an initial surface charge density of  $\sigma_0(\mathbf{r}') = 0$ , one can make an initial estimate of the potential at the boundary from Eq. (8). This potential is then fed into Eq. (6) and a new density  $\sigma_1(\mathbf{r}')$  is obtained.

Eqs. (6) and (8) are iterated until the results converge, that is, the difference between  $\sigma_{n-1}$  and  $\sigma_n$  is sufficiently small.

The potential energy is calculated from the expression

$$U(\mathbf{r}) = \frac{1}{2} \int \varphi(\mathbf{r}) \rho_{\text{free}}(\mathbf{r}) dV - U_{\text{self}}, \quad (9)$$

by numerical integration. Here  $U_{\text{self}}$  is the Born self-energy of the ion and  $\rho_{\text{free}}$  refers to the charge density excluding the polarization charges induced on the boundary.

## 2.2. Iterative method of determining surface charges

This method is implemented as follows. The boundary is divided into small sectors of area  $\Delta S_i$ , each represented by a point charge  $q_i$  at its center. First the charge density at each point is found using Eq. (6), based on the field from fixed charges. Then each point is assigned a charge equal to its charge density times the area it represents,  $q_i = \sigma_i \Delta S_i$ . This process is repeated, using both the fixed charges and the current estimate of the boundary charges, until the boundary charges converge. An applied electric field can be included simply by adding it to the field from the fixed charges.

Because the computation time grows with the square of the number of sectors, it is important to optimize the choice of sector area. Two important considerations in this regard are, first, distance of the surface area to the external charges and, second, curvature of the area. At large distances from the external charges, the induced charge and the solid angle it subtends are small, hence one can use relatively larger areas for such sectors without introducing too much error. In contrast, because the relationship (4) is strictly valid only for a flat surface, one needs to use relatively smaller areas in places where the curvature is high. The method we used for the correction of errors introduced by the curvature is detailed in the following section. In addition, the results of control runs indicate that each sector should have approximately equal vertical and horizontal spacing, and that although the spacing can be varied between different regions of the boundary, such variation must be done smoothly.

For convergence, we test the quantity

$$\delta_i = \left| \frac{q_i^{(n)} - q_i^{(n-1)}}{q_{\text{max}}^{(n)}} \right|, \quad (10)$$

where  $q_{\text{max}}^{(n)}$  is the largest charge in the  $n$ th iteration. The calculation is stopped when  $\delta_i < 0.0001$  for all surface charges. The condition (10) is preferred over the usual one with  $q_i^{(n)}$  in the denominator because it requires fewer iterations without loss in accuracy. The reason is that induced charges at large distances are very small, making them sensitive to small changes in other charges. Thus they take a long time to converge to the same level of accuracy as the larger charges. Yet the effect of these small charges on the calculated potentials are negligible. The computations are carried out using a supercomputer (Fujitsu VPP 300) with a vector processor, which is well suited to this type of algorithm. The iterative method outlined above allows the electric potentials inside and outside of any arbitrarily-shaped vestibule to be computed.

## 2.3. Curvature compensation

Because of the simplifying assumption that sectors are flat, and the induced charge on each sector is affected by the charges on all other sectors but not by their own charge, our method produces small but systematic errors when used on curved surfaces. It overestimates the potential near convex surfaces, and underestimates that near concave surfaces. These errors can be reduced by spacing the surface points more closely, but there is a practical limit to the number of points that can be used, imposed by the available computer time and memory.

We use a method of compensating for curved sectors by incorporating an estimate of self-interaction into the polarizability of each sector. We assume that the charge density  $\sigma$  is constant across the sector, and that the electric field  $\mathbf{E}$  and normal  $\hat{\mathbf{n}}$  at the center of the sector are representative of the whole sector. Including the self-interaction, Eq. (6) can be written as

$$\begin{aligned}\sigma &= P(\mathbf{E}_{\text{ex}} + \mathbf{E}_{\text{self}}) \cdot \hat{\mathbf{n}} \\ &= \sigma_{\text{ex}} + \sigma_{\text{self}},\end{aligned}\quad (11)$$

where  $\mathbf{E}_{\text{ex}}$  is the external electric field due to other sectors and fixed charges, and  $\mathbf{E}_{\text{self}}$  arises from the self-interaction of other points within the sector. The charge densities  $\sigma_{\text{ex}}$  and  $\sigma_{\text{self}}$  are associated with the external field and self-interaction, respectively. Since we assume that the charge density is constant across the sector,  $\sigma_{\text{self}}$  is directly proportional to  $\sigma$ , that is,

$$\sigma_{\text{self}} = Q\sigma, \quad (12)$$

where the constant of proportionality  $Q$  depends only on the shape and size of the sector and the polarizability of the boundary, but not on the external field. Using this relation in Eq. (11), we obtain for the corrected charge density,

$$\begin{aligned}\sigma &= \sigma_{\text{ex}} + Q\sigma \\ &= \frac{1}{1-Q}\sigma_{\text{ex}} \\ &= \frac{P}{1-Q}\mathbf{E}_{\text{ex}} \cdot \hat{\mathbf{n}}.\end{aligned}\quad (13)$$

By precalculating  $Q$  and using Eq. (13) in place of (6) in the iterations, we can compensate for curved sectors without modifying the iterative algorithm.

Towards this end, we first express  $Q$  in terms of  $\mathbf{E}_{\text{self}}$  using Eqs. (11) and (12),

$$Q = \frac{P\mathbf{E}_{\text{self}} \cdot \hat{\mathbf{n}}}{\sigma}. \quad (14)$$

With the assumption of constant charge density  $\sigma$  in the sector,  $\mathbf{E}_{\text{self}}$  depends only on the geometry of the sector and is given by the surface integral

$$\mathbf{E}_{\text{self}} = \frac{\sigma}{4\pi\epsilon_0} \int_S \frac{\delta\mathbf{R}}{|\delta\mathbf{R}|^3} dA, \quad (15)$$

where  $\delta\mathbf{R}$  is a vector from the center of the sector to an arbitrary point on the sector, and the surface area covers the whole sector. Substituting Eq. (15) in (14), we obtain the desired expression for  $Q$ ,

$$Q = \frac{P}{4\pi\epsilon_0} \int_S \frac{\delta\mathbf{R} \cdot \hat{\mathbf{n}}}{|\delta\mathbf{R}|^3} dA. \quad (16)$$

To make further progress, we describe the sector as a parametric surface in variables  $(t, u)$  with the center at  $\mathbf{R}_0 = \mathbf{R}(t_0, u_0)$  and the distance from the center given by  $\delta\mathbf{R} = \mathbf{R}(t, u) - \mathbf{R}_0$ . Since the surface dimensions are small, we can Taylor expand  $\delta\mathbf{R}$  around  $\mathbf{R}_0$  as

$$\delta\mathbf{R} = \mathbf{R}_{t,0}\delta t + \mathbf{R}_{u,0}\delta u + \frac{1}{2}\mathbf{R}_{tt,0}\delta t^2 + \frac{1}{2}\mathbf{R}_{uu,0}\delta u^2 + \mathbf{R}_{tu,0}\delta t\delta u + \dots, \quad (17)$$

where  $\delta t = t - t_0$ ,  $\delta u = u - u_0$  and the subscripts denote derivatives of  $\mathbf{R}(t, u)$  at  $(t_0, u_0)$ , that is,

$$\mathbf{R}_{t,0} = \left. \frac{\partial \mathbf{R}}{\partial t} \right|_{t_0, u_0}, \quad \mathbf{R}_{tt,0} = \left. \frac{\partial^2 \mathbf{R}}{\partial t^2} \right|_{t_0, u_0}, \quad \mathbf{R}_{tu,0} = \left. \frac{\partial^2 \mathbf{R}}{\partial t \partial u} \right|_{t_0, u_0}, \quad \text{etc.} \quad (18)$$

Noting that  $\hat{\mathbf{n}} \propto \mathbf{R}_{t,0} \times \mathbf{R}_{u,0}$ , we obtain for the length and projection of  $\delta \mathbf{R}$  to lowest order,

$$\begin{aligned} |\delta \mathbf{R}|^2 &= |\mathbf{R}_{t,0}|^2 \delta t^2 + |\mathbf{R}_{u,0}|^2 \delta u^2 + \mathbf{R}_{t,0} \cdot \mathbf{R}_{u,0} \delta t \delta u, \\ \delta \mathbf{R} \cdot \hat{\mathbf{n}} &= \frac{1}{2} \mathbf{R}_{tt,0} \cdot \hat{\mathbf{n}} \delta t^2 + \frac{1}{2} \mathbf{R}_{uu,0} \cdot \hat{\mathbf{n}} \delta u^2 + \mathbf{R}_{tu,0} \cdot \hat{\mathbf{n}} \delta t \delta u. \end{aligned} \quad (19)$$

There are no first order terms in  $\delta \mathbf{R} \cdot \hat{\mathbf{n}}$ , indicating absence of self-interaction when the sector is flat. The second order terms take into account deviation from a flat surface with constant curvatures in the  $t$  and  $u$  directions.

