Modeling Ionic Polymer-Metal Composites with Space-Time Adaptive Multimesh *hp*-FEM

D. Pugal[a,d], P. Solin[b,c,**], K. J. Kim[a,*], A. Aabloo[d]

[a] *Mechanical Engineering Department, University of Nevada, Reno, NV, U.S.A.*
[b] *Department of Mathematics and Statistics, University of Nevada, Reno, NV, U.S.A.*
[c] *Institute of Thermomechanics, Prague, Czech Republic*
[d] *Institute of Technology, Tartu University, Estonia*

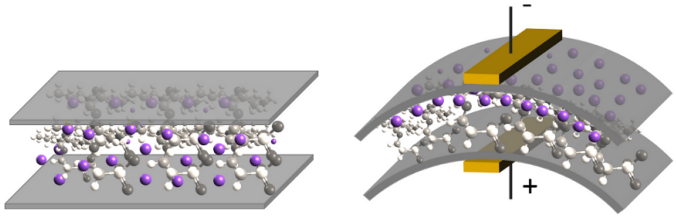
Abstract

In this paper we study a multiphysics coupled problem consisting of the Poisson and Nernst-Planck equations, that is used to model ionic polymer-metal composite (IPMC) materials. The problem is discretized using the finite element method (FEM) and solved using space-time adaptive *hp*-FEM algorithms provided by the Hermes library. Numerical results are presented and discussed, including comparison of various versions of adaptive *hp*-FEM and conventional low-order FEM. In particular, we discuss the performance of a novel adaptive multimesh *hp*-FEM algorithm that is part of the Hermes library.

Keywords: Ionic polymer-metal composites, IPMC, Nernst-Planck equation, Poisson equation, finite element method, space-time adaptive *hp*-FEM

1. Introduction

Ionic Polymer-Metal Composites (IPMC) have been studied during the past two decades for their potential to serve as noiseless mechanoelectrical and electromechanical transducers. The advantages of IPMC over other electroactive polymer actuators are low voltage bending, high strains ($> 1\%$), and an ability to work in wet environments. A typical IPMC consists of a thin sheet of polymer (often Nafion or Teflon) which is sandwiched between noble metal electrodes such as platinum or gold. When fabricated, the polymer membrane is saturated with certain solvent and ions such as water and H^+. When a voltage is applied to the electrodes, the counter ions start migrating due to the imposed electric field. By dragging along the solvent, the osmotic pressure difference near the electrodes results in bending of the material (see Fig. 1).



(a) (b)

Figure 1: Conceptual model of the actuation of IPMC. Initial counter ion distribution (a) and the distribution and resulting bending after applying a voltage (b).

*Corresponding author
**Principal corresponding author
Email addresses: david.pugal@gmail.com (D. Pugal), solin@unr.edu (P. Solin), kwangkim@unr.edu (K. J. Kim), alvo@ut.ee (A. Aabloo)

In this study we will model IPMC materials via a multiphysics coupled problem consisting of the Poisson and Nernst-Planck equations (abbreviated by PNP in the following). These equations are used to model charge transport in materials that includes ionic migration, diffusion, and convection. The charge transport process is a key mechanism for electromechanical transduction [1, 2, 3, 4, 5, 6, 7].

The PNP system is highly nonlinear and for a typical domain with two electrodes, largest differences in charge concentration occur in a very narrow region near the boundary. The computing power required for a full scale problem is significant. This is why we are interested in exploring adaptive algorithms – we hope to obtain meshes that are optimal in terms of calculation time and calculation error.

The Nernst-Planck equation for a mobile species — in our case for counter ions — has the form

$$\frac{\partial C}{\partial t} + \nabla \cdot (-D\nabla C - z\mu FC\nabla\phi) = 0. \tag{1}$$

Here C stands for the counter ion concentration, D is diffusion, μ mobility, F Faraday constant, ϕ voltage, and z the charge number. We have neglected the velocity of the species as in our case it can be assumed zero. The Poisson equation has the form

$$-\nabla^2\phi = \frac{F\rho}{\varepsilon} \tag{2}$$

where ε is the absolute dielectric permittivity. The charge density ρ is defined via

$$\rho = C - C_0 \tag{3}$$

where C_0 is a constant anion concentration.

The outline of the paper is as follows: Section 2 explains why it is extremely difficult to find a good mesh for the PNP problem, and hence why we are interested in exploring adaptive algorithms. The PNP model is presented in Section 3 where also its weak formulation is derived. Numerical results and comparisons are presented in Section 4, and conclusion and outlook are drawn in Section 5.

2. Motivation

Fig. 2 shows a typical solution for C and ϕ at $t = 0.1\ s$ and $t = 3.0\ s$ for a simplified one-dimensional problem. The corresponding solution constants are summarized below in Table 1.

The solution shown in Fig. 2 has two notable characteristics: For the most part of the domain Ω, the gradient $\nabla C = 0$. Close to $\partial\Omega_2$, ∇C is nonzero and moving in time, and ∇C is very large at $\partial\Omega_1$. At the same time, ϕ is a "nice" smooth function for the most part of Ω but it has a large gradient at $\partial\Omega_2$. This makes the choice of an optimal mesh extremely difficult. Even if the solution was stationary, an optimal mesh for C could never be optimal for ϕ and vice versa.

Furthermore, the shape of the solution in Fig. 2 suggests that the polynomial degree of finite elements in the middle of the domain Ω and near the boundaries $\partial\Omega_1$, $\partial\Omega_2$ should be different — large high-degree elements should be used in the middle of the domain while small low-degree ones should be used in the boundary layers. The qualitative differences in the solution components C and ϕ also suggest that using different meshes would be beneficial.

3. Model

The model for solving PNP system is implemented in Hermes. A rectangular 2D domain $\Omega \subset \mathbb{R}^2$ with boundaries $\partial\Omega_{1\ldots4} \subset \partial\Omega$ is considered (see Fig. 3). As there is no flow considered into or out from the domain and there is no constant concentration at the boundaries, zero Neumann boundary conditions were used for Eq. (1):

$$-D\frac{\partial C}{\partial n} - z\mu FC\frac{\partial\phi}{\partial n} = 0. \tag{4}$$

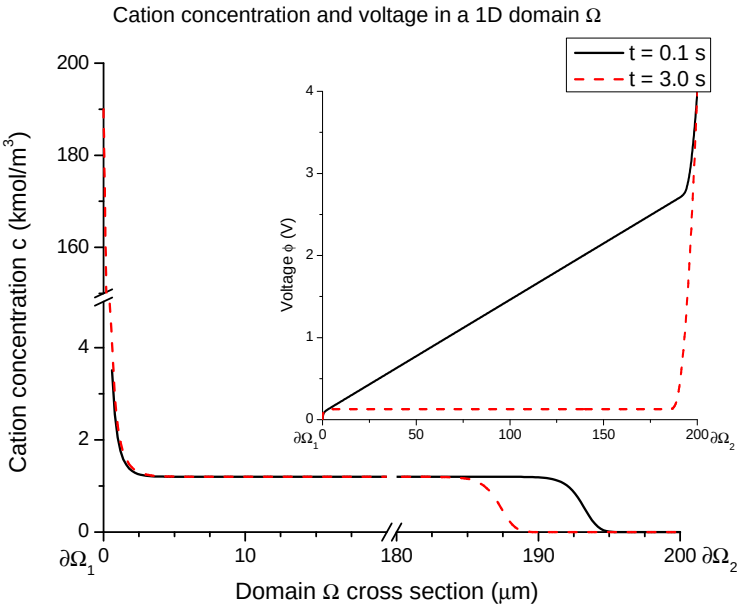


Figure 2: Sample concentration C and voltage ϕ in a 1D domain $\Omega \subset \mathbb{R}$. Dirichlet boundary conditions ($V_{\partial\Omega_1} = 0\ V$ and $V_{\partial\Omega_2} = 4\ V$) were applied to the Poisson equation (2) and Neumann conditions to the Nernst-Planck equation (1).

