Commun. Comput. Phys.
doi: 10.4208/cicp.270713.280214a

Vol. **16**, No. 2, pp. 491-515
August 2014

An Iterative Discontinuous Galerkin Method for Solving the Nonlinear Poisson Boltzmann Equation

Peimeng Yin[1], Yunqing Huang[1] and Hailiang Liu[2,*]

[1] *Hunan Key Laboratory for Computation and Simulation in Science and Engineering; School of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, P.R. China.*
[2] *Iowa State University, Mathematics Department, Ames, IA 50011, USA.*

Received 27 July 2013; Accepted (in revised version) 28 February 2014

Available online 12 June 2014

Abstract. An iterative discontinuous Galerkin (DG) method is proposed to solve the nonlinear Poisson Boltzmann (PB) equation. We first identify a function space in which the solution of the nonlinear PB equation is iteratively approximated through a series of linear PB equations, while an appropriate initial guess and a suitable iterative parameter are selected so that the solutions of linear PB equations are monotone within the identified solution space. For the spatial discretization we apply the direct discontinuous Galerkin method to those linear PB equations. More precisely, we use one initial guess when the Debye parameter $\lambda = \mathcal{O}(1)$, and a special initial guess for $\lambda \ll 1$ to ensure convergence. The iterative parameter is carefully chosen to guarantee the existence, uniqueness, and convergence of the iteration. In particular, iteration steps can be reduced for a variable iterative parameter. Both one and two-dimensional numerical results are carried out to demonstrate both accuracy and capacity of the iterative DG method for both cases of $\lambda = \mathcal{O}(1)$ and $\lambda \ll 1$. The $(m+1)$th order of accuracy for L^2 and mth order of accuracy for H^1 for P^m elements are numerically obtained.

AMS subject classifications: 65D15, 65N30, 35J05, 35J25

Key words: Poisson-Boltzmann equation, nonlinear, existence, uniqueness, DDG methods, numerical flux.

1 Introduction

In this paper, we propose an iterative discontinuous Galerkin method to solve the nonlinear Poisson Boltzmann (PB, for short) equation, following the direct discontinuous

*Corresponding author. *Email addresses:* pemyin@gmail.com (P. Yin), huangyq@xtu.edu.cn (Y. Huang), hliu@iastate.edu (H. Liu)

Galerkin (DDG) method introduced by Liu and Yan [12] for parabolic equations, and then further developed by Huang et al. [13] for linear elliptic equations.

We restrict ourselves to the following nonlinear Poisson Boltzmann model,

$$-\lambda^2\Delta u = f(x)+e^{-u}, \quad \text{in } \Omega, \tag{1.1a}$$

$$u = g(x), \quad \text{on } \partial\Omega, \tag{1.1b}$$

where Ω is a convex bounded domain in $\mathbb{R}^d (d=1,2)$ with smooth boundary $\partial\Omega$, $\lambda>0$ is a physical parameter, $f(x)$,$g(x)$ are given functions.

The PB equation (1.1a) arises in many applications in physics, biology and chemistry. It was first introduced by Debye and Hückel [2] almost a century ago and further developed by Kirkwood [3]. In past twenty years a growing interest in this model has been driven mainly by numerical and experimental advances. Two examples of applications are particularly worth mentioning: the PB continuum electrostatic model has been widely accepted as a tool in theoretical studies of interactions of biomolecules such as proteins and DNAs in aqueous solutions, see e.g., [9]. The PB equation has also been used as a standard tool in modeling the electrostatic potential in plasma physics, see e.g., [10], where an asymptotic preserving numerical method is proposed to compute the PB equation arising in plasma physics.

There are two main challenges in numerically solving the PB problem (1.1), one is the nonlinear term e^{-u}, which requires some iteration techniques, instead of a direct discretization by standard methods. The other is the smallness of the parameter $\lambda \ll 1$, which needs to be properly resolved to maintain the approximation accuracy. Several numerical techniques have been applied to solve the PB equation, such as finite difference methods [4, 6, 7, 10, 15, 16], boundary element methods [9], multigrid methods [14] and finite element methods [1, 8].

The discontinuous Galerkin (DG) method we discuss in this paper is a class of finite element methods, using a completely discontinuous piecewise polynomial space for the numerical solution and the test functions. The flexibility of the DG method is afforded by local approximation spaces combined with the suitable design of numerical fluxes crossing cell interfaces, leading to several obvious advantages such as high order accuracy, flexibility in hp-adaptation, capacity to handle the domain with complex geometry, over the usual continuous Galerkin method even for elliptic problems. Indeed our primary motivation of considering the PB equation is to extend the recent developed direct DG method [11, 12] to nonlinear elliptic problems. Our strategy is to couple an iteration with the DDG spatial discretization. The iteration techniques have been exploited by many authors to prove the existence of solutions to nonlinear elliptic equations, see e.g., [17, 18]. The monotone iteration can also be used as a numerical method to approximate the solution, see e.g., [19]. However, when the parameter $\lambda \ll 1$, solving (1.1) by a monotone iteration with a constant iterative parameter may be inappropriate, as evidenced by our numerical experiments.

The iterative DG method is based on a series of linear PB equations with a variable iterative parameter. Through a careful analysis we identify an appropriate iterative pa-

rameter k^n in each step to ensure convergence of the iteration. Indeed a numerical comparison shows that the iteration with a variable iterative parameter is more efficient. In addition, we propose two initial guesses, in order to better handle two cases of $\lambda \ll 1$ and $\lambda = \mathcal{O}(1)$, respectively. Indeed, the iterative DG method presented in this paper offers an effective and accurate algorithm for solving the PB equation. Illustration of this is the main objective of this work.

In some applications employing the PB equation, say, in the study of the electrostatic field of biomolecular systems, the domain is the whole space consisting of a molecular region and a solvent region. In such cases a practical approach is to transfer the original equation posed on the whole space to a truncated domain using an artificial boundary condition, which is usually taken from an approximate analytical solution, see e.g. [8, Section 3]. With such a reformulation, our analysis and results can be easily generalized to this case as well.

We also point out that computational efficiency is a main concern when employing the PB equation to simulate large molecular structures in modern biophysical processes. There exist some successful iterative approaches, e.g., the package APBS (Adaptive Poisson-Boltzmann Solver), by Baker et al [5], using the (algebraic) multigrid type iterations. As illustrated in this work, the iterative DG method has linear complexity in terms of the degree of freedom even for small λ and enjoys all nice features of DG methods, therefore worth further investigation.

The rest of the paper is organized as follows: in Section 2, we describe the formulation of a series of linear PB equations, followed by a discussion of how to prepare initial guess u^0, and how to choose k^n such that the solutions of the linear PB equations converge to that of the nonlinear PB equation. In Section 3, we utilize the DDG method to discretize these linear PB equations, and discuss the existence, uniqueness and stability of the solutions of every discretized scheme, and the corresponding set of flux formulae is also derived. In Section 4, numerical examples of both one and two dimensions with rectangular and triangular meshes are provided. Finally, in Section 5, conclusions are given.

2 Iteration scheme

In this section, we present a method of iteration to approximate the solution of nonlinear PB equation (1.1).

2.1 Weak formulation and initial guess

In order to conquer the nonlinear difficulty, we introduce the following iteration to approximate (1.1): starting with an initial guess u^0, we find u^n $(n = 1,2,\cdots)$ iteratively by

solving the following linear PB equation

$$\begin{cases} -\lambda^2\Delta u^{n+1}+k^n u^{n+1}=k^n u^n+f(x)+e^{-u^n}, & \text{in } \Omega, \\ u^{n+1}=g(x), & \text{on } \partial\Omega, \end{cases} \tag{2.1}$$

where $g(x)\in H^{3/2}(\partial\Omega)$, $f(x)\in L^2(\Omega)$ is the given function, and $k^n\geq 0$ is to be determined for convergence.

We first show how to prepare u^0.

(i) If $\lambda=\mathcal{O}(1)$, we take

$$u_1^0=w, \tag{2.2}$$

where w solves

$$\begin{cases} -\lambda^2\Delta w=f(x), & \text{in } \Omega, \\ w=g(x), & \text{on } \partial\Omega. \end{cases} \tag{2.3}$$

Our numerical simulation shows that this choice is efficient when $\lambda=\mathcal{O}(1)$.

(ii) If $\lambda\ll 1$, there are two cases: (a) If $f(x)\geq 0$, $x\in\Omega$, we also take $u^0=u_1^0$ as the initial guess. However, for other cases, this choice can lead to divergence of iteration (2.1). We thus consider a second choice for u^0. (b) If there exists $x_0\in\Omega$, such that $f(x_0)<0$, then we take the following initial guess

$$u_2^0=\min\left\{\ln\left(\frac{1}{\text{esssup}(-f_0)}\right),\text{essinf}(g(x))\right\}, \quad x\in\Omega, \tag{2.4}$$

where $f_0=\frac{f-|f|}{2}$.

From now on we use $H_g^1(\Omega)$ to denote the space of $H^1(\Omega)$ with $v=g(x)$ *on* $\partial\Omega$. For weak solutions, it is suggested to adopt the weak formulation of (1.1): find $u\in H_g^1(\Omega)$, such that

$$A(u,v)=(f(x),v)+(e^{-u},v), \quad \forall v\in H_0^1(\Omega), \tag{2.5}$$

where $(u,v)=\int_\Omega uvdx$ denotes the inner product in the L^2 space, and

$$A(u,v)=(\lambda^2\nabla u,\nabla v)=\int_\Omega \lambda^2\nabla u\cdot\nabla v dx,$$

denotes the bilinear operator generated from the diffusion. With Holst's theory in [8], one can show that Eq. (2.5) admits a unique solution $u\in M$, where

$$M:=\{v|\ v\in H^1(\Omega),\ e^{-v}\in L^\infty(\Omega),\ v=g(x) \text{ on } \partial\Omega\}.$$

The weak form of (2.1) is: from $u^n\in M$, we find $u^{n+1}\in H_g^1(\Omega)$ such that

$$A(u^{n+1},v)+k^n(u^{n+1},v)=k^n(u^n,v)+(f(x),v)+(e^{-u^n},v), \quad \forall v\in H_0^1(\Omega). \tag{2.6}$$

The monotone iterative scheme is a standard approach to deal with nonlinear PDEs; we present a self-contained justification by some standard arguments.

Regarding the proposed initial guesses, we assert the following.