Our boundaries are all cylindrically symmetric, so we use cylindrical coordinates  $r(t)$ ,  $z(t)$ , and  $\theta$  to parameterize the surface. Then the sector becomes part of a generalized cylinder,

$$\mathbf{R}(t, \theta) = r(t) \cos \theta \mathbf{i} + r(t) \sin \theta \mathbf{j} + z(t) \mathbf{k}. \quad (20)$$

Using a cylindrically symmetric dielectric boundary does not restrict the whole system to cylindrical symmetry; permanent charges such as ions can be arranged arbitrarily in three-dimensional space. This approach can also be used for boundaries that are not cylindrically symmetric, although the formulas will be different and more complex. The partial derivatives of  $\mathbf{R}(t, \theta)$  defined in Eq. (18) are given by

$$\begin{aligned} \mathbf{R}_{t,0} &= r'_0 \cos \theta_0 \mathbf{i} + r'_0 \sin \theta_0 \mathbf{j} + z'_0 \mathbf{k}, \\ \mathbf{R}_{\theta,0} &= -r_0 \sin \theta_0 \mathbf{i} + r_0 \cos \theta_0 \mathbf{j}, \\ \mathbf{R}_{tt,0} &= r''_0 \cos \theta_0 \mathbf{i} + r''_0 \sin \theta_0 \mathbf{j} + z''_0 \mathbf{k}, \\ \mathbf{R}_{\theta\theta,0} &= -r_0 \cos \theta_0 \mathbf{i} - r_0 \sin \theta_0 \mathbf{j}, \\ \mathbf{R}_{t\theta,0} &= -r'_0 \sin \theta_0 \mathbf{i} + r'_0 \cos \theta_0 \mathbf{j}, \end{aligned} \quad (21)$$

where  $r_0 = r(t_0)$ ,  $z_0 = z(t_0)$ , and primes indicate derivative with respect to  $t$ . The normal  $\hat{\mathbf{n}}$  at  $\mathbf{R}_0$  can be found by taking the cross product of the unit tangents,

$$\begin{aligned} \hat{\mathbf{n}} &= \frac{\mathbf{R}_{t,0}}{|\mathbf{R}_{t,0}|} \times \frac{\mathbf{R}_{\theta,0}}{|\mathbf{R}_{\theta,0}|} \\ &= \frac{1}{\gamma_0} (r'_0 \cos \theta_0 \mathbf{i} + r'_0 \sin \theta_0 \mathbf{j} + z'_0 \mathbf{k}) \times (-\sin \theta_0 \mathbf{i} + \cos \theta_0 \mathbf{j}) \\ &= \frac{1}{\gamma_0} (-z'_0 \cos \theta_0 \mathbf{i} - z'_0 \sin \theta_0 \mathbf{j} + r'_0 \mathbf{k}), \end{aligned} \quad (22)$$

where

$$\gamma_0 = \sqrt{r_0'^2 + z_0'^2}. \quad (23)$$

Substituting Eqs. (21) and (22) into (19) gives

$$\begin{aligned} |\delta \mathbf{R}|^2 &= \gamma_0^2 \delta t^2 + r_0^2 \delta \theta^2, \\ \delta \mathbf{R} \cdot \hat{\mathbf{n}} &= \frac{1}{2\gamma_0} [(r'_0 z_0'' - r''_0 z'_0) \delta t^2 + r_0 z'_0 \delta \theta^2]. \end{aligned} \quad (24)$$

The differential area  $dA$  is the product of the differential path lengths in the  $t$  and  $\theta$  directions given by

$$\begin{aligned}
 dA &= |\mathbf{R}_{t,0}| |\mathbf{R}_{\theta,0}| dt d\theta \\
 &= \gamma_0 r_0 dt d\theta.
 \end{aligned}
 \tag{25}$$

Substituting Eqs. (24) and (25) into the integral (16) gives

$$Q = \frac{P}{4\pi\epsilon_0} \frac{1}{2\gamma_0} \int_{\theta_0-\Delta\theta}^{\theta_0+\Delta\theta} \int_{t_0-\Delta t}^{t_0+\Delta t} \frac{(r'_0 z''_0 - r''_0 z'_0) \delta t^2 + r_0 z'_0 \delta \theta^2}{(\gamma_0^2 \delta t^2 + r_0^2 \delta \theta^2)^{3/2}} \gamma_0 r_0 dt d\theta.
 \tag{26}$$

By making the substitutions  $t \rightarrow t + t_0$  and  $\theta \rightarrow \theta + \theta_0$ , this becomes

$$Q = \frac{Pr_0}{8\pi\epsilon_0} \int_{-\Delta\theta}^{\Delta\theta} \int_{-\Delta t}^{\Delta t} \frac{(r'_0 z''_0 - r''_0 z'_0) t^2 + r_0 z'_0 \theta^2}{(\gamma_0^2 t^2 + r_0^2 \theta^2)^{3/2}} dt d\theta,
 \tag{27}$$

which can be integrated to yield

$$Q = \frac{P}{2\pi\epsilon_0} \left[ \frac{r_0}{\gamma_0^3} (r'_0 z''_0 - r''_0 z'_0) \Delta\theta \sinh^{-1} \left( \frac{\gamma_0 \Delta t}{r_0 \Delta \theta} \right) + \frac{z'_0}{r_0} \Delta t \sinh^{-1} \left( \frac{r_0 \Delta \theta}{\gamma_0 \Delta t} \right) \right].
 \tag{28}$$

### 3. Analytical solution

The solution of Poisson’s equation in toroidal coordinates has been discussed in Ref. [3] to which we refer for details. The resulting expression for the electric potential is a complicated hierarchy of infinite series and continued fractions involving associated Legendre functions. Here we describe how this solution can be implemented efficiently in a computer program. Only the steps needed to describe the programming of the solution are given below.

#### 3.1. Poisson’s equation in toroidal coordinates

A torus shaped channel is generated by rotating around the  $z$ -axis a circle of radius  $r$  whose center is offset from the  $z$ -axis by  $R$  ( $R > r$ ). The system of toroidal coordinates  $(\mu, \eta, \phi)$  are related to the Cartesian ones through the expressions

$$x = \frac{a \sinh \mu \cos \phi}{\cosh \mu - \cos \eta}, \quad y = \frac{a \sinh \mu \sin \phi}{\cosh \mu - \cos \eta}, \quad z = \frac{a \sin \eta}{\cosh \mu - \cos \eta}.
 \tag{29}$$

The inverse relations, which are needed to transform the positions of the charges and field points from Cartesian to toroidal coordinates, are given by

$$\mu = \tanh^{-1} \frac{2a\sqrt{x^2 + y^2}}{x^2 + y^2 + z^2 + a^2},
 \tag{30}$$

$$\eta = \tan^{-1} \frac{2az}{x^2 + y^2 + z^2 - a^2},
 \tag{31}$$

$$\phi = \tan^{-1} \frac{y}{x}.
 \tag{32}$$

In toroidal coordinates, the two radii  $r$  and  $R$  of the torus are given by

$$r = \frac{a}{\sinh \mu_1}, \quad R = a \coth \mu_1,
 \tag{33}$$

with the inverse relations

$$a = \sqrt{R^2 - r^2}, \quad \mu_1 = \cosh^{-1} \frac{R}{r}. \quad (34)$$

As  $\eta$  changes from 0 to  $2\pi$ , constant  $\mu_1$  follows a circle with the minor radius  $r$  centered at the major radius  $R$ .

Solution of Laplace's equation in toroidal coordinates is given in terms of the trigonometric functions for  $\eta$  and  $\phi$ , and the toroidal harmonics (Legendre functions of half-order)  $P_{n-1/2}^m(\cosh \mu)$ ,  $Q_{n-1/2}^m(\cosh \mu)$ . The potential due to a point charge  $q$  at  $\mathbf{r}_0 = (\mu_0, \eta_0, \phi_0)$  can be similarly expanded in toroidal coordinates [6]. The solution of Poisson's equation for the system of a point charge outside the toroidal boundary  $\mu = \mu_1 > \mu_0$ , with dielectric constants  $\epsilon_1$  outside and  $\epsilon_2$  inside the torus, can be written as

$$\begin{aligned} \varphi_{\text{in}} &= f(\mu, \eta) \sum_{n=-\infty}^{\infty} \sum_{m=0}^{\infty} A_{nm} Q_{n-1/2}^m(\cosh \mu) \exp[in(\eta - \eta'_{nm})] \cos(\phi - \phi_0), \\ \varphi_{\text{out}} &= f(\mu, \eta) \sum_{n=-\infty}^{\infty} \sum_{m=0}^{\infty} \left[ B_{nm} P_{n-1/2}^m(\cosh \mu) \exp[in(\eta - \eta''_{nm})] \right. \\ &\quad \left. + C_{nm} Q_{n-1/2}^m(\cosh \mu) \exp[in(\eta - \eta_0)] \right] \cos m(\phi - \phi_0), \end{aligned} \quad (35)$$

where  $A_{nm}$ ,  $B_{nm}$ ,  $\eta'_{nm}$  and  $\eta''_{nm}$  are the expansion coefficients, and

$$C_{nm} = \frac{1}{4\pi\epsilon_0\epsilon_1} \frac{q}{\pi a} f(\mu_0, \eta_0) (2 - \delta_{m0}) \frac{\Gamma(n - m + 1/2)}{\Gamma(n + m + 1/2)} P_{n-1/2}^m(\cosh \mu_0) \quad (36)$$

arise from the expansion of the point charge potential. The function  $f(\mu, \eta)$  is defined as

$$f(\mu, \eta) = \sqrt{\cosh \mu - \cos \eta}. \quad (37)$$

Applying the usual boundary conditions at  $\mu = \mu_1$ ,

$$\varphi_{\text{in}} = \varphi_{\text{out}}, \quad \epsilon_2 \frac{\partial \varphi_{\text{in}}}{\partial (\cosh \mu)} = \epsilon_1 \frac{\partial \varphi_{\text{out}}}{\partial (\cosh \mu)}, \quad (38)$$

and manipulating the ensuing equations, one obtains the following second order difference equation [3]:

$$E_{n+1}^m - q_n^m E_n^m + E_{n-1}^m = \lambda_{n+1}^m - 2 \cosh \mu_1 \lambda_n^m + \lambda_{n-1}^m. \quad (39)$$