As a positive constant voltage V_{pos} was applied on Ω_1 and $V_{neg} = 0$ was applied on Ω_3, Dirichlet boundary conditions were used for Eq. (2) for boundaries Ω_1 and Ω_3:

$$\phi_{\partial\Omega_1} = V_{pos}, \tag{5}$$

$$\phi_{\partial\Omega_3} = 0, \tag{6}$$

and Neumann boundaries for Ω_2 and Ω_4:

$$\frac{\partial\phi_{\Omega_2}}{\partial n} = \frac{\partial\phi_{\Omega_4}}{\partial n} = 0. \tag{7}$$

Table 1: Constants used in the Poisson-Nernst-Planck system of equations.

Constant	Value	Unit	Description
D	10×10^{-11}	$\frac{m^2}{s}$	Diffusion constant
z	1	-	Charge number
F	96,485	$\frac{C}{mol}$	Faraday number
R	8.31	$\frac{J}{mol \cdot K}$	The gas constant
$\mu\ \left(= \frac{D}{RT}\right)$	4.11×10^{-14}	$\frac{s}{mol \cdot K}$	Mobility
C_0	1,200	$\frac{mol}{m^3}$	Anion concentration
ε	0.025	$\frac{F}{m}$	Electric permittivity

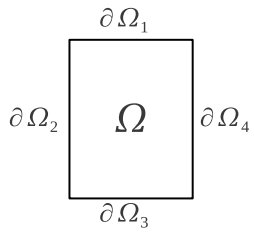


Figure 3: Calculation domain $\Omega \subset \mathbb{R}^2$ with boundaries $\partial\Omega_{1\ldots4} \subset \partial\Omega$.

3.1. Weak form of Poisson-Nernst-Planck system

The initial more implicit version of the derived weak form of PNP was derived in [8]. To make the derivation of the weak forms more convenient, the following constants are used:

$$K = z\mu F, \tag{8}$$

$$L = \frac{F}{\varepsilon}. \tag{9}$$

So Eq. (1) and Eq. (2) become after substituting Eq. (3) and the constants K and L:

$$\frac{\partial C}{\partial t} - D\nabla^2 C - K\nabla\cdot(C\nabla\phi) = 0, \tag{10}$$

$$-\nabla^2\phi - L\left(C - C_0\right) = 0. \tag{11}$$

The boundary condition Eq. (4) becomes:

$$-D\frac{\partial C}{\partial n} - KC\frac{\partial \phi}{\partial n} = 0. \tag{12}$$

The space for solution is $V = H^1(\Omega)$ where $H^1(\Omega) = \left\{v \in L^2(\Omega);\ \nabla v \in \left[L^2(\Omega)\right]^2\right\}$. Let's choose a test function $v^C \in V$. The weak form the Nernst-Planck equation Eq. (10) is found by multiplying it with the test function v^C and then integrating over the domain Ω:

$$\int_\Omega \frac{\partial C}{\partial t} v^C d\mathbf{x} - \int_\Omega D\nabla^2 C v^C d\mathbf{x} - \int_\Omega K\nabla C\cdot\nabla\phi v^C d\mathbf{x} - \int_\Omega KC\nabla^2\phi v^C d\mathbf{x} = 0. \tag{13}$$

After adding the weak form of the boundary term (Eq. (12)) and applying the Green's first identity to the terms that contain Laplacian ∇^2 we get

$$\int_\Omega \frac{\partial C}{\partial t} v^C d\mathbf{x} + D\int_\Omega \nabla C\cdot\nabla v^C d\mathbf{x} - K\int_\Omega \nabla C\cdot\nabla\phi v^C d\mathbf{x} + K\int_\Omega \nabla\left(Cv^C\right)\cdot\nabla\phi d\mathbf{x} \tag{14}$$

$$-D\int_{\partial\Omega}\frac{\partial C}{\partial n} v^C d\mathbf{S} - \int_{\partial\Omega} K\frac{\partial\phi}{\partial n} C v^C d\mathbf{S} = 0. \tag{15}$$

After expanding the nonlinear term and given that the boundary terms do not contribute, the weak form becomes

$$\int_\Omega \frac{\partial C}{\partial t} v^C d\mathbf{x} + D\int_\Omega \nabla C\cdot\nabla v^C d\mathbf{x} - K\int_\Omega \nabla C\cdot\nabla\phi v^C d\mathbf{x} + K\int_\Omega \nabla\phi\cdot\nabla C v^C d\mathbf{x} + K\int_\Omega C\left(\nabla\phi\cdot\nabla v^C\right) d\mathbf{x} = 0. \tag{16}$$

As the second and third term cancel out, the final weak from of the Nernst-Planck equation is

$$\int_\Omega \frac{\partial C}{\partial t} v^C d\mathbf{x} + D\int_\Omega \nabla C\cdot\nabla v^C d\mathbf{x} + K\int_\Omega C\left(\nabla\phi\cdot\nabla v^C\right) d\mathbf{x} = 0. \tag{17}$$

Similarly the weak form of Poisson equation Eq. 11 with a test function $v^\phi \subset V$ is:

$$-\int_\Omega \nabla^2\phi v^\phi d\mathbf{x} - \int_\Omega LCv^\phi d\mathbf{x} + \int_\Omega LC_0 v^\phi d\mathbf{x} = 0. \tag{18}$$

After expanding the ∇^2 terms, the final form becomes

$$\int_\Omega \nabla\phi \cdot \nabla v^\phi d\mathbf{x} - \int_\Omega LCv^\phi d\mathbf{x} + \int_\Omega LC_0 v^\phi d\mathbf{x} = 0. \tag{19}$$

3.2. Implementation in Hermes

To implement the system of equations Eq. (17) and Eq. (19), the residuals and the Jacobian matrix were derived. For that, Crank-Nicolson time stepping was used

$$\frac{\partial C}{\partial t} \approx \frac{C^{n+1} - C^n}{\tau}, \tag{20}$$

where τ is a time step. For the variables C^{n+1} and ϕ^{n+1} the following notation will be used:

$$C^{n+1} = \sum_{k=1}^{N^C} y_k^C v_k^C, j \tag{21}$$

$$\phi^{n+1} = \sum_{k=1}^{N^\phi} y_k^\phi v_k^\phi, \tag{22}$$

where v_k^C and v_k^ϕ are piecewise polynomial functions in V. Considering the Crank-Nicolson time stepping and the notation (21), the time discretized Eq. (17) becomes

$$\begin{aligned} F_i^C(Y) = & \int_\Omega \frac{C^{n+1}}{\tau} v_i^C d\mathbf{x} - \int_\Omega \frac{C^n}{\tau} v_i^C d\mathbf{x} \\ & + \frac{1}{2}\left[D\int_\Omega \nabla C^{n+1} \cdot \nabla v_i^C d\mathbf{x} + D\int_\Omega \nabla C^n \cdot \nabla v_i^C d\mathbf{x}\right] \\ & + \frac{1}{2}\left[K\int_\Omega C^{n+1}\left(\nabla\phi^{n+1} \cdot \nabla v_i^C\right) d\mathbf{x} + K\int_\Omega C^n\left(\nabla\phi^n \cdot \nabla v_i^C\right) d\mathbf{x}\right], \end{aligned} \tag{23}$$

and in the notation (22), Eq. (19) becomes

$$F_i^\phi(Y) = \int_\Omega \nabla\phi^{n+1} \cdot \nabla v_i^\phi d\mathbf{x} - \int_\Omega LC^{n+1} v_i^\phi d\mathbf{x} + \int_\Omega LC_0 v_i^\phi d\mathbf{x}. \tag{24}$$

For the implementation the 2×2 Jacobian matrix DF/DY with elements corresponding to

$$\frac{\partial F_i^C}{\partial y_j^C}, \frac{\partial F_i^C}{\partial y_j^\phi}, \frac{\partial F_i^\phi}{\partial y_j^C}, \frac{\partial F_i^\phi}{\partial y_j^\phi}, \tag{25}$$

must be derived:

$$\frac{\partial F_i^C}{\partial y_j^C} = \int_\Omega \frac{1}{\tau} v_j^C v_i^C d\mathbf{x} + \frac{1}{2} D\int_\Omega \nabla v_j^C \cdot \nabla v_i^C d\mathbf{x} + \frac{1}{2} K\int_\Omega v_j^C\left(\nabla\phi^{n+1} \cdot \nabla v_i^C\right) d\mathbf{x}, \tag{26}$$

$$\frac{\partial F_i^C}{\partial y_j^\phi} = \frac{1}{2} K\int_\Omega C^{n+1}\left(\nabla v_j^\phi \cdot \nabla v_i^C\right) d\mathbf{x}, \tag{27}$$

$$\frac{\partial F_i^\phi}{\partial y_j^C} = -\int_\Omega L v_j^C v_i^\phi d\mathbf{x}, \tag{28}$$

$$\frac{\partial F_i^\phi}{\partial y_j^\phi} = \int_\Omega \nabla v_j^\phi \cdot \nabla v_i^\phi d\mathbf{x}. \tag{29}$$

In Hermes Eq. (23) and (24) define the residuum F and Eq. (26)—(29) define the Jacobian matrix J.