Lemma 2.1. *$u_i^0 \in M$ and $u_i^0 \leq u$ in Ω a.e. for $i=1,2$.*

Proof. For $g(x) \in H^{3/2}(\partial\Omega), f(x) \in L^2(\Omega)$, by the standard elliptic theory, (2.3) admits a unique solution $w \in H^2(\Omega) \hookrightarrow C^0(\overline{\Omega})$ [20, 22], so that $e^{-w} \in L^\infty(\Omega) \subseteq L^2(\Omega)$. Hence the initial guess $u_1^0 = w \in M$. It is obvious $u_2^0 \in M$. For $\xi \in H_g^1(\Omega)$, set the nonlinear operator

$$L(\xi,v) := A(\xi,v) - (f + e^{-\xi}, v), \quad \forall v \in H_0^1(\Omega),\ v \geq 0 \text{ a.e.}$$

(i) By calculation we have

$$L(u_1^0, v) = -(e^{-w}, v) \leq 0 = L(u,v).$$

Hence u_1^0 is a weak subsolution of (1.1). This together with $u_1^0 = u = g(x)$ on $\partial\Omega$ yields $u_1^0 \leq u(x)$ in Ω a.e. by the comparison principle.

(ii) From the choice of u_2^0 it follows that

$$f(x) + e^{-u_2^0} \geq 0, \quad x \in \Omega,$$

hence

$$L(u_2^0, v) = -(f(x) + e^{-u_2^0}, v) \leq 0 = L(u,v).$$

Hence u_2^0 is a weak subsolution, which together with $u_2^0 \leqslant g(x)$ yields $u_2^0 \leq u$ in Ω a.e. □

2.2 Existence, uniqueness and monotonicity

We first prepare the following lemma.

Lemma 2.2. *Let $\tilde{k}$ be a non-negative number. If ξ with $\xi|_{\partial\Omega} = 0$ satisfies*

$$A(\xi,v) + \widetilde{k}(\xi,v) \leqslant 0, \quad \forall v \in H_0^1(\Omega),\ v \geqslant 0, \tag{2.7}$$

then

$$\xi \leqslant 0 \quad \text{in } \Omega \text{ a.e.}$$

Proof. We choose $v = \xi^+$ with $(\cdot)^+ = \max\{\cdot, 0\}$, so that $v \in H_0^1(\Omega), v \geqslant 0$ *a.e.* Hence

$$A(\xi,\xi^+) + \widetilde{k}(\xi,\xi^+) \leqslant 0. \tag{2.8}$$

This implies

$$\int_{\{\xi \geq 0\}} \lambda^2 |\nabla\xi|^2 + \tilde{k}|\xi|^2 dx \leqslant 0, \quad \text{in } \Omega \text{ a.e.}, \tag{2.9}$$

which along with $\xi|_{\partial\Omega} = 0$ infers $\xi \leqslant 0$ in Ω a.e. since $\tilde{k} \geq 0$. □

This is used to establish the following result by the mathematical induction.

Theorem 2.1 (Existence, uniqueness and monotonicity)**.** *Let the iterative parameter be* $k^n = e^{-\text{essinf}(u^n)}$ *or* $k^n \equiv k^0 = e^{-\text{essinf}(u^0)}$. *If* $u^0 = u_i^0 (i = 1,2)$, *then* (2.6) *admits a unique solution* $u^{n+1} \in M$, *and this solution sequence* $\{u^n(x)\}$ $(n = 0,1,\cdots)$ *satisfies*

$$u^0 \leqslant u^1 \leqslant \cdots \leqslant u^n \leqslant u^{n+1} \leqslant \cdots \leqslant u \quad in\ \ \Omega\ a.e., \tag{2.10}$$

where u *is the solution of* (2.5).

Proof. We present the proof for $u^0 = u_1^0$, the proof for $u^0 = u_2^0$ is entirely similar. First, from (2.6) it follows that if $u^n \in M$ and $k^n \geq 0$, then (2.6) admits a unique solution $u^{n+1} \in H^2(\Omega) \hookrightarrow C^0(\overline{\Omega})$, so $e^{-u^{n+1}} \in L^\infty(\Omega) \subseteq L^2(\Omega)$, which means $u^{n+1} \in M$. By induction we conclude that $u^n \in M$ for all n for $u^0 \in M$.

Next we show the monotonicity in two steps. (i) $u^0 \leqslant u^1$ in Ω a.e. Relation (2.6) with $n = 0$ and $A(u^0, v) = (f, v)$ gives

$$A(u^0 - u^1, v) + k^0(u^0 - u^1, v) = -(e^{-u^0}, v) \leq 0 \tag{2.11}$$

for all $v \in H_0^1(\Omega), v \geq 0$ a.e. By Lemma 2.2 we have $u^0 \leqslant u^1$ in Ω a.e. for $k^0 \geqslant 0$.

(ii) Assume $u^0 \leqslant u^1 \leqslant \cdots \leqslant u^n$ in Ω a.e., we only need to show

$$u^n \leqslant u^{n+1} \quad \text{in}\ \ \Omega\ a.e.$$

Note that the assumption implies the following

$$0 < k^n \leqslant \cdots \leqslant k^1 \leqslant k^0,$$

for $k^i = e^{-\text{essinf}(u^i)}$ or $k^i \equiv k^0 = e^{-\text{essinf}(u^0)}$.

In order to show $u^n \leqslant u^{n+1}$ in Ω a.e., we subtract (2.6) for u^n and u^{n+1}, respectively, to obtain

$$\begin{aligned} A(u^n - u^{n+1}, v) + k^n(u^n - u^{n+1}, v) &= \int_\Omega k^{n-1}(u^{n-1} - u^n)v + (e^{-u^{n-1}} - e^{-u^n})v dx \\ &= \int_\Omega (k^{n-1} + \frac{e^{-u^{n-1}} - e^{-u^n}}{u^{n-1} - u^n})(u^{n-1} - u^n)v dx. \end{aligned} \tag{2.12}$$

For $k^{n-1} = e^{-\text{essinf}(u^{n-1})}$ or $k^{n-1} \equiv k^0 = e^{-\text{essinf}(u^0)}$ we have

$$k^{n-1} + \frac{e^{-u^{n-1}} - e^{-u^n}}{u^{n-1} - u^n} \geqslant 0,$$

for $(u^{n-1} - u^n) \leq 0$. This shows that the right hand side of (2.12) is non-positive for all $v \in H_0^1(\Omega), v \geq 0$ a.e. By Lemma 2.2 we have $u^n \leqslant u^{n+1}$ in Ω a.e.

Finally we show $u^n \leqslant u$ in Ω a.e. for all n, still by induction using $u^0 \leq u$. Subtraction of (2.5) from (2.6) for u^{n+1} gives

$$A(u^{n+1} - u, v) + k^{n-1}(u^{n+1} - u, v) = \int_\Omega (k^n + \frac{e^{-u^n} - e^{-u}}{u^n - u})(u^n - u)v dx. \tag{2.13}$$

Similarly as above, for $u^n - u \leqslant 0$, the right hand side of (2.13) is non-positive for all $v \in H_0^1(\Omega), v \geq 0$ a.e. By Lemma 2.2 we have $u^{n+1} \leqslant u$ in Ω a.e. □

2.3 Convergence and convergence speed

The following lemma indicates the effect of the iteration parameter on the convergence speed.

Lemma 2.3. *Given u^n, let u_1^{n+1} and u_2^{n+1} be the solutions of (2.6) with the choices of the parameter k^n to be $e^{-\text{essinf}(u^0)}$ and $e^{-\text{essinf}(u^n)}$, respectively. Then*

$$u_1^{n+1} \leqslant u_2^{n+1}.$$

Proof. Subtracting (2.6) for u_2^{n+1} and (2.6) for u_1^{n+1} we obtain

$$A(u_1^{n+1}-u_2^{n+1},v)+(k^0u_1^{n+1}-k^nu_2^{n+1},v)=\int_\Omega (k^0-k^n)u^nvdx,$$

which leads to

$$A(u_1^{n+1}-u_2^{n+1},v)+k^n(u_1^{n+1}-u_2^{n+1},v)=\int_\Omega (k^0-k^n)(u^n-u_1^{n+1})vdx. \tag{2.14}$$

For $0<k^n\leq k^0, u^n\leq u_1^{n+1}$, we have $(k^n-k^0)(u^n-u_1^{n+1})\leq 0$, hence the right hand side of (2.14) is non-positive for all $v\in H_0^1(\Omega), v\geq 0$ a.e. By Lemma 2.2 we have $u_1^{n+1}\leqslant u_2^{n+1}$ in Ω a.e. □

Regarding the convergence of the solution sequence $\{u^n\}$, we have the following result.

Theorem 2.2 (Convergence)**.** *The solution sequence $\{u^n\}$ of (2.6) converges to the solution u of (2.5) in M.*

Proof. For $u,u^n\in M$, we set $\xi^n=u^n-u$, then $\xi^n\in H_0^1(\Omega)$; and (2.13) with $v=\xi^{n+1}$ yields

$$A(\xi^{n+1},\xi^{n+1})+k^n(\xi^{n+1},\xi^{n+1})=\left(\left(k^n+\frac{e^{-u^n}-e^{-u}}{\xi^n}\right)\xi^n,\xi^{n+1}\right). \tag{2.15}$$

The right hand side is bounded above by

$$\frac{k^n}{2}\|\xi^{n+1}\|^2+\frac{k^n\alpha_n^2}{2}\|\xi^n\|^2,$$

where

$$\alpha_n=\operatorname*{esssup}_\Omega\left|1+\frac{e^{-u^n}-e^{-u}}{k^n\xi^n}\right|\leq\alpha:=1-\frac{e^{-\text{esssup}u}}{k^0}<1,$$

by using the fact that $u^0\leq u^n\leq u$ and $0<k^n\leq k^0$. Hence

$$\frac{2\lambda^2}{k^n}\|\nabla_x\xi^{n+1}\|^2+\|\xi^{n+1}\|^2\leqslant\alpha^2\|\xi^n\|^2. \tag{2.16}$$

From the Poincaré inequality of the form $\|\xi^{n+1}\|^2 \leq \frac{1}{C_\Omega}\|\nabla\xi^{n+1}\|^2$, where C_Ω is a constant related to the domain Ω, (2.16) yields

$$\|\xi^{n+1}\| \leq \frac{\alpha}{\sqrt{1+\frac{2C_\Omega\lambda^2}{k^n}}}\|\xi^n\| \leq \frac{\alpha}{\sqrt{1+\frac{2C_\Omega\lambda^2}{k^0}}}\|\xi^n\|. \tag{2.17}$$

Hence $\|\xi^n\| \to 0$, from which and (2.16) again we see that $\xi^n \to 0$ in $H_0^1(\Omega)$. Therefore $u^n \to u$ in M. □

Remark 2.1. From (2.17) we also obtain the convergence rate

$$\|u^n - u\| \leq \left(\frac{\alpha}{\sqrt{1+\frac{2C_\Omega\lambda^2}{k^0}}}\right)^n \|u^0 - u\|, \tag{2.18}$$

which indicates that the larger λ is, the faster the convergence is.

