A POSTERIORI ERROR ESTIMATOR FOR LINEAR ELASTICITY BASED ON NONSYMMETRIC STRESS TENSOR APPROXIMATION
   KWANG-YEON KIM

A ROBUST AND ACCURATE PHASE-FIELD SIMULATION OF SNOW CRYSTAL GROWTH
   YIBAO LI, DONGSUN LEE, HYUN GEUN LEE, DARA JEONG, CHAEYOUNG LEE,
   DONGGYU YANG, AND JUNSEOK KIM

SURFACE RECONSTRUCTION FROM SCATTERED POINT DATA ON OCTREE
   CHANGSOO PARK, CHOHONG MIN, AND MYUNGJOO KANG

THE DIMENSION REDUCTION ALGORITHM FOR THE POSITIVE REALIZATION OF DISCRETE PHASE-TYPE DISTRIBUTIONS
   KYUNGSUP KIM

THE ISOGEOMETRIC VARIATIONAL MULTISCALE METHOD FOR LAMINAR INCOMPRESSIBLE FLOW
   YOURSELF GAFFERS MOULAGE AND HYUNG TAEK AHN
A POSTERIORI ERROR ESTIMATOR FOR LINEAR ELASTICITY BASED ON NONSYMMETRIC STRESS TENSOR APPROXIMATION

KWANG-YEON KIM

DEPARTMENT OF MATHEMATICS, KANGWON NATIONAL UNIVERSITY, CHUNCHEON, SOUTH KOREA
E-mail address: eulerkim@kangwon.ac.kr

ABSTRACT. In this paper we present an a posteriori error estimator for the stabilized P1 non-conforming finite element method of the linear elasticity problem based on a nonsymmetric $H(\text{div})$-conforming approximation of the stress tensor in the first-order Raviart–Thomas space. By combining the equilibrated residual method and the hypercircle method, it is shown that the error estimator gives a fully computable upper bound on the actual error. Numerical results are provided to confirm the theory and illustrate the effectiveness of our error estimator.

1. INTRODUCTION

In this paper we consider a homogeneous linear elastic material occupying a planar domain $\Omega$ whose displacement $u : \Omega \rightarrow \mathbb{R}^2$ is described by the equations

$$\sigma = C\varepsilon(u), \quad -\text{div} \sigma = f \quad \text{in } \Omega \quad (1.1)$$

subject to the boundary conditions

$$u = u_D \quad \text{on } \Gamma_D, \quad \sigma n = 0 \quad \text{on } \Gamma_N := \partial \Omega \setminus \Gamma_D, \quad (1.2)$$

where $\sigma$ is the Cauchy stress tensor, $\varepsilon(u) = \frac{1}{2}(\nabla u + (\nabla u)^T)$ is the linearized strain tensor, and $n$ is the unit outward normal on $\partial \Omega$. For simplicity, we assume that $\Gamma_D$ is not empty, but it is easy to extend all subsequent results to the pure Neumann case.

The coefficient $C$ is a fourth-order elasticity tensor which is bounded, uniformly positive definite and satisfies the symmetry condition. Hereafter we restrict ourselves to the isotropic material in a state of plane strain, in which case $C$ is given in the form

$$C\varepsilon(u) = 2\mu\varepsilon(u) + \lambda \text{div} u I,$$

where $I$ is the $2 \times 2$ identity tensor and $\mu, \lambda$ are the Lamé constants satisfying $0 < \mu_1 < \mu < \mu_2$ for some fixed $\mu_1, \mu_2$ and $0 < \lambda < \infty$.

Received by the editors July 7 2011; Accepted December 23 2011.
2010 Mathematics Subject Classification. 65N30, 65N15.
Key words and phrases. a posteriori error estimator, linear elasticity, $P1$ nonconforming finite element method.
This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (Grant No. 2009–0073414).
The weak formulation for (1.1)–(1.2) seeks the displacement \( u \in H^{1}_{D}(\Omega; u_D) \) such that
\[
(\mathcal{C}\epsilon(u), \epsilon(v))_\Omega = (f, v)_\Omega \quad \forall v \in H^{1}_{D}(\Omega),
\]
where \((\cdot, \cdot)_\Omega\) is the standard inner product in \((L^2(\Omega))^d\) \((d = 1, 2)\) and
\[
H^{1}_{D}(\Omega; u_D) := \{ v \in (H^1(\Omega))^2 : v|_{\Gamma_D} = u_D \}, \quad H^{1}_{D}(\Omega) := H^{1}_{D}(\Omega; 0).
\]
We adopt the standard notation for the Sobolev space \( H^k(S) \) over a set \( S \) equipped with the norm \( \| \cdot \|_{k,S} \) and semi-norm \( | \cdot |_{k,S} \).