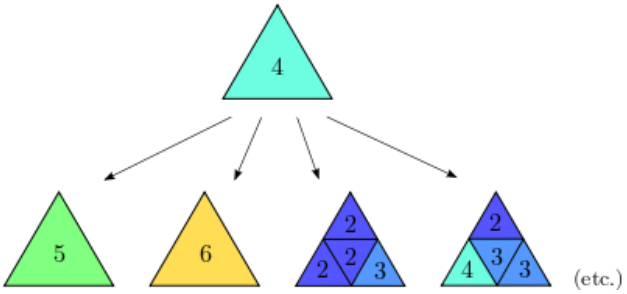


Figure 4: Refinement candidates for a fourth-order element.

3.3. Adaptive multi-mesh solution in Hermes

The defined Jacobian J and residuum F can be simply solved with Hermes by using Newton's iteration. However, we were more interested in the automatic adaptivity to control the error and problem size.

In a traditional low-order FEM, refining an element is not algorithmically complicated, and so the most difficult part is to find out what elements should be refined. To do this, various techniques ranging from rigorous guaranteed a-posteriori error estimates to heuristic criteria such as residual error indicators, error indicators based on steep gradients, etc are employed. Unfortunately, none of these approaches are suitable for real-life multiphysics coupled problems or higher-order finite element methods: rigorous guaranteed error estimates only exist for very simple problems (such as linear elliptic PDE), and moreover only for low-order finite elements (such as piecewise linear approximations). These heuristic techniques are not employed in Hermes since they may fail in non-standard situations, and because they lack a transparent relation to the true approximation error. In order to obtain fast, usable adaptivity, one has to resort to adaptive *hp*-FEM. Automatic adaptivity in the *hp*-FEM is substantially different from adaptivity in low-order FEM, since every element can be refined in many different ways. Fig. 4 shows several illustrative refinement candidates for a fourth-order element. The number of possible element refinements is implementation dependent. In general it is very low in h or p adaptivity, but much higher in *hp*-adaptivity, and it rises even more when anisotropic refinements are enabled. Hermes supports 8 different refinement modes, namely, 3 isotropic and 5 anisotropic refinements. The isotropic refinements are h-isotropic (H_ISO), p-isotropic (P_ISO), *hp*-isotropic (HP_ISO). Anisotropic refinement modes are h-anisotropic (H_ANISO), *hp*-anisotropic-h (HP_ANISO_H), p-anisotropic (P_ANISO), *hp*-anisotropic-p (HP_ANISO_P), and *hp*-anisotropic (HP_ANISO). Due to the large number of refinement options, classical error estimators that provide a constant error estimate per element, cannot be used to guide automatic *hp*-adaptivity. For this, one needs to know the shape of the approximation error. Hermes uses a pair of approximations with different orders of accuracy to obtain this information: coarse mesh solution and fine mesh solution. The initial coarse mesh is read from the mesh file, and the initial fine mesh is created through its global refinement both in h and p. The fine mesh solution is the approximation of interest both during the adaptive process and at the end of computation. The coarse mesh solution represents its low-order part. Here global orthogonal projection of the fine mesh solution on the coarse mesh was used instead instead of solving on the coarse mesh.

Refinement mode is selected by a user from among 8 modes listed before. Hermes selects a particular refinement scheme from several candidates based on the score which is calculated as follows

$$s = \frac{\log_{10} e_0 - \log_{10} e}{(d_0 - d)^{\xi}}, \tag{30}$$

where e and d are an estimated error and an estimated number of DOF of a candidate respectively, e_0 and d_0 are are an estimated error and an estimated number of DOF of the examined element, respectively, and ξ is a convergence exponent. Here candidate refers to a proposed refinement. Fig. 5 shows all the proposed refinements for all refinement modes in case of quad elements. More information about the refinement modes and technical details about the score calculation can be found in [9].

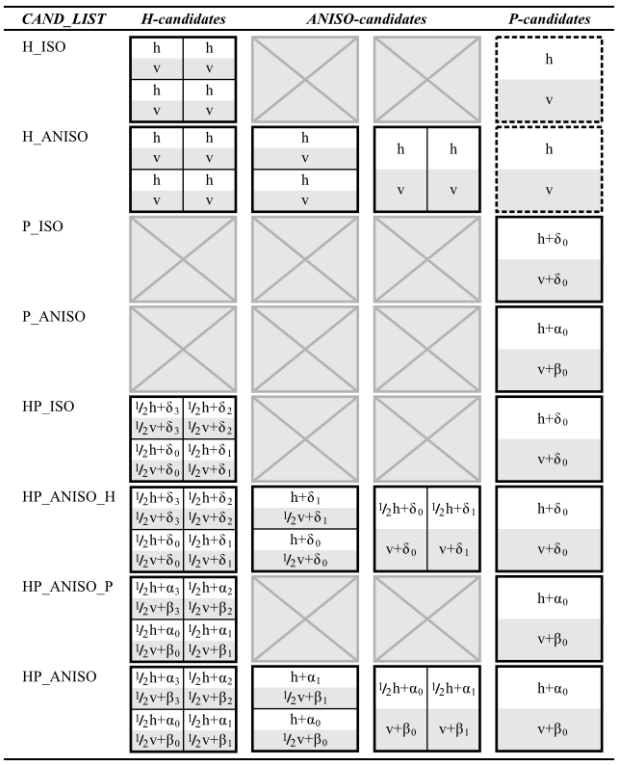


Figure 5: Refinement candidates for every refinement mode for quad type elements.

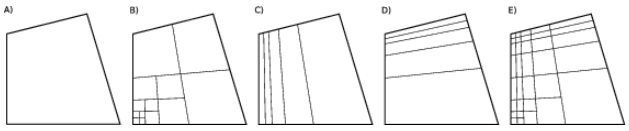


Figure 6: Master mesh (A) and three different meshes for a coupled problem (B) — (D), and the virtual union mesh (E).

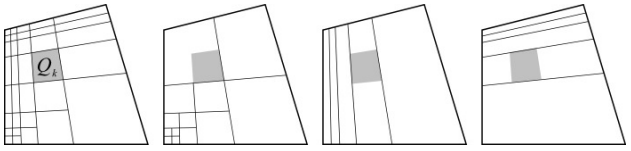


Figure 7: Integration over an element Q_k of the virtual union mesh and the appropriate subelements of the existing elements where this integration is performed.