Here the various coefficients are defined as

$$\begin{aligned} E_n^m &= (\epsilon_2 Q' - \epsilon_1 P' Q/P) A_{nm} \exp[-in\eta'_{nm}], \\ q_n^m &= 2 \cosh \mu_1 + \frac{(\epsilon_2 - \epsilon_1) Q}{\epsilon_2 Q' - \epsilon_1 P' Q/P}, \\ \lambda_n^m &= \epsilon_1 (Q' - P' Q/P) C_{nm} \exp[-in\eta_0], \end{aligned} \quad (40)$$

and we have introduced the compact notation for the constants;  $P = P_{n-1/2}^m(\cosh \mu_1)$ ,  $Q = Q_{n-1/2}^m(\cosh \mu_1)$  and  $f = f(\mu_1, \eta)$ . Similarly, the primes over  $P$ ,  $Q$  and  $f$  denote derivatives with respect  $\cosh \mu$  evaluated at  $\mu = \mu_1$ . The real and imaginary parts of Eq. (39) must be satisfied separately leading to two difference equations which determine both the amplitude  $A_{nm}$  and the phase  $\eta'_{nm}$ . The remaining coefficients  $B_{nm}$  and  $\eta''_{nm}$  are obtained from

$$B_{nm} \exp[-in\eta''_{nm}] = (A_{nm} \exp[-in\eta'_{nm}] - C_{nm} \exp[-in\eta_0]) Q/P. \quad (41)$$

The second order difference equation (39) has to be solved for each value of  $m$  for real and imaginary parts separately. For convenience, we suppress the superscript  $m$  in the following. The first step is to find the Green function corresponding to Eq. (39) which satisfies [7]

$$G_{n+1,N} - q_n G_{n,N} + G_{n-1,N} = \delta_{n,N+1} - 2 \cosh \mu_1 \delta_{n,N} + \delta_{n,N-1}, \tag{42}$$

for each value of  $N$ . Here  $\delta_{n,N}$  denotes the Kronecker delta. Solutions of Eq. (39) are then given by

$$E_n = \sum_{N=-\infty}^{\infty} G_{n,N} \lambda_N. \tag{43}$$

To construct the Green function, one first finds the solutions of the homogeneous equation

$$G_{n+1,N} - q_n G_{n,N} + G_{n-1,N} = 0, \tag{44}$$

and then implements the “boundary conditions” implied in Eq. (42). The two independent solutions of Eq. (44) with the correct asymptotics are given in terms of the continued fractions as [8]

$$\begin{aligned} \frac{G_{n+1,N}}{G_{n,N}} &= \frac{1}{q_{n+1} - \frac{1}{q_{n+2} - \frac{1}{q_{n+3} - \dots}}} \equiv \alpha_{n+1}, \\ \frac{G_{n-1,N}}{G_{n,N}} &= \frac{1}{q_{n-1} - \frac{1}{q_{n-2} - \frac{1}{q_{n-3} - \dots}}} \equiv \beta_{n-1}. \end{aligned} \tag{45}$$

Eq. (45) can be written as recursion relations among  $\alpha_n$  and  $\beta_n$ ,

$$\alpha_n = \frac{1}{q_n - \alpha_{n+1}}, \quad \beta_n = \frac{1}{q_n - \beta_{n-1}}, \tag{46}$$

which provide a simple method for their calculation by iteration. From the symmetry properties of  $P_{n-1/2}^m$ ,  $Q_{n-1/2}^m$  and their derivatives (they remain invariant under  $n \rightarrow -n$ ), it follows that  $q_{-n} = q_n$  in Eq. (40). Using this fact in Eq. (46), it is seen that  $\alpha_n = \beta_{-n}$ , and therefore only one set of coefficients needs to be calculated. Rewriting Eq. (45) as

$$\begin{aligned} G_{n+1,N} &= \alpha_{n+1} G_{n,N}, \quad n \geq N + 1, \\ G_{n-1,N} &= \beta_{n-1} G_{n,N}, \quad n \leq N - 1, \end{aligned} \tag{47}$$

$G_{n,N}$  can be determined from Eq. (47) recursively, once  $G_{N+1,N}$  and  $G_{N-1,N}$  are specified. Using the “boundary conditions” on Eq. (42) at  $n = N - 1, N, N + 1$  yields the following solutions:

$$\begin{aligned} G_{N-1,N} &= \frac{(2 \cosh \mu_1 - q_N) \beta_{N-1}}{q_N - \alpha_{N+1} - \beta_{N-1}}, \\ G_{N,N} &= \frac{(2 \cosh \mu_1 - \alpha_{N+1} - \beta_{N-1})}{q_N - \alpha_{N+1} - \beta_{N-1}}, \\ G_{N+1,N} &= \frac{(2 \cosh \mu_1 - q_N) \alpha_{N+1}}{q_N - \alpha_{N+1} - \beta_{N-1}}. \end{aligned} \tag{48}$$

Substituting Eqs. (47) and (48) in Eq. (43), we finally obtain for the coefficients  $E_n$ ,

$$E_n = \sum_{N=-\infty}^{\infty} \frac{\lambda_N}{(q_N - \alpha_{N+1} - \beta_{N-1})} \left\{ (2 \cosh \mu_1 - \alpha_{N+1} - \beta_{N-1}) \delta_{n,N} \right. \\ \left. + (2 \cosh \mu_1 - q_N) \left[ \theta(n - N) \prod_{k=N+1}^n \alpha_k + \theta(N - n) \prod_{k=n}^{N-1} \beta_k \right] \right\}, \quad (49)$$

where  $\theta(x)$  is the step function.

The solution for a uniform electric field, applied along the symmetry axis of the torus, follows the same lines as above but it is much simpler due to the axial symmetry [7]. The potential is independent of the coordinate  $\phi$ , hence there are no  $m$ -sums in the solutions contrary to the case for the point charge solution equation (35). Further, there are no phase differences in the  $\eta$  solutions, i.e. they are given by  $\exp[in\eta]$  everywhere. The potential for a uniform field in toroidal coordinates is given by

$$\varphi_{\text{ap}} = E_0 z \\ = E_0 \frac{\sqrt{8a}}{i\pi} f(\mu, \eta) \sum_{n=-\infty}^{\infty} n Q_{n-1/2}(\cosh \mu) \exp[in\eta]. \quad (50)$$

Superposing  $\varphi_{\text{ap}}$  with the free fields in Eq. (35) (without the  $m$ -sums), and applying the boundary conditions Eq. (38), one obtains again a second order difference equation as in Eq. (39) but without the  $m$  indices. The coefficients  $E_n$  and  $q_n$  are the same as in Eq. (40) and  $\lambda_n$  is modified to

$$\lambda_n = \epsilon_1 (Q' - P'Q/P) E_0 \frac{\sqrt{8a}}{\pi} n. \quad (51)$$

This difference equation is solved following the same steps described in Eqs. (42)–(49) above.

### 3.2. Numerical implementation

As seen from Eq. (49), the solution involves an infinite sum of series of products. Therefore, for a fast, yet accurate computation of the potential, an efficient evaluation of the coefficients  $E_n$  is necessary. A properly optimized computer code has been written for this purpose, which computes the electric potential for arbitrary number of ions under an applied electric field.

The expression (49) is evaluated from the bottom up, using a fixed number of terms for all of the series and continued fractions. The intermediate results are stored in arrays as they are generated, and kept for possible reuse. The calculation is divided into three pieces: the boundary, the charges, and the potentials. Thus the calculations for the boundary do not have to be redone if the positions of the charges change, and the calculations for the charges do not have to be redone for each point of interest. Due to the fixed number of terms, the calculations for each charge and each potential are the same, and the algorithm can be easily vectorized.

The alternative top down approach is conceptually simpler, and allows better control of precision, as each potential can be summed until the desired accuracy has been reached and no further. However, it is hopelessly inefficient if implemented simply, as many quantities are repeatedly recalculated, and some of the recurrence relations do not run in the same direction as the algorithm. We attempted to overcome these problems by storing intermediate results and reusing them when available, however the top down algorithm does not generate intermediate results in a predictable order, and keeping track of them severely complicated the code, resulting in a slow and unreliable program. In addition, because each calculation was treated differently, the program was not vectorizable. Because of these considerations, the bottom up algorithm is far superior. Below we describe the basic elements of the program.

### 3.2.1. Legendre functions

The Legendre Functions are calculated using the recurrence relations [9]

$$\begin{aligned}
 P_{n-1/2}^m(z) &= \frac{1}{n-m-1/2} \left[ (2n-2)zP_{n-3/2}^m(z) - (n+m-3/2)P_{n-5/2}^m(z) \right], \\
 Q_{n-1/2}^m(z) &= \frac{1}{n+m+1/2} \left[ (2n+2)zQ_{n+1/2}^m(z) - (n-m+3/2)Q_{n+3/2}^m(z) \right].
 \end{aligned}
 \tag{52}$$

Note that  $P$  is calculated for successively increasing values of  $n$ , while  $Q$  is calculated for decreasing values. This is necessary for the recurrences to be stable.

The key values used to start the recurrences are given by [11]

$$\begin{aligned}
 P_{n-1/2}^m(z) &= \frac{\Gamma(n+m+1/2)}{\Gamma(n-m+1/2)} \frac{(z^2-1)^{m/2} z^{n-m-1/2}}{2^m m!} \\
 &\quad \times F\left( (m-n+1/2)/2, (m-n+3/2)/2; m+1; \frac{z^2-1}{z^2} \right)
 \end{aligned}
 \tag{53}$$

and

$$\begin{aligned}
 Q_{n-1/2}^m(z) &= \frac{\Gamma(1/2)\Gamma(n+m+1/2)}{\Gamma(n+1)} \frac{(-1)^m}{\sqrt{2}(z^2-1)^{1/4}(z+\sqrt{z^2-1})^n} \\
 &\quad \times F(1/2+m, 1/2-m; n+1; -t),
 \end{aligned}
 \tag{54}$$

where

$$t = \frac{z - \sqrt{z^2 - 1}}{2\sqrt{z^2 - 1}}.
 \tag{55}$$

$\Gamma$  is the gamma function and  $F$  is the hypergeometric function: these are calculated using the formulas given in Ref. [9]. Recurrences for increasing and decreasing  $m$  are unstable, so new key values are needed for each value of  $m$ .