3 DDG discretization

3.1 One dimensional formulation

In one-dimensional case, we partition the domain $I=[a,b]$ into computational elements $I_j=(x_{j-1/2},x_{j+1/2})$, $\Delta x = x_{j+1/2}-x_{j-1/2}$, with $x_{1/2}=a$ and $x_{N+1/2}=b$. And we define the finite element space

$$V^m = \{v \in L^2(I) : v|_{I_j} \in P^m(I_j), j=1,2,\cdots,N\}.$$

Eq. (2.3) for $u^0=w$ in one dimensional case reduces to

$$\begin{cases} -\lambda^2 u^0_{xx} = f(x) \quad \text{in } I, \\ u^0(a) = g_1, \\ u^0(b) = g_2. \end{cases} \tag{3.1}$$

By the DDG method we find $u_h^0 \in V^m$ such that

$$\lambda^2\int_{I_j} u^0_{hx}v_x dx - \lambda^2\widehat{(u^0_{hx})}v|_{\partial I_j} + \frac{\lambda^2}{2}[u_h^0](v_x)^-_{j+1/2} + \frac{\lambda^2}{2}[u_h^0](v_x)^+_{j-1/2} = \int_{I_j} fv dx, \tag{3.2}$$

for all $v \in V^m$. Here and in what follows the notation $v|_{\partial I_j}$ is used to denote $v^-_{j+1/2} - v^+_{j-1/2}$. The iteration (2.1) becomes

$$\begin{cases} -\lambda^2 u^{n+1}_{xx} + k^n u^{n+1} = k^n u^n + f(x) + e^{-u^n} \quad \text{in } I, \\ u^{n+1}(a) = g_1, \\ u^{n+1}(b) = g_2. \end{cases} \tag{3.3}$$

The DDG scheme for (3.3) is to find $u_h^{n+1}\in V^m$ so that for all $v\in V^m$,

$$\lambda^2\int_{I_j}u_{hx}^{n+1}v_x dx+k^n\int_{I_j}u_h^{n+1}vdx-\lambda^2\widehat{(u_{hx}^{n+1})}v|_{\partial I_j}$$
$$+\frac{\lambda^2}{2}[u_h^{n+1}](v_x)_{j+1/2}^-+\frac{\lambda^2}{2}[u_h^{n+1}](v_x)_{j-1/2}^+=\int_{I_j}F_h^n vdx, \tag{3.4}$$

where $F_h^n=k^nu_h^n+f(x)+e^{-u_h^n}$, $k^n=e^{-\min(u_h^n)}$. The numerical flux for both (3.2) and (3.4) on each cell interface is of the form:

$$\widehat{u_{hx}}=\beta_0\frac{[u_h]}{\Delta x}+\overline{u_{hx}}+\beta_1\Delta x[u_{hxx}]. \tag{3.5}$$

Here $u_h^\pm=u_h(x\pm0)$, $[u_h]=u_h^+-u_h^-$, $\overline{u_h}=\frac{u_h^++u_h^-}{2}$. On the boundary we take

$$[u_h]_{1/2}=u_{h1/2}^+-g_1,\quad [u_h]_{N+1/2}=g_2-u_{h(N+1/2)}^-,$$

so that the numerical flux for both (3.2) and (3.4) on the boundary becomes

$$\widehat{u_{hx}}_{1/2}=\beta_0\frac{(u_{h1/2}^+-g_1)}{\Delta x}+(u_{hx})_{1/2}^+, \tag{3.6a}$$
$$\widehat{u_{hx}}_{N+1/2}=\beta_0\frac{(g_2-u_{h(N+1/2)}^-)}{\Delta x}+(u_{hx})_{N+1/2}^-. \tag{3.6b}$$

We remark that for non-uniform meshes Δx in the numerical flux formula needs to be replaced by $(x_{j+1}-x_j)/2$. The iteration algorithm is complete once the parameters β_0 and β_1 are chosen to make the scheme stable.

3.2 Two dimensions with rectangular meshes

In two-dimensional case, we present DDG schemes on uniform rectangular meshes and triangular meshes. If the domain Ω is rectangular, i.e., $\Omega=[a,b]\times[c,d]$. We partition Ω by rectangular meshes

$$\Omega=\sum_{j,k}^{N,M}K_{j,k},\quad K_{j,k}=I_j\times J_k=[x_{j-\frac{1}{2}},x_{j+\frac{1}{2}}]\times[y_{k-\frac{1}{2}},y_{k+\frac{1}{2}}] \tag{3.7}$$

of uniform mesh sizes $\Delta=\max\{\Delta x,\Delta y\}$, where $\Delta x=x_{j+1/2}-x_{j-1/2}$, $\Delta y=y_{j+1/2}-y_{j-1/2}$. The DDG scheme is to find $u^n|_{K_{j,k}}\in V^m$ $(n=0,1,2,\cdots)$ such that

$$\lambda^2\iint_{K_{j,k}}\nabla u_h^0\cdot\nabla vdxdy-\int_{J_k}\hat{f}_1{}^0v|_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}}dy-\int_{I_j}\hat{f}_2{}^0v|_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}}dx+B^0=\iint_{K_{j,k}}fvdxdy, \tag{3.8}$$

$$\lambda^2 \iint_{K_{j,k}} \nabla u_h^{n+1}\cdot\nabla v dxdy + k^n \int_{K_{j,k}} u^{n+1} v dxdy - \int_{J_k} \hat{f}_1{}^{n+1} v|_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} dy$$
$$- \int_{I_j} \hat{f}_2{}^{n+1} v|_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}} dx + B^{n+1} = \iint_{K_{j,k}} F_h^n v dxdy, \tag{3.9}$$

where

$$B^n = \frac{1}{2}\left\{ \int_{J_k} ([u_h^n]v_x^-)_{x_{j+\frac{1}{2}}} + ([u_h^n]v_x^+)_{x_{j-\frac{1}{2}}} dy + \int_{I_j} ([u_h^n]v_y^-)_{y_{k+\frac{1}{2}}} + ([u_h^n]v_y^+)_{y_{k-\frac{1}{2}}} dx \right\},$$
$$[u_h]_{x_{j+\frac{1}{2}}} := u_h(x_{j+\frac{1}{2}}^+, y) - u_h(x_{j+\frac{1}{2}}^-, y), \quad [u_h]_{y_{k+\frac{1}{2}}} := u_h(x, y_{k+\frac{1}{2}}^+) - u_h(x, y_{k+\frac{1}{2}}^-).$$

The numerical flux $\hat{f}_i{}^n (i=1,2)$ on each cell interface is of the form:

$$\hat{f}_1{}^n|_{x_{j+\frac{1}{2}}} = \left(\beta_0 \frac{[u_h^n]}{\Delta x} + \overline{u_{hx}^n} + \beta_1 \Delta x [u_{hxx}^n] \right)|_{x_{j+\frac{1}{2}}}, \tag{3.10a}$$

$$\hat{f}_2{}^n|_{y_{k+\frac{1}{2}}} = \left(\beta_0 \frac{[u_h^n]}{\Delta y} + \overline{u_{hy}^n} + \beta_1 \Delta y [u_{hyy}^n] \right)|_{y_{k+\frac{1}{2}}}. \tag{3.10b}$$

Incorporating the boundary data $g(a,y)=g_1, g(b,y)=g_2, g(x,c)=g_3$ and $g(x,d)=g_4$ into the numerical flux $\hat{f}_i{}^n (i=1,2)$ on the boundary, so that

$$\hat{f}_1{}^n|_{x_{\frac{1}{2}}} = \left(\beta_0 \frac{(u_h^n - g_1)}{\Delta x} + (u_{hx}^n)^+ \right)|_{x_{\frac{1}{2}}}, \tag{3.11a}$$

$$\hat{f}_2{}^n|_{y_{\frac{1}{2}}} = \left(\beta_0 \frac{(u_h^n - g_3)}{\Delta y} + (u_{hy}^n)^+ \right)|_{y_{\frac{1}{2}}}, \tag{3.11b}$$

$$\hat{f}_1{}^n|_{x_{N+\frac{1}{2}}} = \left(\beta_0 \frac{(g_2 - u_h^n)}{\Delta x} + (u_{hx}^n)^- \right)|_{x_{N+\frac{1}{2}}}, \tag{3.11c}$$

$$\hat{f}_2{}^n|_{y_{N+\frac{1}{2}}} = \left(\beta_0 \frac{(g_4 - u_h^n)}{\Delta y} + (u_{hy}^n)^- \right)|_{y_{N+\frac{1}{2}}}, \tag{3.11d}$$

$$[u_h]_{x_{\frac{1}{2}}} = u_h(x_{\frac{1}{2}}^+, y) - g_1, \quad [u_h]_{x_{N+\frac{1}{2}}} = g_2 - u_h(x_{N+\frac{1}{2}}^-, y), \tag{3.11e}$$

$$[u_h]_{y_{\frac{1}{2}}} = u_h(x, y_{\frac{1}{2}}^+) - g_3, \quad [u_h]_{y_{N+\frac{1}{2}}} = g_4 - u_h(x, y_{N+\frac{1}{2}}^-). \tag{3.11f}$$

3.3 Two dimensions with unstructured meshes

We partition a general domain Ω into shape-regular meshes $T_\Delta = \{K\}$, consisting of a nonoverlapping open element covering completely the domain Ω. We denote by Δ the piecewise constant mesh function with $\Delta(x) \equiv \Delta_K = diam\{K\}$, when x is in element K.