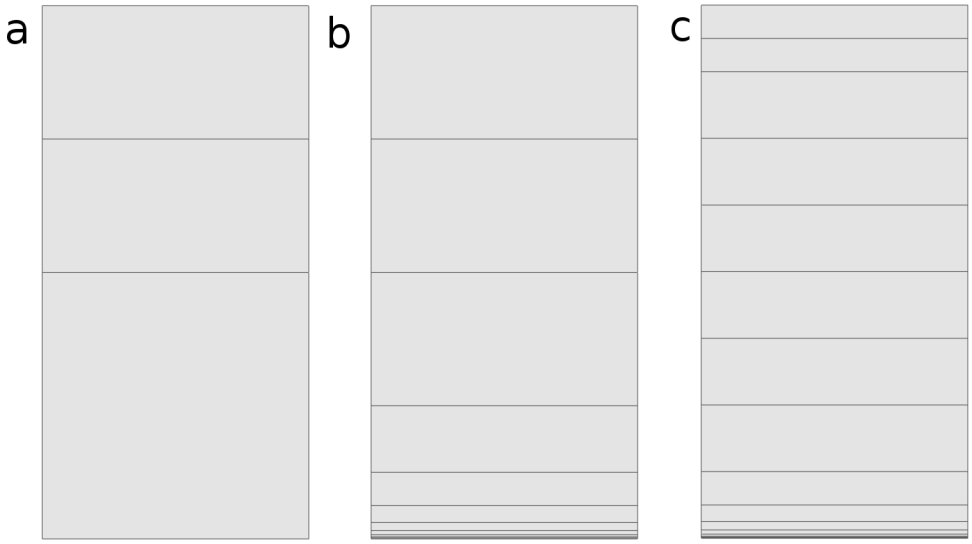


Figure 8: Initial coarse mesh (a), half refined mesh (b) and refined mesh (c). The coarse mesh and refined mesh were used in the initial calculations, the latter one in case of p-adaptivity (including HP_ANISO_P). The half-refined mesh was used later to optimize the hp-adaptive refinement solutions.

In multiphysics PDE systems such as Poisson-Nernst-Planck system it can happen that one physical field is very smooth whereas other one is not. This is well shown in Fig. 2. If all the fields are approximated on the same mesh, then the necessity to refine the mesh at the steep gradient of a field implies new degrees of freedom for the smooth fields as well. This can be very wasteful. Hermes makes it possible to approximate them on individual meshes. These meshes are not completely independent of each other — they have a common coarse mesh that we call master mesh. The master mesh is there for algorithmic purposes only, it may not even be used for discretization purposes: Every mesh in the system is obtained from it via an arbitrary sequence of elementary refinements. This is illustrated in Fig. 6, where (A) is the master mesh, (B) — (D) three different meshes (say, for a coupled problem with three equations), and (E) is the virtual union mesh that is used for assembling. The union mesh is not constructed physically in the computer memory — it merely serves as a hint to correctly transform the integration points while integrating over sub-elements of an elements of the existing meshes. Fig. 7 shows the integration over an element Q_k of the virtual union mesh. As a result, the multimesh discretization of the PDE system is monolithic in the sense that no physics is lost — all integrals in the discrete weak formulations are evaluated exactly up to the error in the numerical quadrature.

4. Results

Calculation for all the refinement modes were performed in both single-mesh and multi-mesh configurations. The following numerical results were recorded for each refinement mode: converged relative error, cumulative CPU time, and the problem size in terms of number of degrees of freedoms (NDOFs) at each time step. Two types of initial meshes were used — in case of only p-adaptivity, more refined mesh was used (Fig. 8 (c)) to ensure the error convergence. When the element size refinement was also enabled (all h/hp refinement modes), very coarse initial mesh was used (see Fig. 8 (a)) to let the adaptivity algorithm find the most optimal mesh. So coarse initial mesh might not be suitable for all the practical applications, as will be demonstrated in the end of this section, however, it provides a good insight into the adaptivity performance of Hermes. In both cases, the initial mesh was loaded at each time step of the calculation.

Example of the solution at $t = 0.7$ s and $t = 3.0$ s with refinement mode HP_ANISO is shown in Fig. 9. The time $t = 0.7$ s was chosen because by that time, some ionic migration has already taken place, i.e.

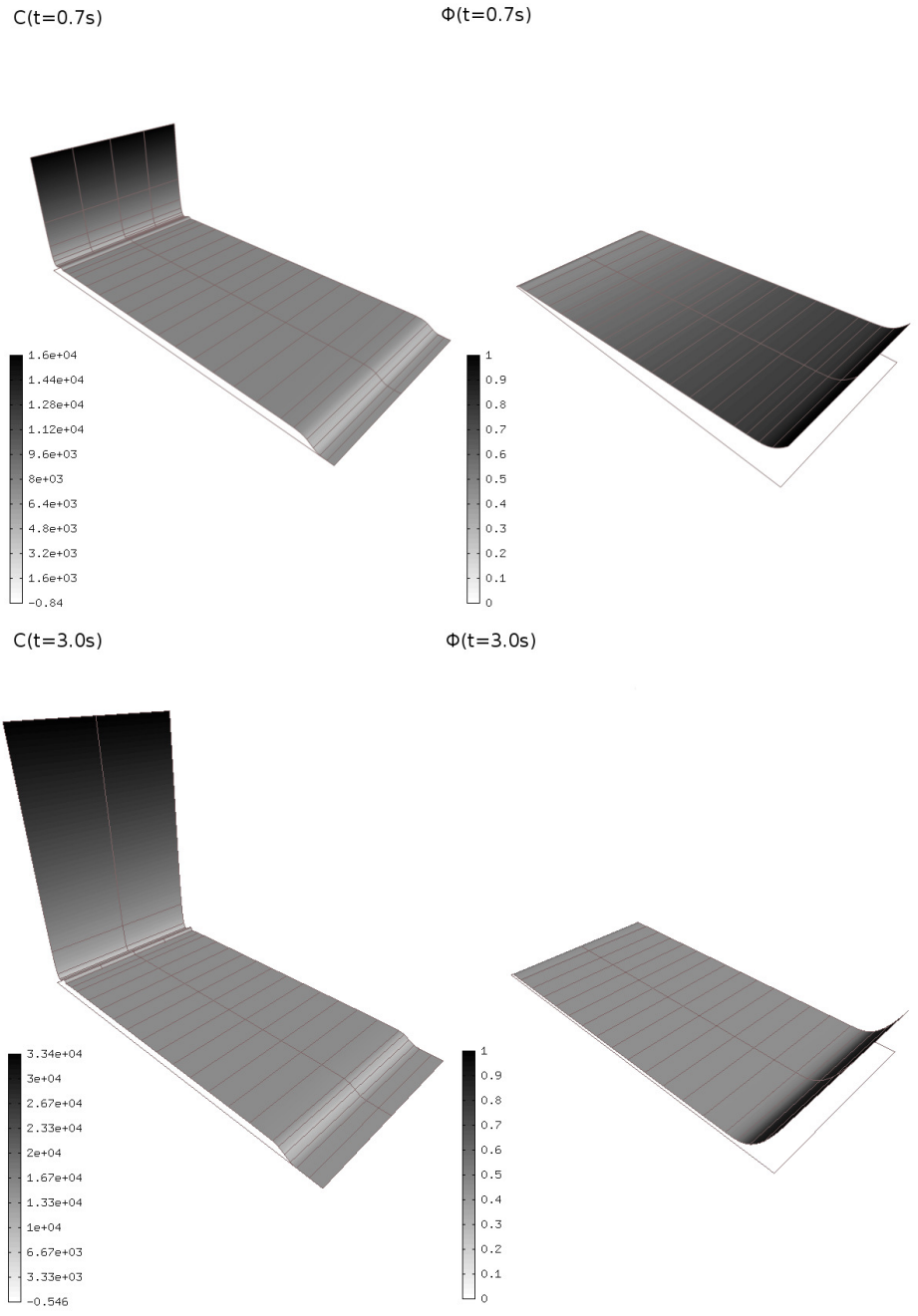


Figure 9: Concentration C and voltage ϕ at two different time steps (HP_ANISO adaptivity was used). This is a 2D solution shown as in 3D, where the height indicates the values of C and ϕ.