Nowadays it is well established that one should apply adaptive mesh refinement based on a posteriori error estimators for efficient implementation of numerical methods. Various types of error estimators have been developed and successfully implemented for the linear elasticity problem; see, for example, the survey paper [1]. We are particularly interested in the error estimators which give fully computable upper bounds on the actual error without involving unknown constants. Such error estimators were constructed in [2] for the \( P^1 \) conforming element and in [3] for the \( P^2 \) conforming and nonconforming elements by combining the equilibrated residual method and the hypercircle method. The key step there was recovery of a symmetric \( H(\text{div}) \)-conforming approximation of the stress tensor \( \sigma \) in an appropriate finite element space of symmetric tensors from the equilibrated normal stress approximation, which is quite complicated due to the large dimension of the local finite element space used (the Arnold–Winther space in [2] and the Arnold–Douglas–Gupta space in [3]), although the computation is done locally on each element.

In this paper we propose a new error estimator which requires much less computation than [2, 3], while achieving fully computable upper bounds on the actual error. This is accomplished by recovering a nonsymmetric \( H(\text{div}) \)-conforming approximation of the stress tensor \( \sigma \) in the nonsymmetric Raviart–Thomas space of first order whose local dimension is smaller and makes the implementation easier than the symmetric tensor spaces mentioned above. When compared with the ones from [2, 3], our estimator contains the additional contribution arising from the non-symmetry of the recovered stress tensor approximation and thus strongly depending on computable upper bounds on the constants of local Korn’s inequality. A similar consideration was given to the Stokes problem in [4] but the derived estimator was not fully computable as the constant in the upper bound was not estimated.

To fix ideas, we consider the stabilized \( P^1 \) nonconforming finite element proposed in [5] for which, unlike the \( P^1 \) conforming element, the equilibrated normal stress approximation is explicitly constructed without solving local linear systems. It is straightforward to apply subsequent results to other \( P^1 \) and \( P^2 \) finite elements as long as the equilibrated normal stress approximation (defined in Section 3) is available.

The rest of the paper is organized as follows. In the next section we introduce the stabilized \( P^1 \) nonconforming finite element method and some computable estimates related to Korn’s inequality. In Section 3 we construct the nonsymmetric \( H(\text{div}) \)-conforming approximation of the stress tensor and then derive the a posteriori error estimator which yields an upper bound on the actual error in Section 4. Finally, Section 5 presents some numerical results to confirm the theory and illustrate the effectiveness of our error estimator.
2. Stabilized P1 Nonconforming FEM and Korn’s Inequality

2.1. Stabilized P1 Nonconforming FEM. Let \( \mathcal{T}_h = \{T\} \) be a regular triangulation of \( \Omega \) into triangles with \( h_T = \text{diam}(T) \) and \( h = \max_{T \in \mathcal{T}_h} h_T \). The set of three edges of an element \( T \in \mathcal{T}_h \) is denoted by \( \mathcal{E}_T \) and the unit outward normal on \( \partial T \) by \( n_T \).

Let \( \mathcal{E}_\Omega \), \( \mathcal{E}_D \) and \( \mathcal{E}_N \) be the collections of all edges of \( \mathcal{T}_h \) lying on \( \Omega \), \( \Gamma_D \) and \( \Gamma_N \), respectively, and set \( \mathcal{E}_h = \mathcal{E}_\Omega \cup \mathcal{E}_D \cup \mathcal{E}_N \). For each edge \( E \in \mathcal{E}_h \), we fix a unit normal vector \( n_E \) and define the jump of \( v \) across \( E = \partial T^+ \cap \partial T^- \) as

\[
[v] = v|_{T^+} - v|_{T^-},
\]
where \( n_E \) is directed from \( T^+ \) to \( T^- \). For a boundary edge \( E \in \mathcal{E}_D \cup \mathcal{E}_N \), \( n_E \) is taken to be outward to \( \partial \Omega \) and \( [v] \) is set to be zero.