The DDG schemes for (2.3) and (2.1) are to find $u_h^0|_K \in V^m(K)$ and $u_h^{n+1}|_K \in V^m(K)$ $(n=0,1,2,\cdots)$, respectively, such that

$$\lambda^2 \iint_K \nabla u_h^0 \cdot \nabla v dx - \lambda^2 \int_{\partial K} \widehat{u_{h\mathbf{n}}^0} v ds + \frac{\lambda^2}{2} \int_{\partial K} [u_h^0] v_{\mathbf{n}} ds = \iint_K f v dx, \tag{3.12}$$

$$\lambda^2 \iint_K \nabla u_h^{n+1} \cdot \nabla v dx + k^n \int_K u_h^{n+1} v dx - \lambda^2 \int_{\partial K} \widehat{u_{h\mathbf{n}}^{n+1}} v ds + \frac{\lambda^2}{2} \int_{\partial K} [u_h^{n+1}] v_{\mathbf{n}} ds = \iint_K F_h^n v dx, \tag{3.13}$$

where $\mathbf{n}=(n_x,n_y)$ is the outward normal unit along the cell boundary ∂K and $u_{h\mathbf{n}}=\nabla u_h \cdot \mathbf{n}$. The numerical flux on each interior cell interface is of the form:

$$\widehat{u_{h\mathbf{n}}} = \beta_0 \frac{[u_h]}{\Delta} + \overline{\frac{\partial u_h}{\partial \mathbf{n}}} + \beta_1 \Delta [u_{h\mathbf{nn}}], \tag{3.14}$$

where $\Delta = diam\{K\}$ is the mesh size and

$$[u_h] = u_h^{ext_K} - u_h^{int_K}, \quad \overline{u_{h\mathbf{n}}} = \frac{u_{h\mathbf{n}}^{ext_K} + u_{h\mathbf{n}}^{int_K}}{2}.$$

Here $u_{h\mathbf{n}}^{ext_K}$ represents u_h evaluated from outside of K (inside the neighboring cell). Using the boundary data $g=g_0$, we take the numerical flux $\widehat{u_{\mathbf{n}}}$ on the boundary as

$$\widehat{u_{h\mathbf{n}}} = \beta_0 \frac{g_0 - u_h^{int_K}}{\Delta} + \partial_{\mathbf{n}} u_h^{int_K}. \tag{3.15}$$

If the cell boundaries are straight lines, such as the triangular meshes, the above numerical fluxes become

$$\widehat{u_{h\mathbf{n}}} = \widehat{u_{hx}} n_x + \widehat{u_{hy}} n_y$$

with

$$\widehat{u_{hx}} = \beta_0 \frac{[u_h]}{\Delta} n_x + \overline{u_{hx}} + \beta_1 \Delta ([u_{hxx}] n_x + [u_{hxy}] n_y), \tag{3.16a}$$

$$\widehat{u_{hy}} = \beta_0 \frac{[u_h]}{\Delta} n_y + \overline{u_{hx}} + \beta_1 \Delta ([u_{hyx}] n_x + [u_{hyy}] n_y). \tag{3.16b}$$

In other words:

$$\widehat{u_{h\mathbf{n}}} = \beta_0 \frac{[u_h]}{\Delta} + \overline{u_{hx} n_x + u_{hy} n_y} + \beta_1 ([u_{hxx}] n_x^2 + 2[u_{hxy}] n_x n_y + u_{hyy}] n_y^2). \tag{3.17}$$

The numerical flux on the boundary then becomes

$$\widehat{u_{h\mathbf{n}}} = \beta_0 \frac{g_0 - u_h^{int_K}}{\Delta} + u_{hx}^{int_K} n_x + u_{hy}^{int_K} n_y. \tag{3.18}$$

3.4 Existence, uniqueness and stability

For simplicity, we discuss the existence, uniqueness and stability of the DDG schemes only for one-dimensional case; similar analysis applies to the two-dimensional DDG scheme as well.

Summation of (3.4) over $j=1,2\cdots N$ gives

$$\lambda^2\sum_{j=1}^{N}\int_{I_j}u_{hx}^{n+1}v_x dx+k^n\int_\Omega u_h^{n+1}v dx-\lambda^2\sum_{j=1}^{N}\widehat{(u_{hx}^{n+1})}v|_{\partial I_j}+\frac{\lambda^2}{2}\sum_{j=1}^{N}[u_h^{n+1}](v_x)_{j+1/2}^{-}$$
$$+\frac{\lambda^2}{2}\sum_{j=1}^{N}[u_h^{n+1}](v_x)_{j-1/2}^{+}=\int_\Omega F_h^n v dx, \quad (3.19)$$

which can be rewritten as

$$A(u_h^{n+1},v)=\int_\Omega F_h^n v dx \quad \forall v\in V^m, \quad (3.20)$$

where the bilinear operator $A(\cdot,\cdot)$ is defined as

$$A(u_h^{n+1},v)=\lambda^2\sum_{j=1}^{N}\int_{I_j}u_{hx}^{n+1}v_x dx+k^n\int_\Omega u_h^{n+1}v dx+\lambda^2\sum_{j=0}^{N}\left(\widehat{u_{hx}^{n+1}}[v]\right)_{j+1/2}$$
$$+\lambda^2\sum_{j=0}^{N}\left([u_h^{n+1}](\overline{v_x})\right)_{j+1/2}. \quad (3.21)$$

Here $(u_{h1/2}^{n+1})^-=g_1$, $(u_{h(N+1/2)}^{n+1})^+=g_2$ and we have used the notation

$$[v]_{1/2}=v_{1/2}^+, \quad [v]_{N+1/2}=v_{N+1/2}^-, \quad (\overline{v_x})_{1/2}=\frac{1}{2}(v_x)_{1/2}^+, \quad (\overline{v_x})_{N+1/2}=\frac{1}{2}(v_x)_{N+1/2}^-.$$

We now discuss the stability of scheme (3.20), which implies both uniqueness and the existence of the scheme.

To identify the admissible parameters (β_0,β_1), we define the discrete DG norm of $v\in V^m$

$$\|v\|_E^2=\sum_{j=1}^{N}\int_{I_j}|v_x|^2dx+\frac{\beta_0}{\Delta x}\sum_{j=0}^{N}[v]_{j+1/2}^2 \quad (3.22)$$

and the quantity

$$\Gamma(\beta_1)=\sup_{u\in P^{m-1}([-1,1])}\frac{(u(1)-2\beta_1\partial_\tau u(1))^2}{\int_{-1}^{1}u^2(\tau)d\tau}. \quad (3.23)$$

Remark 3.1. It is shown in [23] that $\Gamma(\beta_1)$ is related to the numerical flux through the following inequality

$$(2\overline{\partial_x v}+\beta_1\Delta x[\partial_x^2 v])^2|_{x_{j+1/2}}\le\frac{4\Gamma(\beta_1)}{\Delta x}\left(\int_{I_j}+\int_{I_{j+1}}|v_x|^2dx\right). \quad (3.24)$$

Remark 3.2. A quantitative estimate for $\Gamma(\beta_1)$ is known, see [24]: for any $k\geq 1$, it holds that

$$\Gamma(\beta_1)=2m^2\left(1-\beta_1(m^2-1)+\frac{\beta_1^2}{3}(m^2-1)^2\right). \tag{3.25}$$

The stability of the DDG scheme (3.20) is ensured for some parameters (β_0,β_1).

Theorem 3.1 (Stability)**.** *If*

$$\beta_0>\beta_0^*:=\max\left\{2\Gamma(\beta_1),\frac{9}{4}\Gamma(0)\right\}, \tag{3.26}$$

then there exists a constant C such that

$$\|u_h^{n+1}-\widetilde{u}_h^{n+1}\|_E\leq\frac{C}{\lambda^2\gamma+C^{-2}k^n}\|F_h^n-\widetilde{F}_h^n\|,\quad \gamma=1-\sqrt{\frac{\beta_0^*}{\beta_0}}, \tag{3.27}$$

where u_h^{n+1} and $\widetilde{u}_h^{n+1}$ are the solutions of (3.20) associated with F_h^n and $\widetilde{F}^n$, respectively.

Proof. From (3.20) it follows that

$$A(u_h^{n+1}-\widetilde{u}_h^{n+1},v)=\int_\Omega(F_h^n-\widetilde{F}_h^n)vdx. \tag{3.28}$$

If we set $v=u_h^{n+1}-\widetilde{u}_h^{n+1}$, then $v|_{\partial\Omega}=0$ and

$$A(v,v)\leq\|F_h^n-\widetilde{F}_h^n\|\|v\|. \tag{3.29}$$

Note that with $v=0$ on the boundary, it can be verified that $\|v\|_E$ is a norm. By norm equivalence in finite dimensional space, we have

$$C^{-1}\|v\|_E\leq\|v\|\leqslant C\|v\|_E, \tag{3.30}$$

for some constants C. On the other hand, for β_0 satisfying (3.26), we claim that

$$(\lambda^2\gamma+C^{-2}k^n)\|v\|_E^2\leq A(v,v) \tag{3.31}$$

for some $0<\gamma<1$. These put together lead to

$$(\lambda^2\gamma+C^{-2}k^n)\|v\|_E^2\leqslant A(v,v)\leqslant C\|F_h^n-\widetilde{F}_h^n\|\|v\|_E.$$

This gives (3.27).

We now complete the proof by showing claim (3.31). By regrouping terms in $A(v,v)$, we have

$$\begin{aligned}A(v,v)-k^n\int_\Omega v^2dx\geq&\lambda^2\left(\frac{1}{2}\int_{I_1}v_x^2dx+(\widehat{v_x}+\overline{v_x})[v]_{1/2}\right)+\lambda^2\left(\frac{1}{2}\int_{I_N}v_x^2dx+(\widehat{v_x}+\overline{v_x})[v]_{N+1/2}\right)\\&+\lambda^2\sum_{j=1}^{N-1}\left[\frac{1}{2}\left(\int_{I_j}+\int_{I_{j+1}}\right)v_x^2dx+(\widehat{v_x}+\overline{v_x})[v]_{j+1/2}\right].\end{aligned}$$