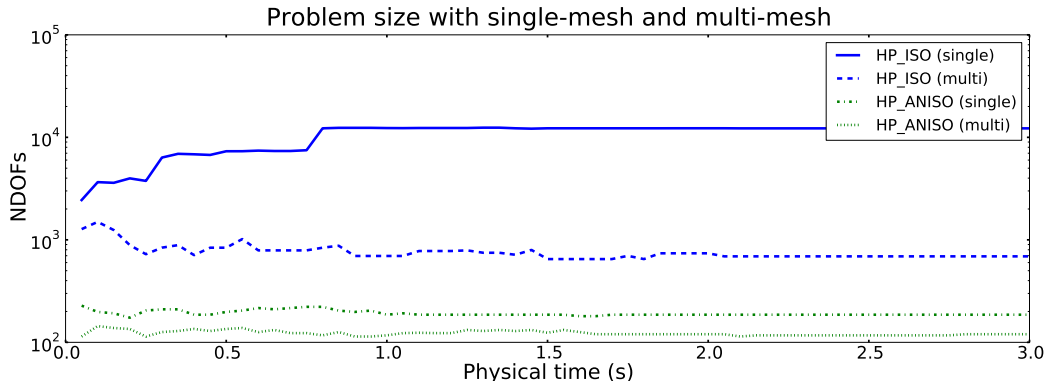


Figure 10: NDOFs in case of single-mesh and multi-mesh solutions with HP_ISO and HP_ANISO refinements. (Notice log Y scale)

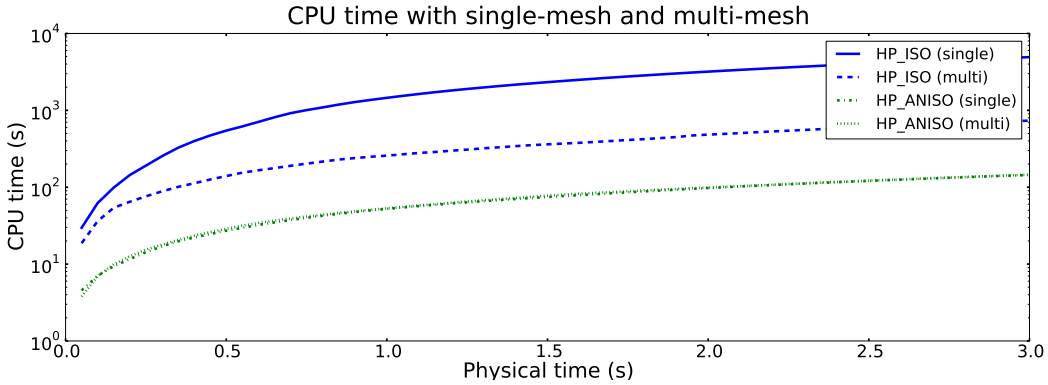


Figure 11: CPU times in case of single-mesh and multi-mesh solutions with HP_ISO and HP_ANISO refinements. (Notice log Y scale)

concentration gradient near the $\partial\Omega_1$ and $\partial\Omega_3$ has formed. The automatic mesh refinements at different time steps are clearly visible in the figure — especially near the top boundary, where the concentration gradient is moving in time (as was also seen in Fig. 2).

The following subsections provide a detailed comparison of the different refinement modes and try to estimate the most suitable one for the given problem.

4.1. Optimal refinement modes

Running the simulation with different refinement modes and meshes showed that the multi-mesh configuration generally results in a smaller problem, faster calculation, and better or similar error convergence compared to the single-mesh configuration. This is well illustrated in Fig. 10 and Fig. 11. The result can be understood from Fig. 2 — in the boundary regions $\partial\Omega_1$ and $\partial\Omega_3$ the concentration gradient is greater than voltage gradient, namely $\nabla C >> \nabla\phi$. Therefore refining the mesh for both variables is not reasonable in terms of number of degrees of freedom. For instance, the solution space with corresponding mesh in case of HP_ANISO at $t = 0.7\ s$ is shown in Fig. 9. The corresponding polynomial degree space is shown in Fig. 12. Notice that the adaptive algorithm has increased the maximum polynomial degree for C space to 7, however, at the same time, the maximum polynomial degree for the ϕ space is one. Furthermore, the mesh is significantly more refined for C. For most cases using the multi-mesh results in similar or better CPU time, i.e. multi-mesh takes less computational resources. Only excpetion was HP_ANISO_H refinement

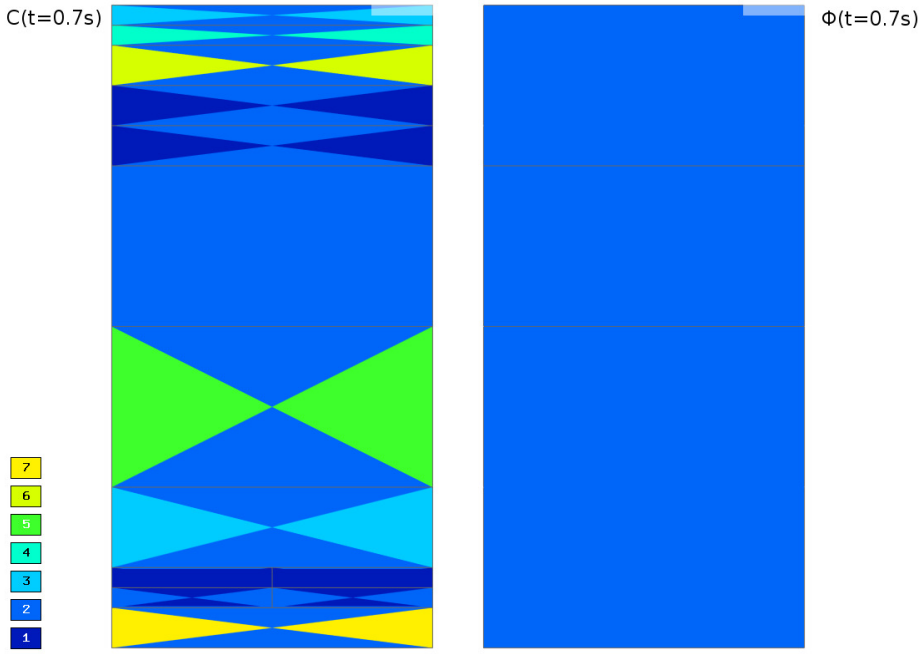


Figure 12: Polynomial degree space for C and ϕ at $t = 0.7\ s$ The color indicates the maximum polynomial degree of the corresponding element.

mode for which the single mesh configuration resulted slightly faster calculation time. Based on the results, only multi-mesh configurations will be considered in the following comparisons.

To narrow down the list of refinement modes for the given problem, first the isotropic and anisotropic adaptivities were compared. The p-adaptivity modes were compared separately as they used more refined mesh (Fig. 8 (c)). Fig. 13 and Fig. 14 show the comparison of H_ISO, H_ANISO, HP_ISO, HP_ANISO, and HP_ANISO_H modes in terms of CPU time and problem size. Fig. 15 and Fig. 16 show the similar comparsion for the P_ISO, P_ANISO, and HP_ANISO_P modes. It can be clearly seen that the anisotropic refinement modes result in a reasonable problem size and the problems solve within a reasonable calculation time. It is interesting to note that HP_ANISO results in the smallest problem size. Also, in p-adaptivity group, HP_ANISO_P results in the smallest problem size at each time step, whereas P_ISO and P_ANISO have a very largy problem size during the first time steps of the solution. Here the term "reasonable problem size" means that the number of degrees of freedom in time converges to so that $N_{dof} < 500$, and the term "reasonable calculation time" means that the calculation (step $\tau = 0.01\ s$, physical time $t_{end} = 3.0\ s$) time t on a givem system was $t < 500\ s$. Although these parameters are empirical, they serve as an upper limit, given that the most refinement modes give significantly smaller results: $t << 500\ s$ and $N_{dof} << 500$.

4.2. Quantitative analysis of the refinement modes

Based on the results in the previous subsection, H_ANISO, HP_ANISO, and HP_ANISO_H from the *hp/h*-adaptivity group and HP_ANISO_P from p-adaptivity group will be compared in terms of problem size and cumulative CPU time. In all of the cases, the relative error at each time step remained below the threshold which were set to $e_{th} = 0.5\%$ between the coarse mesh and fine mesh solutions, therefore the error-time plot will not be considered.