Let \( P_k(T) \) be the space of all polynomials on \( T \) whose total degrees are less than or equal to \( k \) and let \( R.M(T) := \text{span}\{(1,0),(0,1),(y,-x)\} \) be the space of rigid body motions on \( T \). Notice that \( v \in R.M(T) \) if and only if \( \epsilon(v) = 0 \).

The Crouzeix–Raviart P1 nonconforming finite element space is defined as

\[
V_h = \left\{ v_h \in (L^2(\Omega))^2 : v_h|_T \in (P_1(T))^2 \quad \forall T \in \mathcal{T}_h \text{ and } \int_E [v_h] \, ds = 0 \quad \forall E \in \mathcal{E}_\Omega \right\},
\]
and the subspace of \( V_h \) with zero mean values on \( \mathcal{E}_D \) as

\[
V_{h,D} = \left\{ v_h \in V_h : \int_E v_h \, ds = 0 \quad \forall E \in \mathcal{E}_D \right\}.
\]

Then the stabilized P1 nonconforming FEM for the linear elasticity problem (1.3) reads as follows (cf. [5]): find \( u_h \in V_h \) such that \( \int_E (u_h - u_D) \, ds = 0 \) for all \( E \in \mathcal{E}_D \) and

\[
A_h(u_h, v_h) = (f, v_h)_\Omega \quad \forall v_h \in V_{h,D},
\]

where

\[
A_h(u_h, v_h) = (\mathcal{C} \epsilon_h(u_h), \epsilon_h(v_h))_\Omega + \sum_{E \in \mathcal{E}_\Omega} \mu \gamma h_E^{-1} \int_E [u_h] : [v_h] \, ds.
\]

Here \( \gamma > 0 \) is the stabilization parameter, \( h_E = \text{diam}(E) \), and \( \epsilon_h(\cdot) \) is the differential operator \( \epsilon(\cdot) \) applied piecewise over \( \mathcal{T}_h \).

2.2. Korn’s Inequality and related estimates. For a subdomain \( K \subset \Omega \), Korn’s inequality asserts that there exists a constant \( R_K > 0 \) depending only on \( K \) such that

\[
\| \nabla v \|_{0,K}^2 \leq R_K \| \epsilon(v) \|_{0,K}^2
\]

for all \( v \in (H^1(K))^2 \) with \( \int_K \text{curl} v \, dx = 0 \). Using the results in [6, 7], one can obtain the following computable upper bounds

\[
R_T \leq \frac{2}{\sin^2(\theta_{\text{min}}/4)}, \quad R_S \leq \frac{2}{\sin^2(\pi/8)} = 13.6569 \cdots
\]

for a triangle \( T \) with the minimum angle \( \theta_{\text{min}} \) and a square \( S \) (see [8]).
In deriving the a posteriori error estimator based on a nonsymmetric approximation of the stress tensor, we will need the operator \( \Pi_K : (H^1(K))^2 \to \mathcal{R}\mathcal{M}(K) \) defined by
\[
\int_K (v - \Pi_K v) \, dx = \int_K \text{curl}(v - \Pi_K v) \, dx = 0.
\]
The following estimates were first noted in [3, Appendix A]
\[
\|\nabla (v - \Pi_K v)\|_{0,K} \leq R_1 K \|\epsilon(v)\|_{0,K}, \quad (2.4)
\]
\[
\|v - \Pi_K v\|_{0,K} \leq R_1 h K \pi \|\epsilon(v)\|_{0,K}, \quad (2.5)
\]
where \( h_K \) is the diameter of \( K \). The first result is a direct consequence of Korn’s inequality (2.2) and the second result can be obtained by applying the Poincaré inequality [9] and then the first result.