The derivatives of the Legendre functions  $P$  and  $Q$  are given by [11]

$$\begin{aligned}
 \frac{dU_{n-1/2}^m(z)}{dz} &= \frac{1}{z^2-1} \left[ (n-1/2)zU_{n-1/2}^m(z) - (n+m-1/2)U_{n-11/2}^m(z) \right], \\
 \frac{dU_{n-1/2}^m(z)}{dz} &= \frac{1}{z^2-1} \left[ -(n+1/2)zU_{n-1/2}^m(z) + (n-m+1/2)U_{n+1/2}^m(z) \right],
 \end{aligned}
 \tag{56}$$

where  $U$  can be either  $P$  or  $Q$ .

### 3.2.2. Green function

The Green function  $G_{n,N}^m$ , as defined in Eq. (42), depends only on the boundary through  $q_n^m$  (40), that is, the internal and external radii of the torus,  $r$  and  $R$ , and the dielectric constants inside and outside the torus  $\epsilon_2$  and  $\epsilon_1$ . Since it is independent of the position of the charges, it is more economical to calculate it once at the beginning and store the values in an array. Thus we first calculate  $q_n^m$  using Eq. (40), and then generate values of  $\alpha_n^m$  and  $\beta_n^m$  using  $q_n^m$  in the recursion relations (46). Values of  $G_{n,N}^m$  are calculated using the recurrence relations (47) with the key values given by Eqs. (48). The next stage of calculation requires values of  $G_{n,N}^m$  with  $N$  varying between  $n-l_{\max}$  and  $n+l_{\max}$ . These are stored in an array indexed by  $m$ ,  $n$ , and  $l$ , with  $N = n+l$ , so that as  $l$  goes between  $-l_{\max}$  and  $l_{\max}$ ,  $N$  goes between  $n-l_{\max}$  and  $n+l_{\max}$ , as required.

The recurrence relations generate values of  $G_{n,N}^m$  for fixed  $N$  and increasing and decreasing  $n$ , whereas what is needed are values for fixed  $n$  and increasing and decreasing  $N$ . The recurrence series run diagonally across the

array (indexed by  $n$  and  $l$ ), so care must be taken to generate all required values, and some key values outside the table have to be generated. Values of  $G_{n,N}^m$  for negative values of  $N$  can be generated using  $\alpha_{-n}^m = \beta_n^m$ , and  $\beta_{-n}^m = \alpha_n^m$ .

### 3.2.3. Charges

Here we describe the calculation of quantities that depend only on the boundary and the positions of the charges. We assume that there are  $i_{\max}$  charges with magnitude  $q_{0i}$  and coordinates  $\mathbf{r}_i = (\mu_{0i}, \eta_{0i}, \phi_{0i})$ . The expansion coefficients  $E_{nm}$  (43) in the potential involve the Green function and  $\lambda_N^m$  (40). In numerical evaluations, it is easier to deal with real numbers, therefore we separate the real ( $R$ ) and imaginary ( $I$ ) parts of the quantities in Eq. (43) using

$$\lambda_N^m(R) = |\lambda_N^m| \cos n\eta_0, \quad \lambda_N^m(I) = |\lambda_N^m| \sin n\eta_0, \quad (57)$$

and similar relations for  $E_{nm}$ . The real part of Eq. (43) then becomes

$$E_{nm}(R) = \sum_{N=n-I_{\max}}^{n+I_{\max}} \lambda_N^m(R) G_{n,N}^m. \quad (58)$$

An identical equation follows for the imaginary part. Here we have truncated the infinite sum to appropriately chosen minimum and maximum values. For negative values of  $N$ , note that  $\lambda_{-N}^m(R) = \lambda_N^m(R)$  and  $\lambda_{-N}^m(I) = -\lambda_N^m(I)$ . In the same vein, we introduce the following quantities in place of the complex  $A_{nm} \exp[-in\eta'_{nm}]$  and  $B_{nm} \exp[-in\eta''_{nm}]$  in the potentials (35):

$$\begin{aligned} A_{nm}(R) &= A_{nm} \cos n\eta'_{nm}, & A_{nm}(I) &= A_{nm} \sin n\eta'_{nm}, \\ B_{nm}(R) &= B_{nm} \cos n\eta''_{nm}, & B_{nm}(I) &= B_{nm} \sin n\eta''_{nm}. \end{aligned} \quad (59)$$

Terms in the sums for the potentials can then be calculated using the relations

$$\begin{aligned} A_{nm} \cos n(\eta - \eta'_{nm}) &= A_{nm}(R) \cos n\eta + A_{nm}(I) \sin n\eta, \\ B_{nm} \cos n(\eta - \eta'_{nm}) &= B_{nm}(R) \cos n\eta + B_{nm}(I) \sin n\eta. \end{aligned} \quad (60)$$

The effect of all charges on a point outside the torus can be evaluated with a single sum by precalculating the sums involving the  $B_{nm}$  for individual charges. This process is described in the next section, but since they depend only on the boundary and the positions of the charges, we introduce these sums here,

$$\begin{aligned} S_{nm}^1 &= \sum_{i=1}^{i_{\max}} B_{nmi}(R) \cos m\phi_{0i}, & S_{nm}^2 &= \sum_{i=1}^{i_{\max}} B_{nmi}(I) \cos m\phi_{0i}, \\ S_{nm}^3 &= \sum_{i=1}^{i_{\max}} B_{nmi}(R) \sin m\phi_{0i}, & S_{nm}^4 &= \sum_{i=1}^{i_{\max}} B_{nmi}(I) \sin m\phi_{0i}, \end{aligned} \quad (61)$$

where the index  $i$  ranges over all the charges.

### 3.2.4. Potentials and fields

Here we first describe the calculation of the electric potential at a given point with coordinates  $\mathbf{r} = (\mu, \eta, \phi)$ . The potential at point  $\mathbf{r}$  due to the charge  $i$  at  $\mathbf{r}_i$  consists of the trivial Coulomb part plus the contribution from the charges on the boundary, which is given by

$$\begin{aligned} \phi_{Bi} &= f(\mu, \eta) \sum_{m=0}^{m_{\max}} (2 - \delta_{0m}) \cos m(\phi - \phi_{0i}) \\ &\times \sum_{n=0}^{n_{\max}} (2 - \delta_{0n}) [B_{nmi}(R) \cos n\eta + B_{nmi}(I) \sin n\eta] P_{n-1/2}^m(\cosh \mu). \end{aligned} \quad (62)$$

Summing over all charges, the potential at point  $\mathbf{r}$  due to the boundary becomes

$$\begin{aligned} \phi_B &= f(\mu, \eta) \sum_{i=1}^{i_{\max}} \sum_{m=0}^{m_{\max}} (2 - \delta_{0m}) [\cos m\phi \cos m\phi_{0i} + \sin m\phi \sin m\phi_{0i}] \\ &\times \sum_{n=0}^{n_{\max}} (2 - \delta_{0n}) [B_{nmi}(R) \cos n\eta + B_{nmi}(I) \sin n\eta] P_{n-1/2}^m(\cosh \mu), \end{aligned} \quad (63)$$

where we have expanded  $\cos m(\phi - \phi_{0i})$ . Substituting the sums over charges introduced in (61), Eq. (63) can be written as

$$\begin{aligned} \phi_B &= f(\mu, \eta) \sum_{m=0}^{m_{\max}} \sum_{n=0}^{n_{\max}} (2 - \delta_{0m})(2 - \delta_{0n}) [(S_{nm}^1 \cos n\eta + S_{nm}^2 \sin n\eta) \cos m\phi \\ &+ (S_{nm}^3 \cos n\eta + S_{nm}^4 \sin n\eta) \sin m\phi] P_{n-1/2}^m(\cosh \mu). \end{aligned} \quad (64)$$

Hence, the potential at  $N$  different points due to  $i_{\max}$  charges can be calculated in  $O(i_{\max}) + O(N)$  operations, rather than the  $O(i_{\max} \times N)$  operations that would be required if Eq. (63) was used naively.

The electric field is calculated from  $\mathbf{E} = -\nabla\phi$ . The required partial derivatives of the potentials can be found by differentiating Eq. (64). Note that  $\partial\psi/\partial\mu$  has a singularity at  $\mu = 0$ , which is due to the inability of the toroidal coordinate system to describe the  $x$ - $y$  direction of the field for a point on the  $z$ -axis. We solve this problem by moving any point on the  $z$ -axis a distance of  $a \times 10^{-6}$  off the axis for the purpose of calculation.