Fig. 17 shows the cumulative CPU time for different refinement modes at each time step. All the calculations were done on the same computer. Here we see that HP_ANISO_H and HP_ANISO require the most resources which can be understood from the fact that these refinement modes have the largest number of candidates (see Section. 3) from which the refinement method is chosen. At the same time, HP_ANISO_P

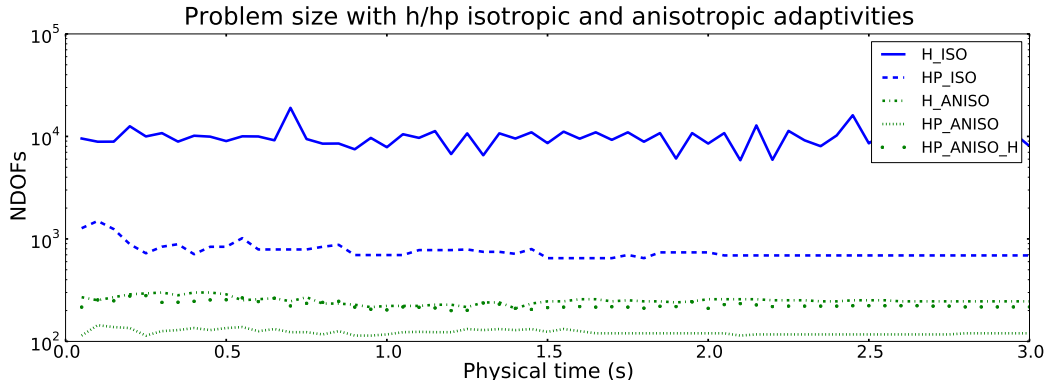


Figure 13: NDOFs in case of multi-mesh solutions with H_ISO, H_ANISO, HP_ISO, HP_ANISO, and single-mesh solution with HP_ANISO_H refinement modes. (Notice log Y scale)

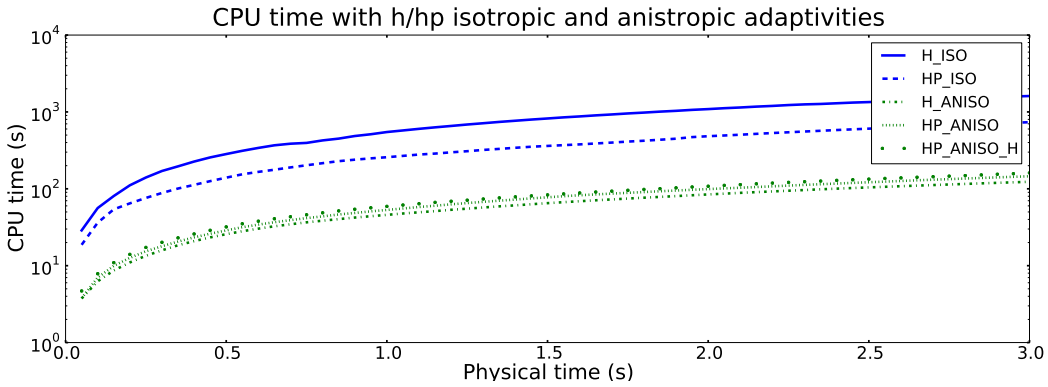


Figure 14: CPU times in case of multi-mesh solutions with H_ISO, H_ANISO, HP_ISO, HP_ANISO, and single-mesh solution with HP_ANISO_H refinement modes. (notice log Y scale)

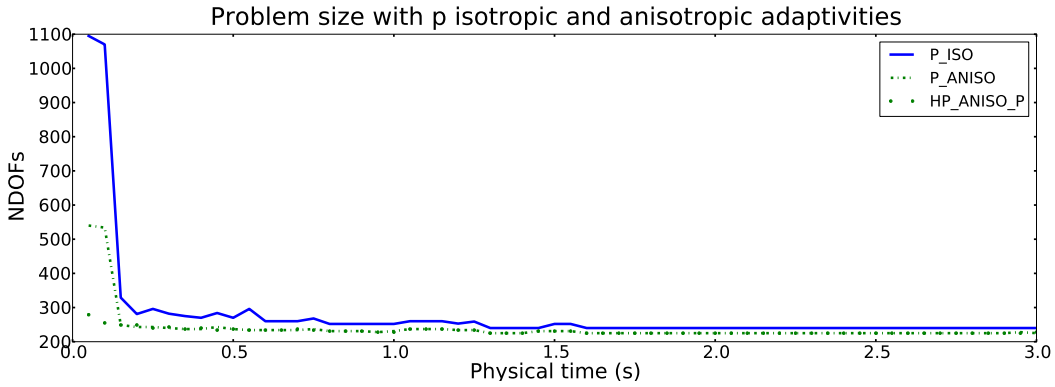


Figure 15: NDOFs in case of multi-mesh solutions with P_ISO, P_ANISO, and HP_ANISO_P refinement modes.

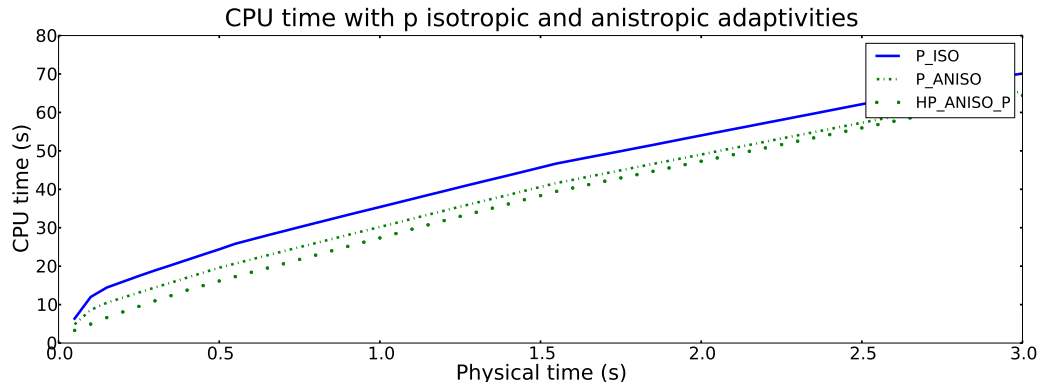


Figure 16: CPU times in case of multi-mesh solutions with P_ISO, P_ANISO, and HP_ANISO_P refinement modes.

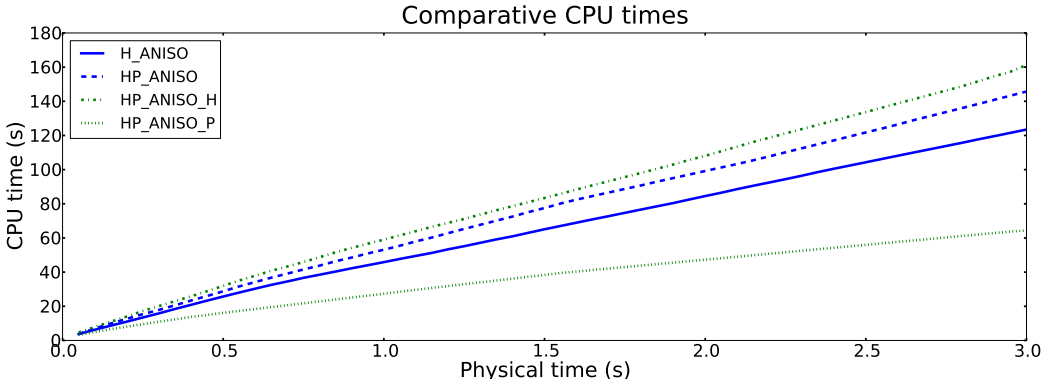


Figure 17: Comparative CPU time for different refinement modes.