The last term in the bracket $[\cdot]$ is shown in [23] bounded from below by

$$\gamma_1\left(\frac{1}{2}\left(\int_{I_j}+\int_{I_{j+1}}\right)v_x^2dx+\beta_0\frac{[v]_{j+1/2}^2}{\Delta x}\right)$$

for

$$\gamma_1=1-\sqrt{\frac{2\Gamma(\beta_1)}{\beta_0}}\in(0,1).$$

A similar analysis can be applied to boundary terms, which we present here for illustration. For the first term, we have

$$\begin{aligned}
&\frac{1}{2}\int_{I_1}|v_x|^2dx+(\widehat{v_x}+\overline{v_x})[v]_{1/2}\\
=&\frac{1}{2}\int_{I_1}|v_x|^2dx+\left(\beta_0\frac{[v]_{1/2}}{\Delta x}+\frac{3}{2}(v_x)_{1/2}^+\right)[v]_{1/2}\\
\geq&\frac{1}{2}\int_{I_1}|v_x|^2dx+\gamma_2\frac{\beta_0}{\Delta x}[v]_{1/2}^2-\frac{9\Delta x}{16\beta_0(1-\gamma_2)}((v_x)_{1/2}^+)^2\\
=&\gamma_2\left(\frac{1}{2}\int_{I_1}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{1/2}^2\right)+\frac{1-\gamma_2}{2}\int_{I_1}|v_x|^2dx-\frac{9\Delta x}{16\beta_0(1-\gamma_2)}((v_x)_{1/2}^+)^2\\
=&\left(\gamma_2+\frac{\frac{1-\gamma_2}{2}\int_{I_1}|v_x|^2dx-\frac{9\Delta x}{16\beta_0(1-\gamma_2)}((v_x)_{1/2}^+)^2}{\frac{1}{2}\int_{I_1}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{1/2}^2}\right)\left(\frac{1}{2}\int_{I_1}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{1/2}^2\right)\\
=&\left(\gamma_2+\frac{\frac{1-\gamma_2}{2}\left(\int_{I_1}|v_x|^2dx-\frac{1}{2\Gamma(0)}((v_x)_{1/2}^+)^2\right)}{\frac{1}{2}\int_{I_1}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{1/2}^2}\right)\left(\frac{1}{2}\int_{I_1}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{1/2}^2\right),
\end{aligned}$$

where we have chosen $\beta_0=\frac{9}{4(1-\gamma_2)^2}\Gamma(0)$ so that

$$\gamma_2=1-\frac{3}{2}\sqrt{\frac{\Gamma(0)}{\beta_0}}\in(0,1).$$

Note that

$$\Gamma(0)=\sup_{u\in P^{m-1}([-1,1])}\frac{u^2(1)}{\int_{-1}^1u^2(\tau)d\tau}\geq\frac{\Delta x((v_x)_{1/2}^+)^2}{2\int_{I_1}|v_x|^2dx},$$

we therefore obtain

$$\frac{1}{2}\int_{I_1}|v_x|^2dx+(\widehat{v_x}+\overline{v_x})[v]_{1/2}\geq\gamma_2\left(\frac{1}{2}\int_{I_1}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{1/2}^2\right).$$

Similarly,

$$\frac{1}{2}\int_{I_N}|v_x|^2dx+(\widehat{v_x}+\overline{v_x})[v]_{N+1/2}\geq\gamma_2\left(\frac{1}{2}\int_{I_N}|v_x|^2dx+\frac{\beta_0}{\Delta x}[v]_{N+1/2}^2\right).$$

Taking $\gamma=\min\{\gamma_1,\gamma_2\}$, we thus obtain

$$A(v,v)\geq\lambda^2\gamma\|v\|_E^2+k^n\|v\|^2\geq(\lambda^2\gamma+C^{-2}k^n)\|v\|_E^2.$$

The proof is complete. □

Remark 3.3. From the stability estimate (3.27) we see that the amplification factor is inversely proportional to

$$\lambda^2\left(1-\sqrt{\frac{\beta_0^*}{\beta_0}}\right)+C^{-2}k^n.$$

This implies that the convergence may be affected by three factors: λ, (β_0,β_1) and the step size k^n. For a fixed k^n and the given numerical flux, the smaller λ is, the poorer the numerical performance (see Fig. 3) is. For small λ, this suggests the use of a numerical flux with larger β_0 to improve the performance of the iterative DG method.

Remark 3.4. The error estimate of optimal order $\mathcal{O}((\Delta x)^{m+1})$ is more involved, and may be obtained by using some global projection similar to that introduced in [24]. However, we choose not to pursue these details in this paper.

4 Numerical examples

In this section we present several numerical examples to solve one-dimensional and two-dimensional Poisson-Boltzmann problems. In these examples we demonstrate both the accuracy of the iterative DG method and the effectiveness of the iteration in terms of both initial guess and the iterative parameter. Also, we investigate the numerical performance of the iterative DG method as the parameter λ varies, ranging from $\mathcal{O}(1)$ to $o(1)$. In the following examples, three typical values of λ are tested.

Example 4.1. We solve the nonlinear Poisson-Boltzmann equation of the form

$$-\lambda^2u_{xx}=f+e^{-u},\quad u(0)=0,\quad u(1)=1 \tag{4.1}$$

on the domain $[0,1]$, where $f=-\lambda^2\pi^2\sin\pi x-e^{-\sin\pi x-x}$. The exact solution is

$$u=-\sin\pi x-x.$$

Test case 1. We test problem (4.1) with $\lambda=1$, using the numerical flux (3.5) with $\beta_0=2,\beta_1=\frac{1}{12}$, as introduced by Liu and Yan [12]:

$$u_{hx}=2\frac{[u_h]}{\Delta x}+\overline{u_{hx}}+\frac{1}{12}\Delta x[u_{hxx}]. \tag{4.2}$$

The iterative DG method based on polynomials of degree m with $m=1,2,\cdots,7$ is tested, $(m+1)$th order of accuracy for L^2 and mth order of accuracy for H^1 are obtained. The

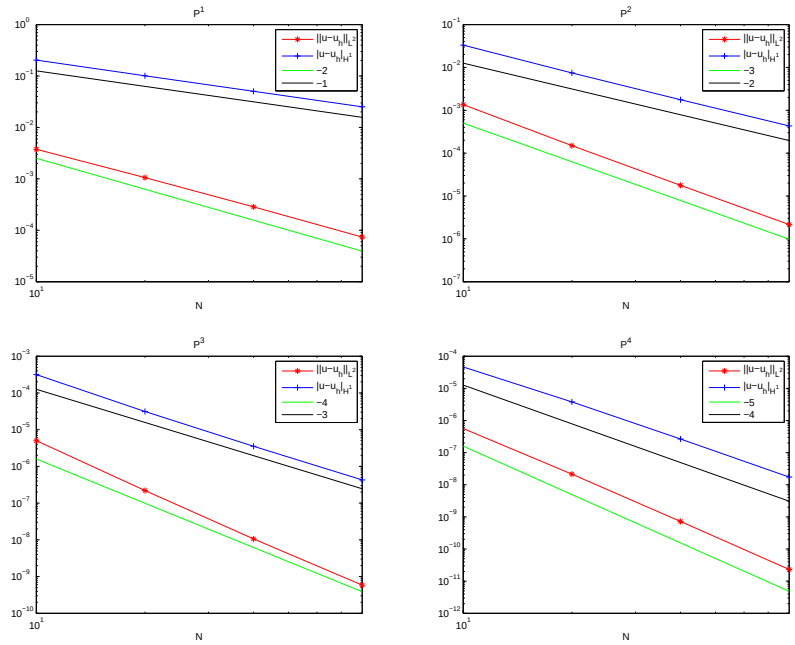

Figure 1: L^2 errors and H^1 errors for 1D PB equation using the iterative DG method with numerical flux (4.2), $k^n=e^{-\min(u_n)}$, $\lambda=1$. (a) P^1 approximation; (b) P^2 approximation; (c) P^3 approximation; (d) P^4 approximation.

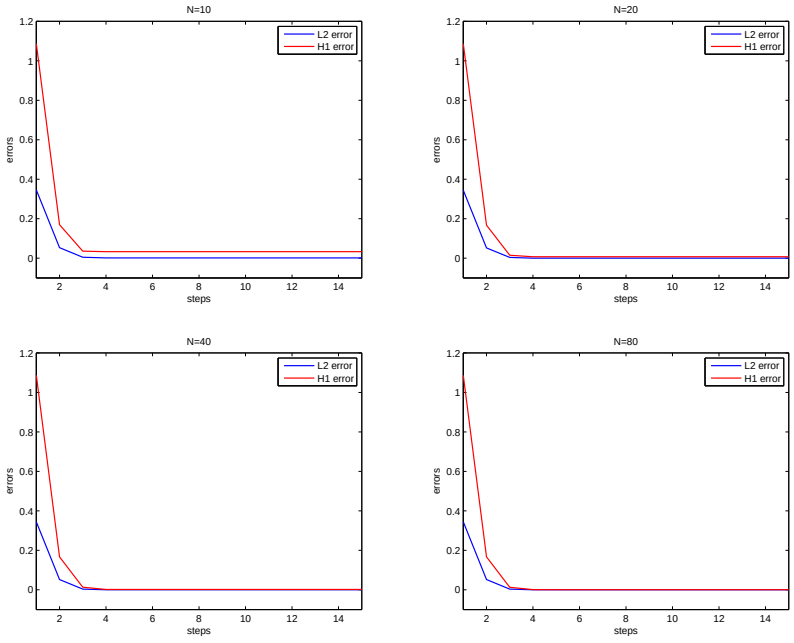

Figure 2: The L^2 errors and H^1 errors between numerical solutions at every step and the exact solution for the 1D nonlinear PB equation, using the iterative DG method with numerical flux (4.2) and mesh number N, $k^n=e^{-\min(u_n)}$, $\lambda=1$, P^2 approximation. (a) $N=10$; (b) $N=20$; (c) $N=40$ and (d) $N=80$.