3. \( H(\text{div}) \)-CONFORMING APPROXIMATION OF STRESS TENSOR

To recover a \( H(\text{div}) \)-conforming approximation of the stress tensor \( \sigma \), we first need to construct the normal stress approximation \( g_T \approx \sigma n_T |\partial T \) on every element \( T \in \mathcal{T}_h \) which fulfills the following equilibration conditions (see, e.g., [2, 10, 11])
\[
g_T|_E + g_{T_2}|_E = 0 \quad \text{for } E = \partial T_1 \cap \partial T_2, \quad (3.1)
\]
\[
g_T|_E = 0 \quad \text{for } E \in \mathcal{E}_T \cap \mathcal{E}_N, \quad (3.2)
\]
\[
\int_T f \cdot v \, dx + \int_{\partial T} g_T \cdot v \, ds = 0 \quad \text{for } v \in \mathcal{R}\mathcal{M}(T). \quad (3.3)
\]
For the stabilized \( P_1 \) nonconforming FEM (2.1), this can be done by an easy adaptation of the result for the Stokes problem given in [4].

**Definition 3.1.** For each \( T \in \mathcal{T}_h \) and \( E \in \mathcal{E}_T \), we define the normal stress approximation
\[
g_T|_E = \frac{1}{|E|} \left\{ \int_T \mathbb{C} \epsilon(u_h) \nabla \phi_E^{(T)} \, dx - \int_T f \phi_E^{(T)} \, dx \right.
\]
\[
+ \sum_{E' \in \mathcal{E}_T \setminus \mathcal{E}_E} \mu \gamma^{-1} \int_{E'} [u_h]_{\partial T} \phi_E^{(T)} \, ds \left\} - \mu \gamma^{-1} [u_h]|_{\partial T}|_E,
\]
where \( [v]|_{\partial T} \) is the jump of \( v \) from the interior to the exterior of \( T \)
\[
[v]|_{\partial T}|_E = (n_T|_E \cdot n_E)[v]|_E.
\]
Here \( \phi_E^{(T)} \) denotes the scalar-valued local basis function for \( \mathbb{P}_1(T) \) associated with the edge \( E \in \mathcal{E}_T \) satisfying
\[
\int_{E'} \phi_E^{(T)} \, ds = \delta_{E,E'}|_E \quad \forall E' \in \mathcal{E}_T.
\]
The proof of the equilibration conditions (3.1)–(3.3) can be done similarly to [4, Theorem 1].
Now we recover a nonsymmetric $H(\text{div})$-conforming approximation $\sigma_h$ of the stress tensor in the nonsymmetric Raviart–Thomas space of first order from the equilibrated normal stress approximation $g_T$ given in Definition 3.1 which is piecewise linear over $\partial T$. We remark that the same recovery process was exploited in [12] for a posteriori error estimation of the Stokes equation involving the nonsymmetric gradient tensor. Below $P_k$ denotes the $L^2$ projection onto the space of piecewise constant ($k = 0$) or linear ($k = 1$) functions over $T_h$.

**Definition 3.2.** For each element $T \in T_h$, we determine $\sigma^1_h|_T \in (P_1(T))^2 \times 2$ by the condition

$$\sigma^1_h|_T n_T = g_T \quad \text{on} \partial T$$

and define the stress tensor approximation

$$\sigma_h|_T = \sigma^1_h|_T - \frac{1}{3} \sum_{i=1}^3 P_1 f(x_i) \otimes (x - x_i) \lambda_i,$$

where $\{x_i\}_{i=1,2,3}$ are the vertices of $T$, $\{\lambda_i\}_{i=1,2,3}$ are the barycentric coordinates such that $\lambda_i(x_j) = \delta_{ij}$, and $(u \otimes v)_{ij} := u_i v_j$.

Since $(x - x_i) \lambda_i$ has vanishing normal components on $\partial T$, we immediately get

$$\sigma_h|_T n_T = g_T \quad \text{on} \partial T,$$

from which it follows by (3.1)–(3.2) that $\sigma_h$ is indeed $H(\text{div})$-conforming and $\sigma_h \cdot n|_{\Gamma_N} = 0$. Moreover, we have the following result.