To find the electric field in Cartesian coordinates, the partial derivatives with respect to the toroidal coordinates have to be converted to partial derivatives with respect to Cartesian coordinates. This is done using the relations

$$\begin{aligned} \frac{\partial\phi}{\partial x} &= \left( \alpha \frac{\partial\psi}{\partial\mu} + \beta \frac{\partial\psi}{\partial\eta} \right) \cos\phi - \gamma \frac{\partial\psi}{\partial\phi} \sin\phi, \\ \frac{\partial\phi}{\partial y} &= \left( \alpha \frac{\partial\psi}{\partial\mu} + \beta \frac{\partial\psi}{\partial\eta} \right) \sin\phi + \gamma \frac{\partial\psi}{\partial\phi} \cos\phi, \\ \frac{\partial\phi}{\partial z} &= \beta \frac{\partial\psi}{\partial\mu} - \alpha \frac{\partial\psi}{\partial\eta}, \end{aligned} \quad (65)$$

where

$$\alpha = \frac{1}{a}(1 - \cosh\mu \cos\eta), \quad \beta = -\frac{1}{a} \sinh\mu \sin\eta, \quad \gamma = \frac{1}{a} \left( \frac{\cosh\mu - \cos\eta}{\sinh\mu} \right). \quad (66)$$

The description so far has been for both points and charges outside the toroidal boundary. For charges inside the boundary and points still all outside, some modifications need to be made to the formulas. This involves exchanging  $P$  with  $Q$ , and  $\epsilon_1$  with  $\epsilon_2$  in the expansion of point charge, Eqs. (35), (36). The coefficients  $B_{nm}$  are modified to

$$B_{nm}(R) = (A_{nm}(R) - C_{nm} \cos n\eta_0), \quad B_{nm}(I) = (A_{nm}(I) - C_{nm} \sin n\eta_0). \quad (67)$$

The factor  $P_{n-1/2}^m(\cosh\mu)$  in the formula for the potential is unchanged.

#### 4. Testing

The precision of the analytical solution depends on the number of terms included in the sums. We investigate its convergence characteristics by comparing the results with different number of terms in the sums. For this purpose we use a torus with internal and external radii of 40 Å and 44 Å, respectively, which generates a typical size ion channel in membranes (4 Å radius in the narrowest region). The best precision for a given amount of computational effort occurs when  $n_{\max}$  was about twice  $m_{\max}$ . With  $n_{\max} \sim 100$ , an accuracy of better than 1% is obtained everywhere except at very close distances to the boundary ( $\sim 1$  Å). Because of the large repulsive image forces at close distances, ions are unlikely to come very near the boundary in computer simulations. Hence in simulation studies, these errors are not expected to play any significant role. Finally,  $l_{\max} \sim 10$  is used in the calculation of the Green function. Since the value of  $l_{\max}$  affects the precision of all calculations, not just those near the boundary, a larger value is required for higher accuracy.

As a second test, we compare the analytical results with those obtained from the iterative numerical method for the torus boundary described above. For this purpose, we examine the potential barrier and the magnitude of a repulsive force the ion experiences as it moves in various directions in the channel. For the numerical calculations, the toroid is divided into 9200 sectors. The size of sectors is smallest at the narrow channel region and becomes progressively larger at the wider region of the vestibules. The values calculated from the analytical solution (solid lines) are superimposed on those calculated by using the iterative numerical methods (filled circles). In Fig. 1, the ion moves into the channel along the  $z$ -direction but offset from the  $z$ -axis by 3 Å. Agreement between the two methods gets better when the offset is smaller than 3 Å. In Fig. 2, the ion moves from the center of the torus towards the boundary on the  $z = 0$  plane. Finally, in Fig. 3, the ion moves on the  $z = 30$  Å plane towards to the boundary. Because the segments have larger sizes in this region, there appears an appreciable error in the numerical solutions when the ion is very near the boundary ( $\sim 1$  Å). The above comparisons indicate that both methods provide reliable solutions of Poisson's equation in channel-like geometries, and would be very useful in computer simulation studies of ion transport problem in membrane channels.

#### 5. The BICS package

The BICS package is a collection of software useful for generating channel boundaries and solving Poisson's equation for those boundaries. BICS stands for Biological Ion Channel Simulator. Typically, the package does not include a simulator as yet, but it has been used with a separate program to perform Brownian dynamics simulations [4]. The software is written in FORTRAN 90 and consists of a main program and 12 modules. The main program `bics_profile` generates outlines of the channel shape and potential profiles for an ion. The module `bics_channel` acts as a library interface, allowing the software to be incorporated into other programs.

##### 5.1. Top level design

We have exploited the new features FORTRAN 90, which allows a modular approach to programming [12], and structured BICS as a collection of modules using the concept of abstract data types [13]. The idea is to hide data structures and algorithms inside each module, and provide a simplified interface in the form of subroutines which perform the necessary operations on the data. Each module contains data which is preserved between subroutine calls, and is accessible to all of the module's subroutines, but is not accessible outside the module except via the interface routines. This approach avoids the use of global data structures, and makes the software easier to understand and modify due to the increased independence of the modules.

The overall design of the software is shown in Fig. 4. The arrows indicate dependencies: the module at the tail of an arrow calls routines from the module at the head, and so depends on it. Not all dependences are shown,

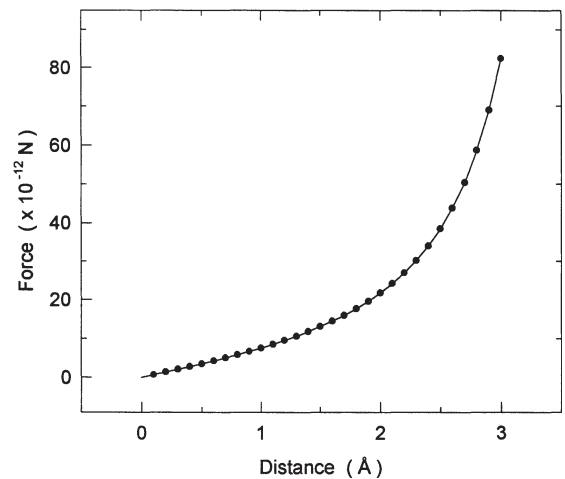
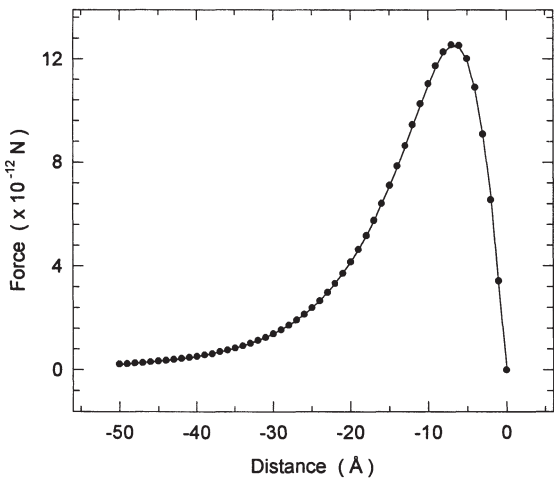
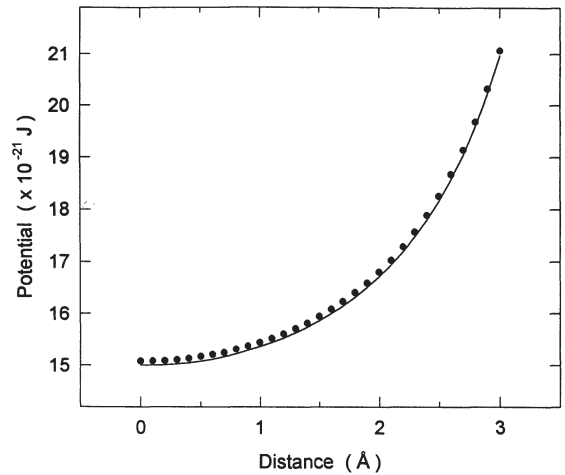
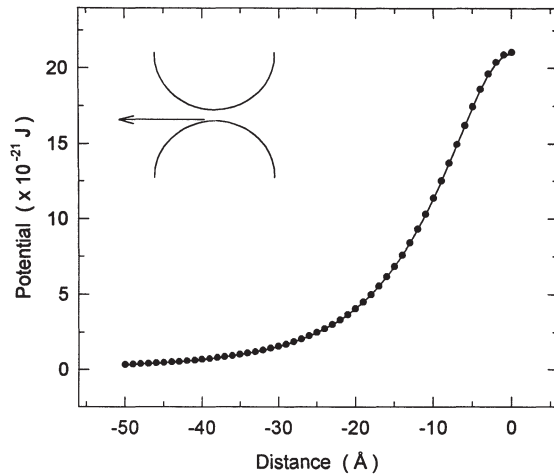


Fig. 1.

Fig. 2.

Fig. 1. Comparison of analytical (line) and numerical results (dots) for the electric potential and force calculated for a toroidal boundary. The ion moves along the  $z$ -direction but offset from the  $z$ -axis by 3 Å.

Fig. 2. Same as Fig. 1 but the ion moves from the center of the torus towards the boundary on the  $z = 0$  plane.

to avoid cluttering the diagrams, e.g. the `an_main` module also depends on `an_boundary` and `an_functions`. All the modules depend on `bics_basics` which is not shown in Fig. 4.