Table 1: 1D PB equation with numerical flux (4.2), $k^n = e^{-\min(u_n)}$, $\lambda = 1$.

m	steps		$N=10$	$N=20$		$N=40$		$N=80$	
			error	error	order	error	order	error	order
1	15	$\|u-u_h\|_{L^2}$	0.00375734	0.00105854	1.83	0.000284523	1.90	7.38155e-05	1.95
		$\|u-u_h\|_{H^1}$	0.203656	0.101007	1.01	0.0503995	1.00	0.025187	1.00
2	15	$\|u-u_h\|_{L^2}$	0.00135051	0.000149582	3.17	1.76829e-05	3.08	2.15174e-06	3.04
		$\|u-u_h\|_{H^1}$	0.0331646	0.00742854	2.16	0.00176604	2.07	0.000430958	2.04
3	15	$\|u-u_h\|_{L^2}$	5.00619e-06	2.20369e-07	4.51	1.06276e-08	4.37	5.84378e-10	4.19
		$\|u-u_h\|_{H^1}$	0.000315741	3.12041e-05	3.34	3.53595e-06	3.14	4.28119e-07	3.05
4	15	$\|u-u_h\|_{L^2}$	5.60650e-07	2.13301e-08	4.72	7.18633e-10	4.89	2.31376e-11	4.96
		$\|u-u_h\|_{H^1}$	4.60193e-05	3.78428e-06	3.60	2.64381e-07	3.84	1.73076e-08	3.93
			$N=4$	$N=8$		$N=16$		$N=32$	
5	15	$\|u-u_h\|_{L^2}$	1.42907e-07	1.83359e-09	6.28	2.69864e-11	6.09	4.14942e-13	6.02
		$\|u-u_h\|_{H^1}$	8.20283e-06	2.21214e-07	5.21	6.58277e-09	5.07	2.02953e-10	5.02
6	15	$\|u-u_h\|_{L^2}$	1.09212e-08	1.07461e-10	6.67	9.10789e-13	6.88	7.36054e-15	6.95
		$\|u-u_h\|_{H^1}$	4.88549e-07	1.05659e-08	5.53	1.88938e-10	5.81	3.1199e-12	5.92
7	15	$\|u-u_h\|_{L^2}$	8.93136e-11	3.01587e-13	8.21	1.14287e-15	8.04	9.54879e-17	–
		$\|u-u_h\|_{H^1}$	6.88814e-09	4.96217e-11	7.12	3.77111e-13	7.04	3.82719e-15	–

Table 2: 1D PB equation with numerical flux (4.2), $k^n \equiv e^{-\min(u_0)}$, $\lambda = 1$.

m	steps		$N=10$	$N=20$		$N=40$		$N=80$	
			error	error	order	error	order	error	order
1	24	$\|u-u_h\|_{L^2}$	0.00375734	0.00105854	1.83	0.000284523	1.90	7.38155e-05	1.95
		$\|u-u_h\|_{H^1}$	0.203656	0.101007	1.01	0.0503995	1.00	0.025187	1.00
2	24	$\|u-u_h\|_{L^2}$	0.00135051	0.000149582	3.18	1.76829e-05	3.08	2.15174e-06	3.04
		$\|u-u_h\|_{H^1}$	0.0331646	0.00742854	2.16	0.00176604	2.07	0.000430958	2.04
3	24	$\|u-u_h\|_{L^2}$	5.00619e-06	2.20369e-07	4.51	1.06276e-08	4.37	5.84378e-10	4.19
		$\|u-u_h\|_{H^1}$	0.000315741	3.12041e-05	3.34	3.53595e-06	3.14	4.28119e-07	3.05
4	24	$\|u-u_h\|_{L^2}$	5.60650e-07	2.13301e-08	4.72	7.18633e-10	4.89	2.31378e-11	4.96
		$\|u-u_h\|_{H^1}$	4.60193e-05	3.78428e-06	3.60	2.64381e-07	3.84	1.73076e-08	3.93
			$N=4$	$N=8$		$N=16$		$N=32$	
5	24	$\|u-u_h\|_{L^2}$	1.42907e-07	1.83359e-09	6.28	2.69864e-11	6.09	4.14945e-13	6.02
		$\|u-u_h\|_{H^1}$	8.20283e-06	2.21214e-07	5.21	6.58277e-09	5.07	2.02953e-10	5.02
6	24	$\|u-u_h\|_{L^2}$	1.09212e-08	1.07461e-10	6.67	9.10798e-13	6.88	7.34704e-15	6.95
		$\|u-u_h\|_{H^1}$	4.88549e-07	1.05659e-08	5.53	1.88937e-10	5.81	3.12018e-12	5.92
7	24	$\|u-u_h\|_{L^2}$	8.93136e-11	3.01589e-13	8.21	1.14813e-15	8.04	8.87827e-17	–
		$\|u-u_h\|_{H^1}$	6.88814e-09	4.96218e-11	7.12	3.77157e-13	7.04	7.08744e-15	–

L^2 errors and H^1 errors are reported in Table 1 with $k^n = e^{-\min(u_n)}$, and in Table 2 with $k^n \equiv e^{-\min(u_0)}$. From Tables 1 and 2 we find that the iterative DG scheme (3.4) with $k^n = e^{-\min(u_n)}$ converges quicker than that with $k^n \equiv e^{-\min(u_0)}$, where the iteration process is terminated when $\|u_h^{n+1} - u_h^n\|_2 < 1.0 \times 10^{-15}$.

In Fig. 1, the L^2 errors and H^1 errors for one dimensional PB problem are shown by using the data in Table 1. In Fig. 2, we show the L^2 errors and H^1 errors between numerical solutions and the exact solution at every step as the iteration proceeds. From this figure, we see that the iterative DG method only needs few steps for numerical solutions to converge to the exact solution.

Remark 4.1. Since the iterative DG scheme (3.4) with the variable iteration parameter $k^n = e^{-\min(u_n)}$ converges more quickly, we choose to use $k^n = e^{-\min(u^n)}$ in the remaining numerical tests.

Test case 2. We test problem (4.1) again with the parameter $\lambda = 0.1$ and $\lambda = 0.01$. For these cases, the iterative DG method with flux (4.2) does not seem to converge to the correct solution, as shown in Fig. 3, and explained in Remark 3.3 earlier. The remedy is to take the numerical flux (3.5) with β_0, β_1 listed in Table 3, where $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ are parameters adapted in terms of the degree of the polynomial elements. We choose $\alpha_0 = \alpha_1 = \alpha_2 = \alpha_3 = 4.5$ in this test case so that the theoretical bound (3.26) is satisfied. The comparison of results using flux (4.2) and flux (3.5), as given in Fig. 3, indicates that Theorem 3.1 indeed provides a reliable guide of selecting numerical fluxes to ensure the convergence to the correct solution.

Table 3: The choice of β_i for P^m polynomials.

m	1	2	3	4	5
β_0	$\frac{9}{4}$	$\frac{15\alpha_0}{4}$	$\frac{225\alpha_1}{32}$	$\frac{225\alpha_2}{32}$	$\frac{225\alpha_3}{32}$
β_1	0	$\frac{3}{80}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$

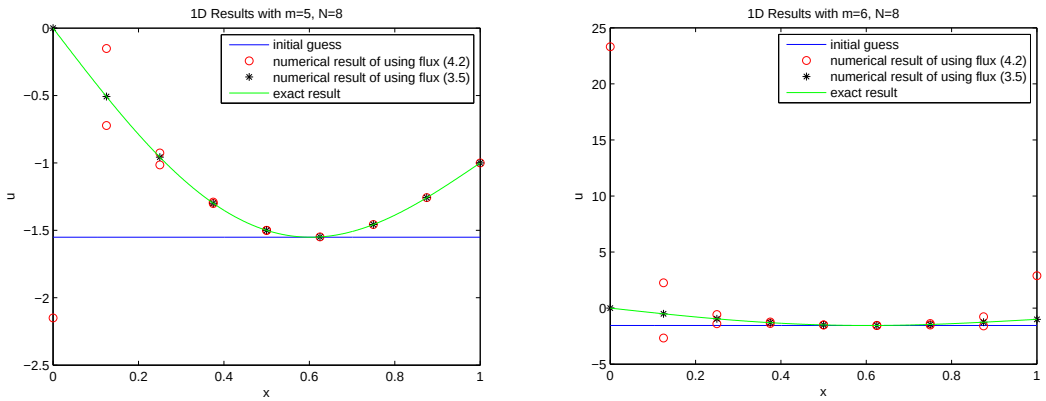

Figure 3: Some wrong numerical results for 1D PB equation of using the flux (4.2), $k^n = e^{-\min(u_n)}$, $\lambda = 1$. (a) P^5 approximation, $N = 8$; (b) P^6 approximation, $N = 8$.

Table 4: 1D PB equation with numerical flux (3.5), $k^n = e^{-\min(u_n)}$, $\lambda = 0.1$.

m	steps		$N=10$	$N=20$		$N=40$		$N=80$	
			error	error	order	error	order	error	order
1	41	$\|u-u_h\|_{L^2}$	0.00272401	0.000660542	2.04	0.00016337	2.02	4.07333e-05	2.00
		$\|u-u_h\|_{H^1}$	0.202102	0.100825	1.00	0.0503763	1.00	0.025184	1.00
2	41	$\|u-u_h\|_{L^2}$	0.000196287	2.38116e-05	3.04	2.9066e-06	3.03	3.59369e-07	3.02
		$\|u-u_h\|_{H^1}$	0.0110174	0.00249597	2.14	0.000590115	2.08	0.000143552	2.04
3	41	$\|u-u_h\|_{L^2}$	2.36491e-06	1.44073e-07	4.04	8.95017e-09	4.01	5.58557e-10	4.00
		$\|u-u_h\|_{H^1}$	0.000232211	2.75155e-05	3.08	3.40055e-06	3.02	4.24034e-07	3.00
			$N=4$	$N=8$		$N=16$		$N=32$	
4	41	$\|u-u_h\|_{L^2}$	2.89394e-06	9.75097e-08	4.89	3.13445e-09	4.96	9.91100e-11	4.98
		$\|u-u_h\|_{H^1}$	0.000175772	1.0715e-05	4.04	6.63452e-07	4.01	4.13049e-08	4.01
5	41	$\|u-u_h\|_{L^2}$	1.02671e-07	1.66779e-09	5.94	2.63473e-11	5.98	4.19109e-13	5.97
		$\|u-u_h\|_{H^1}$	6.71679e-06	2.07011e-07	5.02	6.46345e-09	5.00	2.07793e-10	4.96

Table 5: 1D PB equation with numerical flux (3.5), $k^n = e^{-\min(u_n)}$, $\lambda = 0.01$.

m	steps		$N=10$	$N=20$		$N=40$		$N=80$	
			error	error	order	error	order	error	order
1	80	$\|u-u_h\|_{L^2}$	0.00260376	0.000651137	2.00	0.00016264	2.00	4.06445e-05	2.00
		$\|u-u_h\|_{H^1}$	0.201373	0.100753	1.00	0.0503707	1.00	0.0251835	1.00
2	80	$\|u-u_h\|_{L^2}$	7.10296e-05	9.58057e-06	2.89	1.35586e-06	2.82	1.78935e-07	2.92
		$\|u-u_h\|_{H^1}$	0.00907876	0.00230191	1.98	0.000581707	1.98	0.000144037	2.01
3	80	$\|u-u_h\|_{L^2}$	1.40190e-06	9.68992e-08	3.86	7.39186e-09	3.71	5.25593e-10	3.81
		$\|u-u_h\|_{H^1}$	0.000250604	2.94131e-05	3.09	3.46043e-06	3.09	4.24843e-07	3.02
			$N=4$	$N=8$		$N=16$		$N=32$	
4	80	$\|u-u_h\|_{L^2}$	2.17520e-06	7.53449e-08	4.85	2.63829e-09	4.84	8.89299e-11	4.89
		$\|u-u_h\|_{H^1}$	0.000208239	1.18607e-05	4.13	6.93167e-07	4.10	4.1928e-08	4.05
5	80	$\|u-u_h\|_{L^2}$	7.02893e-08	1.1915e-09	5.88	2.21184e-11	5.75	4.50900e-13	5.62
		$\|u-u_h\|_{H^1}$	8.60295e-06	2.40058e-07	5.16	6.71722e-09	5.16	2.03235e-10	5.05

The iterative DG method based on polynomials of degree m with $m = 1,2,\cdots,5$ is tested. We observe $(m+1)$th order of accuracy for L^2 and mth order of accuracy for H^1. The L^2 errors and H^1 errors are reported in Table 4 ($\lambda = 0.1$) and Table 5 ($\lambda = 0.01$), respectively. The iteration process is terminated when $\|u_h^{n+1} - u_h^n\|_2 < 1.0 \times 10^{-12}$.