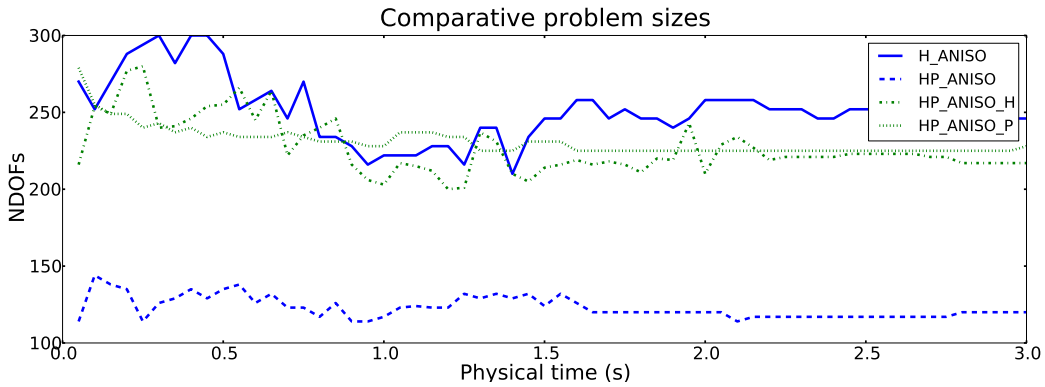


Figure 18: Comparative NDOFs at each time step for different refinement modes.

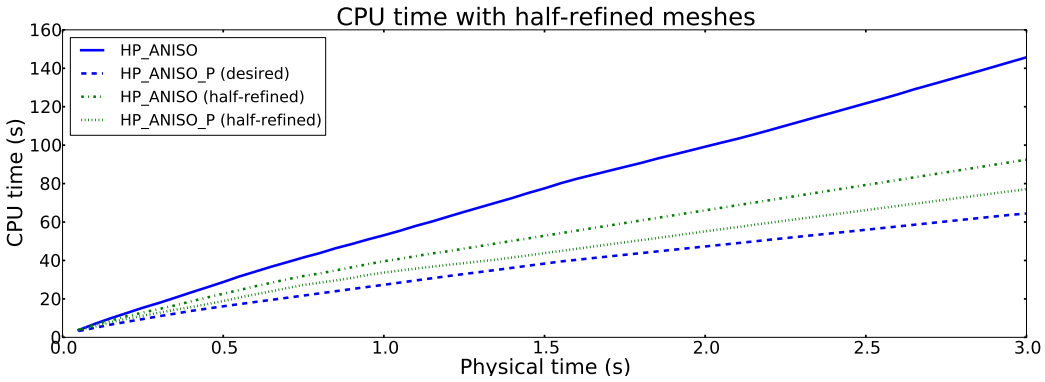


Figure 19: Cumulative CPU time for HP_ANISO and HP_ANISO_P with different initial meshes.

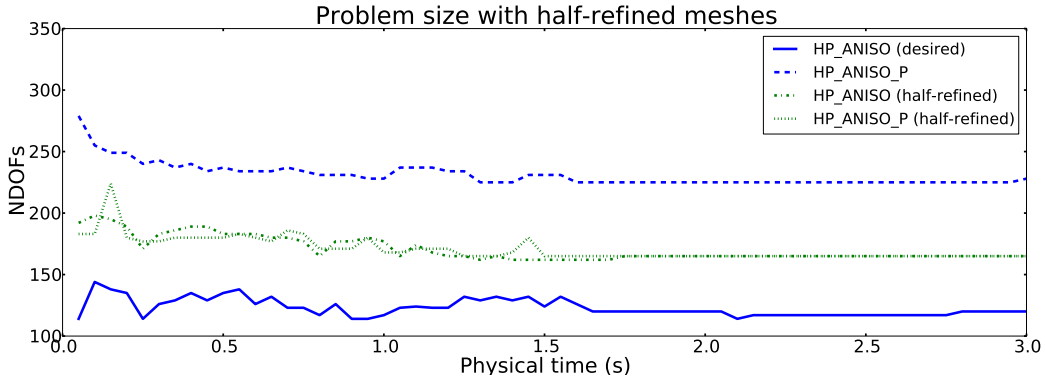


Figure 20: NDOFs at each time step for HP_ANISO and HP_ANISO_P with different meshes.

is the fastest among all the other refinement modes. Fig. 18 shows the NDOFs at each time step. It can be seen that the HP_ANISO results in the smallest problem size — $N_{dof} \approx 125$. All the other refinement modes result in a problem size of approximately ($N_{dof} \approx 250$). So we have two notable refinement modes — HP_ANISO_P because of the fast solution, and HP_ANISO because of the small problem size. In the following subsection, optimization of those two modes will be considered.

4.3. Optimizations of HP_ANISO and HP_ANISO_P refinement modes

So far we have seen that HP_ANISO results in the smallest problem size, but requires quite a lot CPU time compared to HP_ANISO_P that is the fastest. When it comes to a large domain or 3D modeling the problem size becomes the most important factor. Therefore, we consider HP_ANISO the most suitable refinement mode to the given problem. Thus some ways to optimize the time factor will be considered. The desirable output would be HP_ANISO problem size close to HP_ANISO_P CPU time. One way to optimize the problem is to choose somewhat more refined initial mesh. Other way one could think to optimize the problem would be to change the refinement frequency during the solving process. Recall that up to this point, the initial mesh has been loaded in the beginning of each time step. True, by employing the optimizations, we already must know something about the problem and its solution beforehand. However, it could still be practial when solving a real problem in a large domain.

The problem size and CPU time with HP_ANISO and HP_ANISO_P adaptivities on more refined initial mesh (see Fig. 8 (c)) compared to the coarse initial mesh (Fig. 8 (a)) and HP_ANISO_P solution is shown in Fig. 19 and Fig. 20. By using initially more refined mesh, the problem solving time can be reduced in case of

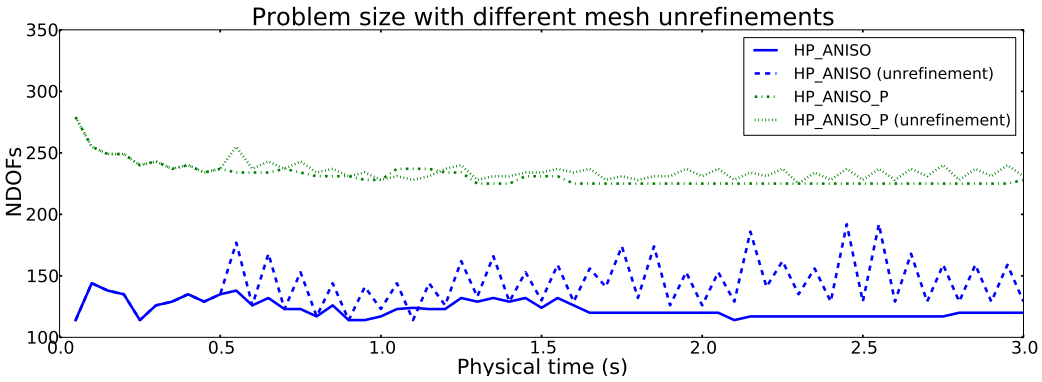


Figure 21: NDOFs at each time step for HP_ANISO and HP_ANISO_P with mesh unrefinement at each time step and at over each time step after first 0.5 s physical solution time.

HP_ANISO, at the same time, the problem size increases. In this situation, HP_ANISO_P and HP_ANISO perform equally well.

The next proposed optimization involved changing the unrefinement frequency. It is known that the concentration gradient ∇C changes the most in the initial phase of the calculation, therefore, the unrefinement after each time step was performed until $t = 0.5$ s (physical time). After that, the unrefinement was performed in $\Delta t = 0.10$ s interval. However, this optimization did not result in a stable solution, i.e. the problem size does not remain steady, but starts to oscillate depending on the unrefinement frequency. This is shown in Fig. 21.

Therefore varying the unrefinement frequency will not likely result in desired results in real applications for given system of equation. At the same time, by varying a mesh size, optimal initial mesh could be found for both HP_ANISO and HP_ANISO_P refinement modes.