## 5.2. The `bics_profile` program

The `bics_profile` program generates a potential profile along a straight line in the vicinity of the channel. The profile line is divided into segments, and the program reports the potential energy of the ion as well as the force on the ion at the boundaries of each segment. Optionally, the program can also include the effect of an external field and of fixed charges. Alternatively, the program can generate an electric potential profile for an external field or fixed charges in the absence of an ion. The program then reports the electric potential and electric field at each segment position, instead of potential energy and force. In addition this program will, if

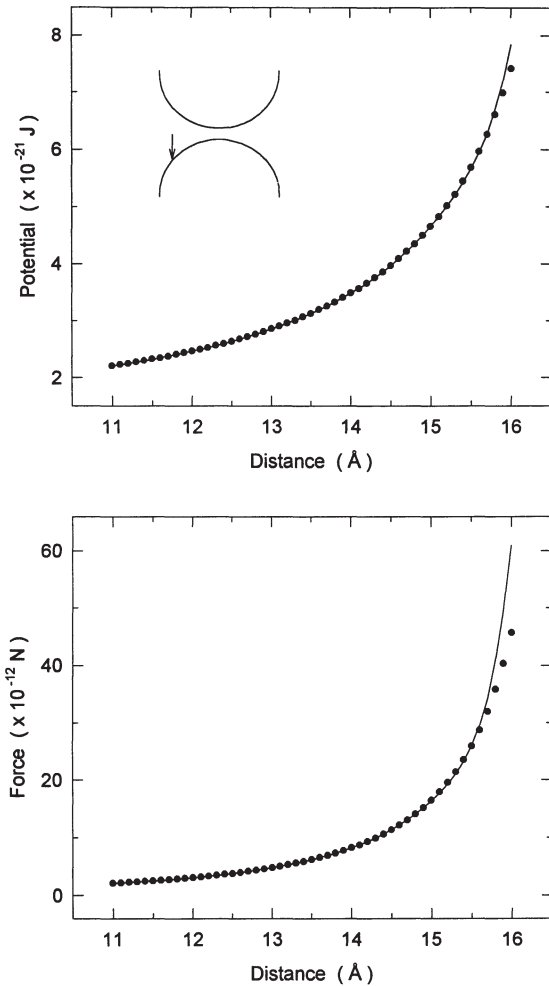


Fig. 3.

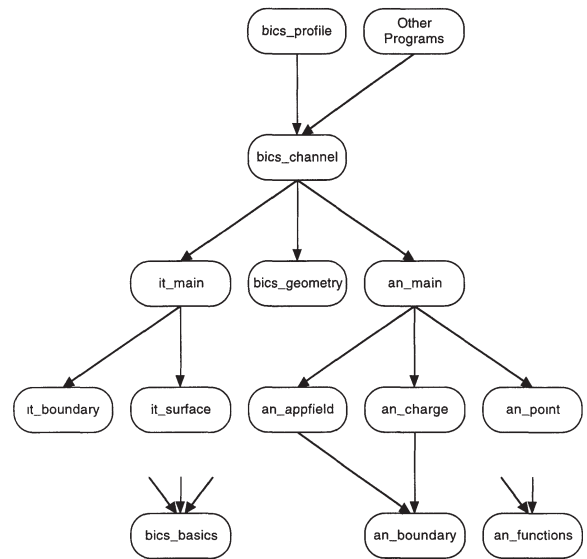


Fig. 4.

Fig. 3. Same as Fig. 1 but the ion moves on the  $z = 30$  Å plane towards to boundary.

Fig. 4. Top level dependency diagram for BICS. The prefixes ‘it’ and ‘an’ stand for iterative and analytic, respectively.

requested, generate a two-dimensional outline of the channel shape, or a radius function offset from this outline by a specified amount. This facility is useful for generating plots of the channel shape.

### 5.2.1. Input

The program reads its parameters from the standard input in FORTRAN 90 namelist format (see [12]). The parameters are recorded in an input file, e.g. `run1.in`, and the program starts with the command `bics_profile.exe < run1.in`. Since there are many parameters, and only a few need to be changed between runs, copying and editing an input file would be more convenient. Example input files are provided with the code. All of the input parameters are expressed in unmultiplied SI units to avoid confusion.

The parameters are divided into four groups. The profile parameters specify the profile line and the options to use when generating the profile, as well as the output file name (Table 1, top). The outline parameters specify whether or not to generate the outline or radius function, and if so what options and output file names

Table 1  
Input parameters for potential profile, channel outline and method

Variable	Description
use_ion	If this is <i>t</i> the program will calculate the potential energy and force on a single ion at positions along the profile line. Otherwise it will calculate the electric potential and field in the absence of any ions.
use_fixed_charges	If this is <i>t</i> the program will include the effects of fixed charges and the external field. Otherwise it will ignore these, even if they are specified in the channel parameters.
profile_segments	The number of segments to divide the profile line into.
x0, y0, z0	The start of the line in Cartesian coordinates.
x1, y1, z1	The end of the line in Cartesian coordinates.
t0	The start of the line for distances measured along the line.
q	The charge on the ion.
output_file_name	The name of the file to write the output to.
trace_outline	If this is <i>t</i> the program will generate a two-dimensional cross section of the channel boundary.
trace_radius	If this is <i>t</i> the program will generate a radius function representing a generalized cylinder based on the channel boundary.
min_z, max_z	The <i>z</i> coordinates of the ends of the generalized cylinder.
max_r	The maximum radius of the generalized cylinder.
buffer_dist	The minimum distance from the channel boundary to the generalized cylinder.
outline_segments	The number of segments to divide the cross section and radius function into, for the purpose of output in numerical form.
outline_file_name	The name of the file to write the cross section to.
radius_file_name	The name of the file to write the radius function to.
method	Indicates which method to use: 1 iterative, 2 analytical method.
max_m, _n, _l, _k	The number of terms to truncate at for various series in the analytical method. Set to $-1$ to select the default.
sector_count	The number of sectors to use in the iterative method. Set to $-1$ to select the default.
boundary_segments	The number of segments used to describe the channel outline to the iterative method. Set to $-1$ to select the default.
neck_spc, vest_spc, out_spc	The relative spacing between sectors in the neck, vestibule, and outer regions of the channel respectively. Set to $-1$ to select the default.

to use (Table 1, middle). The method parameters specify which method to use to calculate the profile, as well as parameters which affect the accuracy of the method (Table 1, bottom). See Sections 3 and 5.5.2 for more information on the parameters for the analytical and iterative methods. The channel parameters specify the dielectric boundary, the fixed charges, and the external field (Table 1). See Section 5.4 and Figs. 5 and 6 for more information.

### 5.2.2. Output

The program creates the file named in the profile parameters, and writes the potential profile to it. The output is in ASCII format, with one line per position. The output file has six columns. The first column gives the distance of each position along the profile line from the origin set by the *t0* parameter. The second column gives the potential energy of an ion (or the electric potential if there is no ion). The third column gives the force on an ion *in the direction of the profile line* (or the electric field in the same direction if there is no ion). The fourth, fifth, and sixth columns give the force or electric field in Cartesian coordinates. To assist in plotting, the output is given in atomic units: Å for length,  $10^{-21}$  J for potential energy,  $10^{-12}$  N for force, mV for electric potential, and  $10^6$  V/m for electric field.

The program creates files containing the channel outline and radius function if these are requested in the outline parameters. These files are in ASCII format, and the two columns give the *r* and *z* coordinates in Å. The program also writes diagnostic information to the standard output. This is mostly a description of what the

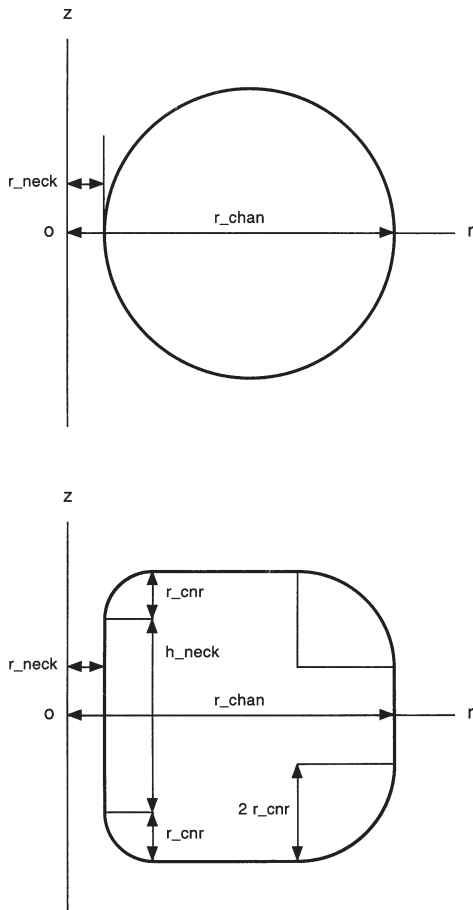


Fig. 5.

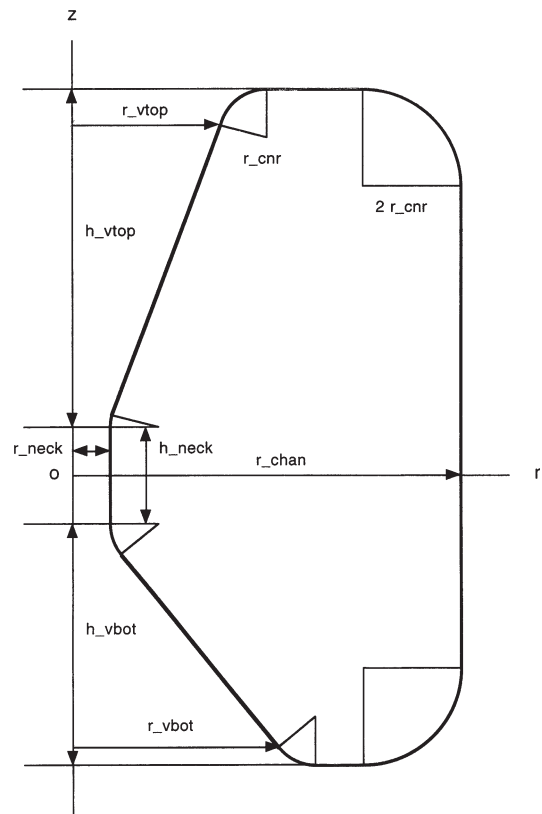


Fig. 6.

Fig. 5. Geometric parameters for toroidal and cylindrical channels.