We also record the CPU times of these three tests in Table 6 ($\lambda = 1$), Table 7 ($\lambda = 0.1$), and Table 8 ($\lambda = 0.01$), respectively. And we find that the complexity of using the iterative DG method to solve one-dimensional PB equations is $\mathcal{O}(N)$.

This example shows that the iterative DG method with a variable iteration parameter k^n and appropriate initial guess u^0 is quite efficient for solving the one-dimensional nonlinear Poisson-Boltzmann equation (4.1) for a large range of λ. The iteration steps may increase dramatically as λ becomes smaller, as explained in Remark 3.3.

Table 6: CPU time for solving 1D PB equation with numerical flux (3.5), $k^n=e^{-\min(u_n)}$, $\lambda=1$.

m	steps	$N=10$	$N=20$		$N=40$		$N=80$	
		CPU(s)	CPU(s)	times	CPU(s)	times	CPU(s)	times
1	15	0.01	0.03	3.00	0.05	1.67	0.11	2.20
2	15	0.02	0.04	2.00	0.08	2.00	0.15	1.88
3	15	0.02	0.05	2.50	0.10	2.00	0.19	1.90
4	15	0.03	0.06	2.00	0.12	2.00	0.25	2.08
		$N=4$	$N=8$		$N=16$		$N=32$	
5	15	0.02	0.03	1.50	0.06	2.00	0.13	2.17
6	15	0.02	0.04	2.00	0.08	2.00	0.14	1.75
7	15	0.03	0.05	1.67	0.09	1.80	0.18	2.00

Table 7: CPU time for solving 1D PB equation with numerical flux (3.5), $k^n=e^{-\min(u_n)}$, $\lambda=0.1$.

m	steps	$N=10$	$N=20$		$N=40$		$N=80$	
		CPU(s)	CPU(s)	times	CPU(s)	times	CPU(s)	times
1	41	0.05	0.09	1.80	0.18	2.00	0.35	1.94
2	41	0.06	0.12	2.00	0.24	2.00	0.48	2.00
3	41	0.08	0.16	2.00	0.31	1.94	0.64	2.06
		$N=4$	$N=8$		$N=16$		$N=32$	
4	41	0.04	0.08	2.00	0.16	2.00	0.32	2.00
5	41	0.05	0.10	2.00	0.20	2.00	0.41	2.05

Table 8: CPU time for solving 1D PB equation with numerical flux (3.5), $k^n=e^{-\min(u_n)}$, $\lambda=0.01$.

m	steps	$N=10$	$N=20$		$N=40$		$N=80$	
		CPU(s)	CPU(s)	times	CPU(s)	times	CPU(s)	times
1	80	0.10	0.19	1.90	0.35	1.84	0.70	2.00
2	80	0.13	0.24	1.85	0.46	1.92	0.95	2.07
3	80	0.15	0.30	2.00	0.63	2.10	1.28	2.03
		$N=4$	$N=8$		$N=16$		$N=32$	
4	80	0.08	0.16	2.00	0.32	2.00	0.64	2.00
5	80	0.10	0.19	1.90	0.39	2.05	0.80	2.05

Remark 4.2. In this example, we find that when λ is small, fluxes with β_0,β_1 listed in Table 3 work well and will be adopted in the following example. However, as shown in Fig. 3, (4.2) leads to the wrong solution when λ is small.

Example 4.2. We consider

$$\begin{cases} -\lambda^2\Delta u=f(x)+e^{-u}, & (x,y)\in\Omega=[0,1]^2\subset\mathbb{R}^2, \\ u=g(x), & \text{on } \partial\Omega. \end{cases} \tag{4.3}$$

We test the case with $f=2\lambda^2\pi^2\cos\pi x\cos\pi y-e^{-(\cos\pi x\cos\pi y)}$ and with $g=\cos\pi y\cos\pi x$ on

the boundary. The exact solution is

$$u = \cos \pi x \cos \pi y.$$

In our numerical simulation the numerical flux (3.10) with β_0, β_1 listed in Table 3 is adopted, where we take $\alpha_0 = \alpha_1 = \alpha_2 = \alpha_3 = 25$ so that (3.26) is satisfied.

Table 9: 2D PB equation with numerical flux (3.10), $k^n = e^{-\min(u_n)}$, $\lambda = 1$.

m	steps		$N=4$	$N=8$		$N=16$		$N=32$	
			error	error	order	error	order	error	order
1	10	$\|u-u_h\|_{L^2}$	0.0384311	0.00907175	2.08	0.00200517	2.18	0.000438976	2.19
		$\|u-u_h\|_{H^1}$	0.571003	0.273687	1.06	0.131827	1.05	0.0644315	1.03
2	10	$\|u-u_h\|_{L^2}$	0.00146901	0.000148121	3.31	1.72802e-05	3.10	2.12061e-06	3.03
		$\|u-u_h\|_{H^1}$	0.0580651	0.0141317	2.04	0.00350617	2.01	0.000874728	2.00
3	10	$\|u-u_h\|_{L^2}$	0.000259793	1.86371e-05	3.80	1.21185e-06	3.94	7.65025e-08	3.99
		$\|u-u_h\|_{H^1}$	0.00614743	0.000716945	3.10	7.83854e-05	3.19	8.77698e-06	3.16
4	10	$\|u-u_h\|_{L^2}$	2.48821e-06	8.52127e-08	4.87	3.22777e-09	4.72	1.84377e-10	–
		$\|u-u_h\|_{H^1}$	0.000170959	1.11810e-05	3.93	8.19604e-07	3.77	8.04222e-08	–

Table 10: 2D PB equation with numerical flux (3.10), $k^n = e^{-\min(u_n)}$, $\lambda = 0.1$.

m	steps		$N=4$	$N=8$		$N=16$		$N=32$	
			error	error	order	error	order	error	order
1	58	$\|u-u_h\|_{L^2}$	0.0231556	0.0056022	2.05	0.00138851	2.01	0.000347826	2.00
		$\|u-u_h\|_{H^1}$	0.505541	0.252421	1.00	0.126003	1.00	0.0629688	1.00
2	58	$\|u-u_h\|_{L^2}$	0.00127086	0.00014274	3.15	1.71164e-05	3.06	2.11542e-06	3.02
		$\|u-u_h\|_{H^1}$	0.0571981	0.014094	2.02	0.00350399	2.01	0.000874592	2.00
3	58	$\|u-u_h\|_{L^2}$	0.000186514	1.37671e-05	3.76	9.00703e-07	3.93	5.69821e-08	3.98
		$\|u-u_h\|_{H^1}$	0.006235	0.000738052	3.08	7.97449e-05	3.21	8.84077e-06	3.17
4	58	$\|u-u_h\|_{L^2}$	2.41178e-06	8.44386e-08	4.84	3.22323e-09	4.71	1.84220e-10	–
		$\|u-u_h\|_{H^1}$	0.000171034	1.11785e-05	3.94	8.19411e-07	3.77	8.03926e-08	–

Table 11: 2D PB equation with numerical flux (3.10), $k^n = e^{-\min(u_n)}$, $\lambda = 0.01$.

m	steps		$N=4$	$N=8$		$N=16$		$N=32$	
			error	error	order	error	order	error	order
1	139	$\|u-u_h\|_{L^2}$	0.0165748	0.0042745	1.96	0.00107781	1.99	0.000267543	2.01
		$\|u-u_h\|_{H^1}$	0.503224	0.253107	0.99	0.126554	1.00	0.0631373	1.00
2	139	$\|u-u_h\|_{L^2}$	0.00108095	0.000135074	3.00	1.68938e-05	3.00	2.1097e-06	3.00
		$\|u-u_h\|_{H^1}$	0.0558528	0.0139832	2.00	0.00349775	2.00	0.000874302	2.00
3	139	$\|u-u_h\|_{L^2}$	5.69636e-05	4.4864e-06	3.67	3.71585e-07	3.59	2.72899e-08	3.78
		$\|u-u_h\|_{H^1}$	0.00418383	0.000556026	2.91	7.23784e-05	2.94	8.79737e-06	3.04
4	139	$\|u-u_h\|_{L^2}$	1.92748e-06	7.25892e-08	4.73	3.0392e-09	4.58	1.73680e-10	–
		$\|u-u_h\|_{H^1}$	0.000177023	1.12462e-05	3.98	8.1001e-07	3.80	7.81402e-08	–

Table 12: CPU time for solving 2D PB equation with numerical flux (3.10), $k^n=e^{-\min(u_n)}$, $\lambda=1$.

m	steps	$N=4$	$N=8$		$N=16$		$N=32$	
		CPU(s)	CPU(s)	times	CPU(s)	times	CPU(s)	times
1	10	0.31	1.47	4.74	7.19	4.89	38.67	5.38
2	10	0.96	4.99	5.20	25.16	5.04	170.24	6.77
3	10	4.20	20.50	4.88	89.65	4.37	575.97	6.42

For $\lambda=1,0.1,0.01$, the iterative DG method based on P^m polynomial elements with $m=1,2,3,4$ is tested, $(m+1)$th order of accuracy for L^2 and mth order of accuracy for H^1 are obtained. The L^2 errors and H^1 errors are reported in Tables 9, 10 and 11.

In Table 12 ($\lambda=1$), we recorded the CPU times of using the iterative DG method to solve 2D PB equations on rectangular meshes.

Example 4.3. We still consider the problem in Example 4.2 but on triangular meshes.