4.4. More general results

Based on the results, cation concentration and voltage was calculated for different boundary conditions. For instance, when voltage is applied as follows

$$\phi_{\Omega_1} = 0.5\frac{x}{width_{\Omega_1}} + 0.5, \tag{31}$$

the concentration gradient ∇C and the voltage gradient $\nabla\phi$ are no longer effectively 1D. The calculated C and ϕ in Ω and corresponding meshes and polynomial degrees of the elements are shown in Fig. 22. HP_ANISO refinement mode was used. Notice that the solution is different to the one in Fig. 9 and the adapted mesh and the polynomial degrees are also more complicated than in Fig. 12. It must be noted that in case of non uniform boundary conditions which results in 2D problem, refined initial mesh was more efficient to use.

PAVEL: Should I try to include the similar solution from comsol and do some comparison here? I got this idea just before commiting the final draft to you?

5. Conclusion and Outlook

In this work the system of Nernst-Planck-Poisson equations was solved using *hp*-finite element method with adaptive multimesh configuration. The weak form, residuals and the Jacobian matrix of the system were explicitly derived and implemented in Hermes *hp*-FEM adaptive solver. The solution for Nernst-Planck-Poisson problem with two field variables C and ϕ results in very different field gradients in the space and time. Conventional finite element solvers do not provide the means how to deal with such problems such

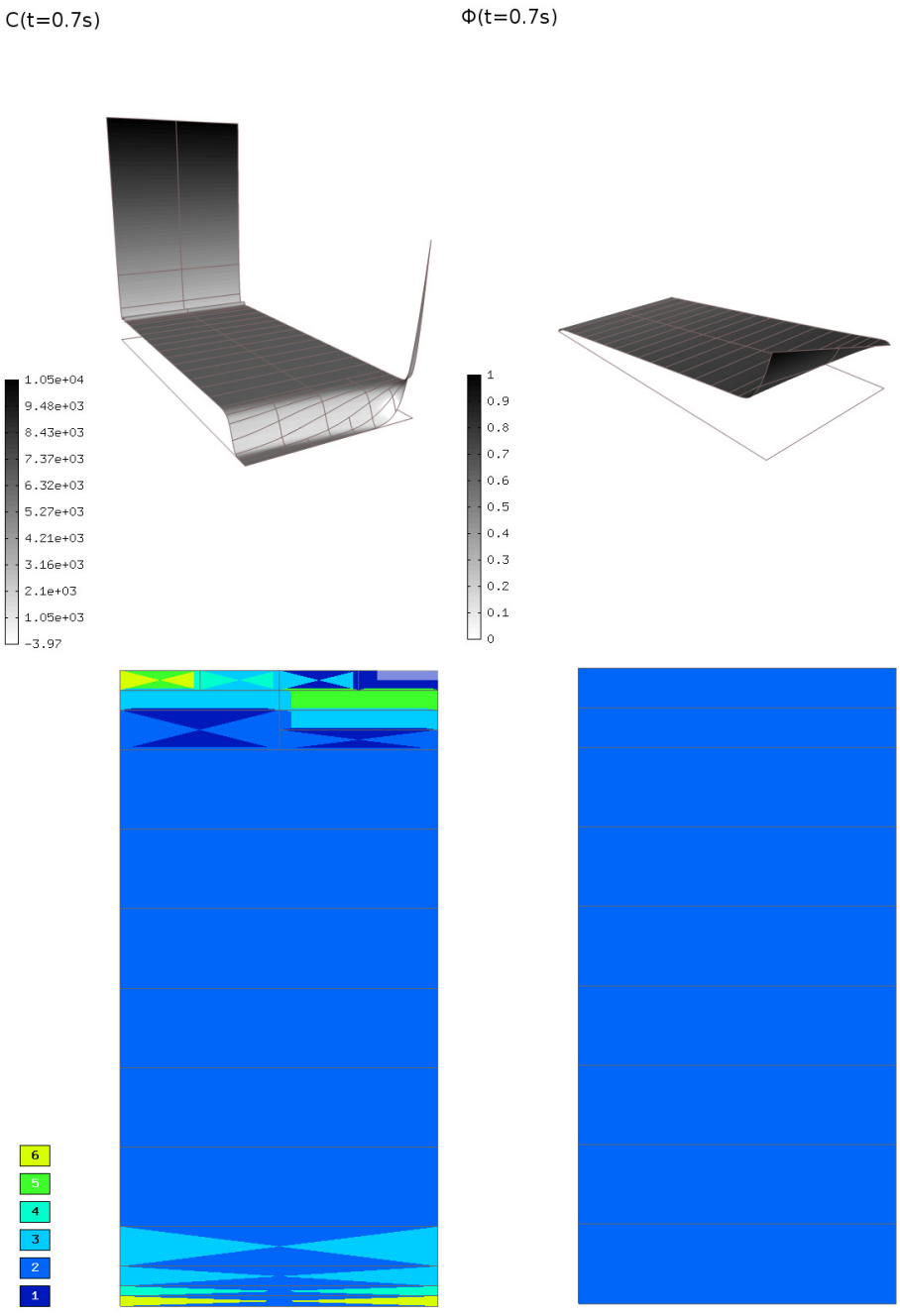


Figure 22: Solutions C and ϕ and corresponding polynomial degrees of the elements at $t = 0.7\ s$. HP_ANISO refinement mode was used. The height in the solution graphs indicates the value.

that both the error of the solution and problem size remain small throughout the time dependent solving process.

It was shown that using time dependent adaptivity, multi-mesh configuration, and anisotropic *hp* refinements, the problem size remains acceptably small throughout the solving process. Namely, hermes refinement modes HP_ANISO and HP_ANISO_P resulted the smallest and fastest problem, respectively. At the same time, relative error of the solution was known. Furthermore, using the multi-mesh configuration for the variables C and ϕ was justified — the adaptivity algorithm did not refine the mesh of ϕ nor did increase the polynomial degree throughout the adaptivity process. However, the mesh was significantly refined for C and also the maximum polynomial degree was varied in the range of $2 \dots 9$. So it is efficient to use multi-mesh in terms of the number of degrees of freedom.

Conclusively, by using *hp*-FEM with adaptive multi-mesh configuration we can possibly reduce the problem size of the Nernst-Planck-Poisson equation system significantly while still maintaining prescribed precision of the solution. We believe, and this is yet to be demonstrated, that this becomes especially important when dealing with 3D problems with a large physical domain.

Acknowledgments

The second author was partially supported by the Grant Agency of the Academy of Sciences of the Czech Republic under Grant No. IAA100760702, and by the U.S. Department of Energy Research Subcontract No. 00089911. The third author acknowledges the financial support of the U.S. Office of Naval Research under Award N000140910218.

[1] S. Basu, M. Sharma, An improved space-charge model for flow through charged microporous membranes, Journal of Membrane Science 124 (1) (1997) 77–91.

[2] M. Shahinpoor, K. J. Kim, Ionic polymer-metal composites: I. fundamentals, Smart Materials and Structures 10 (4) (2001) 819.

[3] S. Nemat-Nasser, Micromechanics of actuation of ionic polymer-metal composites, Journal of Applied Physics 92 (5) (2002) 2899–2915.

[4] K. Newbury, D. Leo, Linear electromechanical model of ionic polymer transducers-Part I: Model Development, Journal of Intelligent Material Systems and Structures 14 (6) (2003) 333.

[5] T. Wallmersperger, D. J. Leo, C. S. Kothera, Transport modeling in ionomeric polymer transducers and its relationship to electromechanical coupling, Journal of Applied Physics 101 (2) (2007) 024912.

[6] D. Pugal, K. J. Kim, A. Punning, H. Kasemagi, M. Kruusmaa, A. Aabloo, A self-oscillating ionic polymer-metal composite bending actuator, Journal of Applied Physics 103 (8) (2008) 084908.

[7] D. Pugal, K. Jung, A. Aabloo, K. Kim, Ionic polymer-metal composite mechanoelectrical transduction: review and perspectives, Polymer international 59 (3) (2010) 279–289.

[8] D. Pugal, S. J. Kim, K. J. Kim, K. K. Leang, Ipmc: recent progress in modeling, manufacturing, and new applications, Vol. 7642, SPIE, 2010, p. 76420U.

[9] P. Solin, et al., Hermes - Higher-Order Modular Finite Element System (User's Guide).
URL `http://hpfem.org`