Fig. 6. Geometric parameters for biconical and catenary channels. The diagram is an outline for a biconical channel, but catenary channels use the same parameters.

program is doing, but does include the values of some quantities calculated by the program. The path length is the length of the channel outline; it is only reported if the channel outline is requested. The quantity  $a$  is the scale factor for the catenary  $z = a \cosh(r/a) + b$  which forms the vestibules for the catenary channel. It is reported for the top and bottom vestibules when a catenary channel is used. The quantity  $\lambda$  is the ratio of the final spacing between sectors to the spacing requested. It is reported if the iterative method is used. The path length and  $a$  are given in Å and  $\lambda$  is dimensionless.

### 5.3. The `bics_channel` module

This module is a simplified interface to the analytical method, iterative method, and the `bics_geometry` module. It allows a program to switch between the two methods easily. It allows channels to be described by geometric parameters rather than by a cross section as required by the iterative method. It can also generate boundaries for simulation programs. Input to and output from the routines is in unmultiplied SI units.

Table 2  
Input parameters for the ion channel; see Figs. 5 and 6 for more information

Variable	Description
shape	Indicates the overall shape of the channel: 1 toroidal channel, 2 cylindrical channel, 3 biconical channel, 4 catenary channel.
r_chan	The outer radius of the channel at its widest point.
r_neck	The inner radius of the channel at its narrowest point.
h_neck	The length of the narrow neck region.
r_cnr	The radius of the rounded corners used to connect pieces of the boundary. The outer corners have twice this radius.
r_vtop	The width of the top vestibule.
h_vtop	The height of the top vestibule.
r_vbot	The width of the bottom vestibule.
h_vbot	The height of the bottom vestibule.
eps1	The dielectric constant outside the boundary (inside the pore).
eps2	The dielectric constant inside the boundary.
fc_count	The number of fixed charges.
fc_x,_y,_z	The position of the fixed charges in Cartesian coordinates. Each of these is a list of fc_count values.
fc_q	The amount of charge on the fixed charges. This is a list of fc_count values.
fc_q_mult	This value multiplies <i>all</i> the fc_q values. It is used to scale or turn off the fixed charges without having to modify the fc_q list.
ext_fld	The strength of the external electric field. This is a constant electric field parallel with the z-axis.
ext_pt1	The potential due to the external field at $z = 0$ .

### 5.3.1. Subroutine `ch_set_parameters(max_m,max_n,max_l,sector_count,neck_spc,vest_spc,out_spc)`

This routine sets parameters affecting the accuracy of the two methods. These parameters have reasonable defaults, so the routine may not need to be called. Giving  $-1$  as a parameter value sets the default. This allows some parameters to be set while keeping the defaults for others. Calling `ch_set_parameters` invalidates any previous call to `ch_specify_model`. Hence `ch_set_parameters` should be called before `ch_specify_model`. See Table 1 for a description of the parameters.

### 5.3.2. Subroutine `ch_get_parameters(max_m,max_n,max_l,sector_count,neck_spc,vest_spc,out_spc)`

This routine retrieves the current parameter settings. This can be used to see what the defaults are if it is called before `ch_set_parameters`. Its arguments are the same as those of `ch_set_parameters` except that they are outputs not inputs. See Table 1 for a description of the parameters.

### 5.3.3. Subroutine `ch_specify_model(shape,r_chan,r_neck,h_neck,r_cnr,r_vtop,h_vtop,r_vbot,h_vbot,eps1,eps2,fc_count,fc_c,fc_y,fc_z,fc_q,ext_fld,ext_pt)`

This routine sets the shape of the channel, the dielectric constants, the fixed charges around the channel, and the external electric field. It must be called before calculating electric fields or generating channel outlines. See Table 2 for a description of the parameters.

### 5.3.4. Subroutine `ch_calculate_field(method,calc_self_pt1,use_fixed_charges,ion_count,x0,y0,z0,q0,point_count,x1,y1,z1,fld_x,fld_y,fld_z,pt1,self_pt1)`

This routine calculates the electric field and potential at a group of points in and around the channel. A group of ions may also be given. The routine calculates the field and potential due to the dielectric boundary (as polarized by the ions, fixed charges, and external field), and includes the field and potential directly due to the fixed charges and the external field. It does *not* include the Coulomb field and potential arising directly from the ions – this must be calculated separately if required. The model must be specified before calling this subroutine.

*Input*

*method*: The method used in calculating the field: 1 for the iterative method, or 2 for the analytical method.

*calc\_self\_ptl*: If this is true the subroutine assumes that the positions of the points ( $x_1, y_1, z_1$ ) are the same as the positions of the ions ( $x_0, y_0, z_0$ ) and calculates the self-potential for each ion (see below). Setting this flag to true greatly slows down the iterative method in the case of multiple ions.

*use\_fixed\_charges*: If this is true the subroutine includes the effects of fixed charges and the external field, otherwise it does not.

*ion\_count*: The number of ions.

$x_0()$ ,  $y_0()$ ,  $z_0()$ : The positions of the ions.

$q_0()$ : The charges on the ions.

*point\_count*: The number of points. These are the points at which the electric field and potential are calculated.

$x_1()$ ,  $y_1()$ ,  $z_1()$ : The positions of the points.

*Output*

$f_{ld\_x}()$ ,  $f_{ld\_y}()$ ,  $f_{ld\_z}()$ : The electric field at the points.

$pt_1()$ : The electric potential at the points.

*self\_ptl* $()$ : The electric potential at the position of each ion due only to that ion's interaction with the boundary. The effects of fixed charges, the external field, and other ions are ignored. This only makes sense if the positions of the points correspond to the positions of the ions, and is only calculated if *calc\_self\_ptl* is true, in which case *ion\_count* must equal *point\_count*. The self-potential is needed to calculate the potential energy of an ion.

### 5.3.5. Subroutine *ch\_trace\_radius*(*segments, min\_z, max\_z, max\_r, buffer\_dist, r*)

This routine fills in an array giving the minimum radius of the channel boundary and a limiting cylinder, at evenly spaced  $z$  coordinates. A buffer distance may also be specified to provide an offset from the channel boundary. The model must be specified before calling this subroutine.

*Input*

*segments*: The number of segments to divide the  $z$ -axis into. The array *r* must run from 0:*segments*.

*min\_z*, *max\_z*: The range of the  $z$ -axis.

*max\_r*: The radius of the limiting cylinder.

*buffer\_dist*: The offset from the channel boundary.

*Output*

$r()$ :  $r(i)$  gives the radius at  $z = i * ((\text{max\_z} - \text{min\_z}) / \text{segments}) + \text{min\_z}$ . This radius is as large as possible while being within the limiting cylinder ( $r(i) \leq \text{max\_r}$ ) and being at least *buffer\_dist* away from the boundary.

### 5.3.6. Subroutine *ch\_trace\_outline*(*segments, path\_length, z, r*)

This routine fills in arrays which describe the outline of the channel boundary, being the  $z$  and  $r$  coordinates of points evenly spaced around the outline.

*Input*

*segments*: As in above routine.

*Output*

*path\_length*: The length of the outline.

$z()$ ,  $r()$ : The  $z$  and  $r$  coordinates of points on the outline. The point  $(z_i, r_i)$  is located at a distance  $(l * i) / n$  around the boundary, where  $l$  is the path length, and  $n$  is the number of segments. The starting point is that

with the minimum  $r$  coordinate of those with the minimum  $z$  coordinate, and the distance is measured clockwise around the outline. Note that  $(z_0, r_0)$  and  $(z_n, r_n)$  are the same point.

#### 5.4. The `bics_geometry` module

This module generates channel outlines suitable for use with the iterative method. It also generates boundaries useful to simulation programs. It can be used independently from the rest of the package if desired. The interface routines are `geo_specify_model`, `geo_trace_radius`, and `geo_trace_outline`. These perform the same functions as their equivalents in the `bics_channel` module, but the `geo_specify_model` only requires geometric channel parameters.

The module supports four types of channels: torus, cylinder, bicone and catenary. The size and shape of a particular boundary is described by geometric parameters supplied by the user, see Figs. 5 and 6. For all types of channel the neck is centered on the  $x$ - $y$  plane, `r_neck` gives the minimum radius, and `r_chan` gives the maximum. For the cylinder `h_neck` does not include the rounded corners, so the total height of the channel is `h_neck + 2 * r_cnr`. Rounded corners are necessary as the iterative method cannot cope with sharp edges, however they are also realistic – a gramicidin-like channel would cause dimples in the membrane similar to those in Fig. 5. For the bicone and catenary channels `h_vtop` and `h_vbot` do include the rounded corners, so the total height of the channel is `h_neck + h_vtop + h_vbot`. Note, however, that `r_vtop` and `r_vbot` are measured to the inner edge of the corner at the mouth of the vestibule, slightly below the top (or above the bottom) of the channel.

The output of the module is the two functions  $z(t)$  and  $r(t)$  where  $t$  is the path length clockwise around the outline. The functions are returned as large arrays, with elements containing the function values at evenly spaced values of  $t$ . Intermediate values can be obtained by interpolation. The module also provides a radius function  $r(z)$ , where  $r$  is the minimum of the inner edge of the outline and a cylinder specified by the user. This radius can be offset from the outline by a specified buffer distance. The radius function is useful for setting up a physical boundary at a certain distance from the dielectric boundary, for instance in a simulation program.

The process of generating the outline of the channel is as follows. The outline is divided into several sections, each part of a simple geometric curve: an arc, a line, or a catenary. The sections are described by parametric equations. Each section needs to be placed so that its endpoints match with those of the adjacent sections, both in position and slope. In addition the whole outline must match the geometric parameters supplied by the user. These constraints are solved using Ridders' method for finding the root of a one-dimensional function [10]. Once the sections are in position, the functions  $z(t)$  and  $r(t)$  for the whole outline are calculated by choosing the appropriate section for each value of  $t$  and then evaluating that section's parametric equations. Thus the outline is assembled piece by piece.