Let a partition of Ω be denoted by triangular meshes $T_\Delta=\{K\}$. We apply the iterative DG schemes (3.8)-(3.9) to solve problem (4.3). The numerical flux (3.17) with β_0,β_1 listed in Table 3 is adopted, where we take $\alpha_0=\alpha_1=\alpha_2=25$ to satisfy (3.26).

For $\lambda=1,0.1,0.01$, the iterative DG schemes on triangular meshes are tested with P^m polynomials, where $m=1,2,3$, and $(m+1)$th order of accuracy for L^2 and mth order of accuracy for H^1 are obtained. The L^2 errors and H^1 errors are reported in Tables 13, 14 and 15.

Table 13: 2D PB equation with numerical flux (3.17), $k^n=e^{-\min(u_n)}$, $\lambda=1$.

m	steps		$N=4$	$N=8$		$N=16$		$N=32$	
			error	error	order	error	order	error	order
1	10	$\|u-u_h\|_{L^2}$	0.0259938	0.00781174	1.73	0.00214009	1.87	0.000557754	1.95
		$\|u-u_h\|_{H^1}$	0.647868	0.328298	0.98	0.163701	1.00	0.0815838	1.00
2	10	$\|u-u_h\|_{L^2}$	0.00324821	0.000418215	2.96	5.22119e-05	3.00	6.49584e-06	3.01
		$\|u-u_h\|_{H^1}$	0.110847	0.0283651	1.97	0.00714056	1.99	0.00178911	2.00
3	10	$\|u-u_h\|_{L^2}$	0.00061284	3.53173e-05	4.12	2.19363e-06	4.01	1.30518e-07	4.07
		$\|u-u_h\|_{H^1}$	0.0187405	0.00204057	3.20	0.000266413	2.94	2.63866e-05	3.34

Table 14: 2D PB equation with numerical flux (3.17), $k^n=e^{-\min(u_n)}$, $\lambda=0.1$.

m	steps		$N=4$	$N=8$		$N=16$		$N=32$	
			error	error	order	error	order	error	order
1	49	$\|u-u_h\|_{L^2}$	0.0206556	0.00533035	1.95	0.0013348	2.00	0.00033237	2.01
		$\|u-u_h\|_{H^1}$	0.642811	0.326972	0.98	0.163471	1.00	0.0815497	1.00
2	49	$\|u-u_h\|_{L^2}$	0.00239552	0.000314025	2.93	4.01245e-05	2.97	5.06518e-06	2.99
		$\|u-u_h\|_{H^1}$	0.111528	0.0284419	1.97	0.00714915	1.99	0.00179009	2.00
3	49	$\|u-u_h\|_{L^2}$	0.000275597	1.76915e-05	3.96	1.10900e-06	4.00	6.92706e-08	4.00
		$\|u-u_h\|_{H^1}$	0.0122049	0.00153368	2.99	0.000191629	3.00	2.39329e-05	3.00

Table 15: 2D PB equation with numerical flux (3.17), $k^n = e^{-\min(u_n)}$, $\lambda = 0.01$.

m	steps		$N=4$	$N=8$		$N=16$		$N=32$	
			error	error	order	error	order	error	order
1	112	$\|u-u_h\|_{L^2}$	0.0195516	0.00496765	1.98	0.00125193	1.99	0.00031442	1.99
		$\|u-u_h\|_{H^1}$	0.624371	0.318817	0.97	0.161208	0.98	0.0811283	0.99
2	112	$\|u-u_h\|_{L^2}$	0.00219109	0.000279128	2.97	3.60964e-05	2.95	4.6591e-06	2.95
		$\|u-u_h\|_{H^1}$	0.110065	0.0280879	1.97	0.00710074	1.98	0.00178588	1.99
3	112	$\|u-u_h\|_{L^2}$	0.000201862	1.39748e-05	3.85	9.63149e-07	3.86	6.51691e-08	3.89
		$\|u-u_h\|_{H^1}$	0.0134849	0.00162723	3.05	0.000194973	3.06	2.39792e-05	3.02

Table 16: CPU time for solving 2D PB equation with numerical flux (3.17), $k^n = e^{-\min(u_n)}$, $\lambda = 1$.

m	steps	$N=4$	$N=8$		$N=16$		$N=32$	
		CPU(s)	CPU(s)	times	CPU(s)	times	CPU(s)	times
1	10	0.34	1.62	4.76	8.63	5.33	53.26	6.17
2	10	0.76	3.84	5.05	21.10	5.49	136.00	6.45
3	10	1.91	10.38	5.43	57.93	5.58	507.08	8.75

In Table 16 ($\lambda = 1$), we recorded the CPU times of using the iterative DG method to solve the 2D PB problem on triangular meshes.

For both rectangular and triangular meshes, the complexity is found of order $\mathcal{O}(N^2)$, and the tests were carried out on a PC with 1.86GB memory, which may lead to the increase of the CPU time greatly as mesh is refined. The 2D results in the above two examples indicate that the iterative DG method is also quite efficient for solving the two dimensional Poisson-Boltzmann equation (4.3), for a large range of λ.

5 Conclusion

In this paper, we have developed an iterative discontinuous Galerkin method to solve the nonlinear Poisson Boltzmann (PB) equation. The iteration with a variable iterative parameter k^n is more efficient than a constant one, and robust in terms of the size of the Debye length parameter. The corresponding initial guess u^0 proposed works well for different cases in the numerical tests. The numerical fluxes were selected to assure the existence, uniqueness and stability of the resulting DG schemes. Both one-dimensional and two-dimensional numerical results show that the numerical fluxes with a large set of (β_0,β_1) pairs gives the optimal accuracy. The optimal error estimation of the iterative DG method for nonlinear Poisson-Boltzmann equation is left for a future work.

Acknowledgments

The authors thank the anonymous referees who provided valuable comments resulting in improvements in this paper. Yin's research was partially supported by NSFC Project

(11201397), Hunan Education Department Project (12B127) and Hunan Provincial Innovation Foundation for Postgraduate (CX2012B272). Huang's research was supported in part by the NSFC Key Project (11031006), IRT1179 of PCSIRT and ISTCP of China (2010DFR00700). Liu's research was partially supported by the National Science Foundation under the Grant DMS09-07963 and DMS13-12636.

References

[1] M. Holst, J. A. McCammom, Z. Yu, Y. C. Zhou, Y. Zhu. Adaptive finite element modeling techniques for the Poisson-Boltzmann equation. Commun. Comput. Phys., 11(1):179–214, 2012/01.
[2] P. Debye, E. Hückel. Zur theorie der elektrolyte. Phys. Zeitschr., 24:185-206, 1923.
[3] J. G. Kirkwood. On the theory of strong electrolyte solutions. J. Chem. Phys., 2:767-781, 1934.
[4] A. H. Boschitsch, M. O. Fenley. Hybrid boundary element and finite difference method for solving the nonlinear Poisson-Boltzmann equation. J. Comput. Chem., 25(7):935-955, 2004.
[5] N. A. Baker, D. Sept D., S. Joseph, M. J. Holst, J. A. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. Proc. Natl. Acad. Sci. USA, 98:10037-10041, 2001.
[6] Z.-H. Qiao, Z.-L. Li, T. Tang. A finite difference scheme for solving the nonlinear Poisson-Boltzmann equation modeling charged spheres. J. Comput. Math., 24(3):252-264, 2006/03.
[7] Phillip Colella, Milo R Dorr, Daniel D Wake. A conservative finite difference method for the numerical solution of plasma fluid equations. J. Comput. Phys., 149(1):168-193, 1999.
[8] L. Chen, M. J. Holst, J. Xu. The finite element approximation of the nonlinear Poisson-Boltzmann equation. SIAM J. Numer. Anal., 45(6):2298-2320, 2007.
[9] B. Lu, X. Cheng, J. Huang. AFMPB: An adaptive fast multipole Poisson-Boltzmann solver for calculating electrostatics in biomolecular systems. C. Phys. Commun., 181(6):1150-1160, 2010.
[10] P. Degond, H. Liu, D. Savelief, M. H. Vignal. Numerical approximation of the Euler-Poisson-Boltzmann model in the quasineutral limit. J. Sci. Comput., 51:59-86, 2012.
[11] H. Liu, J. Yan. The Direct Discontinuous Galerkin (DDG) method for diffusion problems. SIAM Journal on Numerical Analysis., 47(1): 675–698, 2009.
[12] H. Liu, J. Yan. The Direct Discontinuous Galerkin (DDG) method for diffusion with interface corrections. Commun. Comput. Phys., 8(3): 541–564, 2010.
[13] Y. Huang, H. Liu, N. Yi. Recovery of normal derivatives from the piecewise L2 projection. J. Comput. Phys., 231(4):1230-1243, 2012.
[14] H. Oberoi, N. M. Allewell. Multigrid solution of the nonlinear Poisson-Boltzmann equation and calculation of titration curves. Biophys J., 65(1):48-55, 1993.
[15] A. Nicholls, B. Honig. A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. Journal of Computational Chemistry, 12(4):435-445, 1991.
[16] M. E. Davis, J. A. McCammon. Solving the finite difference linearized Poisson-Boltzmann equation: A comparison of relaxation and conjugate gradient methods. Journal of Computational Chemistry, 10(3):386-391, 1989.
[17] Y. Deng, G. Chen, W. Ni, J. Zhou. Boundary element monotone iteration scheme for semi-linear elliptic partial differential equations. Mathematics of Computation, 65(215):943-982,

1996.
[18] L. F. Shampine. Monotone iterations and two-sided convergence. SIAM Journal on Numerical Analysis, 3(4):607-615, 1996.
[19] I. Chern, J. Liu, W. Wang. Accurate evaluation of electrostatics for macromolecules in solution. Methods and Applications of Analysis, 10(2):309-328, 2003.
[20] A. Quarteroni, A. Valli. Numerical approximation of partial differential equations. Springer Series in Computational Mathematics, 2008.
[21] L. C. Evans. Partial differential equations. American Mathematics Society, 19, 2010.
[22] J. L. Lions, E. Magenes. Non-Homogeneous Boundary Value Problems and Applications. New-York: Springer-Verlag, 1972.
[23] H. Liu, H. Yu. The entropy satisfying discontinuous Galerkin method for Fokker-Planck equations. Journal on Scientific Computing, accepted (2014).
[24] H. Liu. Optimal error estimates of the Direct Discontinuous Galerkin method for convection-diffusion equations. Math. Comp., to appear (2014).