#### 5.5. Iterative method

The iterative method is a group of three modules which solve Poisson's equation for an arbitrary cylindrically symmetric boundary. The names of these modules are prefixed `it_` for identification. They can be used independently if desired, via the subroutines in `it_main`. See Fig. 4.

##### 5.5.1. The `it_main` module

This module is the interface to the iterative method. It uses the `it_boundary` module to generate sectors, and the `it_surface` module to calculate potential and field by applying the iterative method to those sectors. It has four interface routines: `it_set_parameters`, `it_get_parameters`, `it_specify_model`, and `it_calculate_field`. These perform the same functions as their equivalents in the `bics_channel` module,

except that they are specific to the iterative method and the channel shape is specified as an outline rather than by geometric parameters.

### 5.5.2. The *it\_boundary* module

The channel boundary needs to be divided into sectors to allow the iterative algorithm to operate. This is done by the *it\_boundary* module, which has a single interface routine *ibdy\_tile\_boundary*. The module takes the outline of the channel and rotates it by 360 degrees to form a cylindrically symmetric boundary. It then divides the boundary into rings by circular cuts around the  $z$ -axis, and each ring into sectors by longitudinal cuts co-planar with the  $z$ -axis. The distance between longitudinal cuts is kept as close as possible to the thickness of the ring, so the sectors are approximately square. The spacing between sectors varies in different parts of the boundary – the rings have different thicknesses and are divided into different numbers of sectors. The number of sectors increases as the inverse square of the sector spacing, and memory and computer time increase as the square of the number of sectors, or the inverse fourth power of the sector spacing. Therefore it is difficult to improve the accuracy by reducing the spacing uniformly. We instead vary sector spacing nonuniformly, reducing it where ions are expected to approach the boundary, and increasing it elsewhere. We define three regions of the boundary with different sector spacing: the neck, the vestibules, and the outer region. There are transition zones between these regions, where the spacing takes on intermediate values. The spacing needs to be smallest in the neck where ions approach the boundary closely and accuracy is most important. It can be larger in the vestibules where ions are likely to be further away. In the outer region (outside the reservoirs that contain ions) the spacing can be very large.

The sector spacings in the different regions are specified by the *neck\_spc*, *vest\_spc*, and *out\_spc* parameters, however these are only targets, not the absolute values of the spacing. The *it\_boundary* module makes the sectors as small as possible without exceeding the *sector\_count* parameter: this approach makes the memory use of the iterative method predictable. The sector sizes are kept in proportion but are scaled by a factor of  $\lambda$  to meet the specified sector count. For example, if *neck\_spc* = 0.5 Å, *vest\_spc* = 1.0 Å, and *out\_spc* = 10 Å (the default values), and the program reports that  $\lambda = 1.2$ , then the actual spacings are 0.6 Å in the neck, 1.2 Å in the vestibules, and 12 Å in the outer regions.

The location of the spacing regions is controlled by parameters passed to *it\_main* and *it\_boundary*. The edge of the neck region is defined by its  $z$  coordinate, while the edge of the vestibule region is defined by its  $r$  coordinate. The outer region is the remainder of the boundary. The neck region runs from *h\_neck\_spc\_bot* below the  $x$ - $y$  plane to *h\_neck\_spc\_top* above it, with transition zones extending 50% beyond these limits. The vestibule region runs from the neck region out to radius *r\_vest\_spc\_top* in the top half of the channel and *r\_vest\_spc\_bot* in the bottom half. Again, transition zones extend 50% beyond these limits. The *bics\_channel* module defines the spacing regions automatically as follows (parameters are the same for the top and bottom halves of the channel). For a cylinder,

$$\begin{aligned} h_{\text{neck\_spc}} &= \frac{2}{3} \left( \frac{1}{2} h_{\text{neck}} + r_{\text{cnr}} \right), \\ r_{\text{vest\_spc}} &= r_{\text{chan}} - 2r_{\text{cnr}}. \end{aligned} \quad (68)$$

For a torus,

$$\begin{aligned} r_{\text{torus}} &= \frac{1}{2} (r_{\text{chan}} - r_{\text{neck}}), \\ h_{\text{neck\_spc}} &= \frac{1}{4} r_{\text{torus}}, \\ r_{\text{vest\_spc}} &= r_{\text{neck}} + \frac{3}{4} r_{\text{torus}}. \end{aligned} \quad (69)$$

For a bicone or catenary channel,

$$\begin{aligned} h\_neck\_spc &= \frac{1}{2}h\_neck + r\_cnr, \\ r\_vest\_spc &= r\_chan - 2r\_cnr. \end{aligned} \tag{70}$$

If these need to be changed either the routines in `it_main` can be called directly, or `bics_channel` can be modified.

### 5.5.3. The `it_surface` module

This module applies the iterative algorithm described in Section 2 to solve Poisson's equation for a boundary represented as a grid of sectors. It has two interface functions. The first of these, `isfc_initialize`, accepts a description of the sectors and fixed charges, then precalculates the interactions between sectors. The second, `isfc_calculate_field`, given a group of ions and a set of points, calculates the potential and field at the points.

## 5.6. Analytical method

The analytical method is a group of six modules which solve Poisson's equation for a toroidal boundary. The names of these modules are prefixed `an_` for identification. They can be used independently, via the subroutines in `an_main`. See Fig. 4.

The modules in the analytical method are not strictly abstract data types, since they operate on derived data types (records), whose elements can be accessed from outside the module. This was done for reasons of efficiency, to prevent the compiler from unnecessarily copying large arrays in an attempt to avoid aliasing. Nevertheless, they are still intended to be used as abstract data types, with all operations on the types, except for reading the data, being done by the routines of the owning module.

### 5.6.1. The `an_main` module

This module is the interface to the analytical method. It implements the algorithm described in Section 3.2, calling routines from the other analytical method modules to calculate the coefficients and function values. It has four interface routines: `an_set_parameters`, `an_get_parameters`, `an_specify_model`, and `an_calculate_field`. These perform the same functions as their equivalents in the `bics_channel` module, except that they are specific to the analytical method.

### 5.6.2. The `an_appfield` module

This module calculates the potential and field due to the charges induced on the toroidal boundary by the external (applied) field. Its interface consists of a derived data type and four subroutines. The data type, `afld_record`, holds all the arrays used to perform the calculation (as well as the results). This storage is allocated by the `afld_create` routine and deallocated by the `afld_destroy` routine. The `afld_calculate` routine performs the calculation, given the strength of the field, a set of points of interest, and a set of boundary coefficients from the `an_boundary` module. The `afld_print` routine dumps the contents of an `afld_record` to a given unit; it is used for debugging.

### 5.6.3. The `an_charge` module

This module calculates the coefficients described in Section 3.2.3, which depend only on the boundary and the position of the charges. Its interface works in the same way as that of the `an_appfield` module: it consists of a data type `achg_record`, and four routines `achg_create`, `achg_destroy`, `achg_calculate`, and `achg_print`. The `achg_calculate` routine requires a set of boundary coefficients and the position and magnitude of the charges.

#### 5.6.4. The *an\_point* module

This module calculates the coefficients which depend only on the position of the points of interest, and not the boundary or charges. Together with the coefficients from the *an\_charge* module, these allow *an\_main* to calculate the potential and field using Eq. (64). The module's interface works in the same way as those of *an\_appfield* and *an\_charge*; the prefix is *apnt*.

#### 5.6.5. The *an\_boundary* module

This module calculates the coefficients described in Section 3.2.2, which depend only on the boundary. It has three modes of operation, and *an\_main* uses it to generate three sets of coefficients: one for point charges inside the boundary (fixed charges), one for point charges outside the boundary (usually ions), and one for the external field. The module's interface works in the same way as those of *an\_appfield*, *an\_charge*, and *an\_point*; the prefix is *abdy*. The *abdy\_calculate* routine requires a description of the boundary and the mode of operation.

#### 5.6.6. The *an\_functions* module

This module contains a collection of routines which calculate the basic functions used by the other modules: the trigonometric functions, hyperbolic trigonometric functions, and the associated Legendre functions and their derivatives.

#### 5.7. The *bics\_basics* module

This module is used by all other components: it defines constants, type descriptors, and diagnostic functions.

## References

- [1] B. Hille, *Ionic Channels of Excitable Membranes* (Sinauer Associates, Sunderland, MA, 1992).
- [2] M. Hoyles, S. Kuyucak, S.H. Chung, *Biophys. J.* 70 (1996) 1628.
- [3] S. Kuyucak, M. Hoyles, S.H. Chung, *Biophys. J.* 74 (1998) 22.
- [4] S.C. Li, M. Hoyles, S. Kuyucak, S.H. Chung, *Biophys. J.* 74 (1998) 37.
- [5] D.G. Levitt, *Biophys. J.* 22 (1978) 209.
- [6] P.M. Morse, H. Feshbach, *Methods of Theoretical Physics*, Vol. II (McGraw-Hill, New York, 1953).
- [7] J.D. Love, *J. Math. Phys.* 13 (1972) 1297.
- [8] L.M. Milne-Thompson, *The Calculus of Finite Differences* (Macmillan, London, 1960).
- [9] M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions* (Dover, New York, 1965).
- [10] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes* (Cambridge Univ. Press, New York, 1992).
- [11] National Bureau of Standards, *Tables of Associated Legendre Functions* (Columbia Univ. Press, New York, 1945).
- [12] M. Metcalf, J. Reid, *FORTRAN 90/95 Explained* (Oxford Science Publications, Oxford, 1996).
- [13] D.F. Stubbs, N.W. Webre, *Data Structures* (Brooks/Cole Publishing Company, Monterey, CA, 1985).