**Theorem 3.3.** Let $\sigma_h$ be defined by Definition 3.2. Then we have

$$\text{div} \sigma_h + P_1 f = 0 \quad \text{and} \quad P_0 \sigma^A_S = 0.$$  \hfill (3.5)

**Proof.** The proof of the first result is essentially given in [12]. We recall the proof for the reader’s convenience. Taking $v \in (P_0(T))^2$ in (3.3), we can show that

$$\text{div} \sigma^1_h + P_0 f = 0.$$

Now use the identity

$$\lambda_i(x) = \nabla \lambda_i \cdot (x - x_i) + 1$$

to obtain

$$\text{div} \sigma_h = \text{div} \sigma^1_h - \frac{1}{3} \sum_{i=1}^3 P_1 f(x_i) \text{div} ((x - x_i) \lambda_i)$$

$$= -P_0 f - \frac{1}{3} \sum_{i=1}^3 P_1 f(x_i) (3\lambda_i - 1)$$

$$= -P_0 f - (P_1 f - P_0 f) = -P_1 f,$$
which is the first result. To prove the second result, we use (3.4), the first result and then (3.3) to obtain for $v_h \in \mathcal{R}_M(T)$

$$
\int_T \sigma_h : \nabla v_h \, dx = \int_{\partial T} \sigma_h n_T \cdot v_h \, ds - \int_T \text{div} \, \sigma_h \cdot v_h \, dx
= \int_{\partial T} g_T \cdot v_h \, ds + \int_T f \cdot v_h \, dx = 0.
$$

Now the second result is obtained by taking $v_h = (y, -x)$. \hfill \Box

4. A POSTERIORI ERROR ESTIMATION

In this section we will derive and analyze an a posteriori error estimator which gives a fully computable upper bound on the numerical error $u - u_h$ of the $P1$ nonconforming FEM (2.1) measured in the energy semi-norm

$$
\|v\|^2 := (C \epsilon_h(v), \epsilon_h(v))_\Omega = \sum_{T \in T_h} \|v\|^2_T, \quad \|v\|^2_T := (C \epsilon(v), \epsilon(v))_T.
$$

Following [3, 4], we decompose the error $u - u_h$ into two contributions

$$
u - u_h = (u - \xi) + (\xi - u_h),$$

where $\xi \in H^1_D(\Omega; u_D)$ is the solution of

$$
(C \epsilon(\xi), \epsilon(v))_\Omega = (C \epsilon_h(u_h), \epsilon(v))_\Omega \quad \forall v \in H^1_D(\Omega).
$$

By the Galerkin orthogonality $(C \epsilon_h(\xi - u_h), \epsilon(u - \xi))_\Omega = 0$, we obtain the Pythagorean relationship

$$
\|u - u_h\|^2 = \|u - \xi\|^2 + \|\xi - u_h\|^2.
$$

The following lemma presents the abstract error formulas for the two contributions on the right-hand side, the proof of which is a simple modification of that of [4].

**Lemma 4.1.** Let $\xi \in H^1_D(\Omega; u_D)$ be the solution of (4.1). Then we have

$$
\|u - \xi\| = \sup_{v \in H^1_D(\Omega)} \frac{(f, v)_\Omega - (C \epsilon_h(u_h), \epsilon(v))_\Omega}{\|v\|} \quad (4.2)
$$

and

$$
\|\xi - u_h\| = \inf_{\chi \in H^1_D(\Omega; u_D)} \|\chi - u_h\|. \quad (4.3)
$$

**Proof.** The first result is an immediate consequence of the equality

$$(C \epsilon(u - \xi), \epsilon(v))_\Omega = (f, v)_\Omega - (C \epsilon_h(u_h), \epsilon(v))_\Omega \quad \forall v \in H^1_D(\Omega),$$

while the second result follows from the fact that $\xi$ is the orthogonal projection of $u_h$ onto the hyperplane $H^1_D(\Omega; u_D)$ with respect to the energy semi-inner product $(C \epsilon_h(\cdot), \epsilon_h(\cdot))_\Omega$. \hfill \Box

By the estimate (4.3), $\|\xi - u_h\|$ measures the distance between the numerical solution $u_h$ and the continuous solution space $H^1_D(\Omega; u_D)$ in the energy semi-norm, commonly referred to as the *nonconforming error*, while $\|u - \xi\|$ is called the *conforming error*. 
4.1. **Conforming error estimator.** In this subsection we combine the equilibrated residual method and the hypercircle method to derive an estimator for the conforming error $\|u - \xi\|$ based on the estimate (4.2). To begin with, let us define the symmetric and antisymmetric parts of a tensor $\tau = \tau^S + \tau^{AS}$ by

$$\tau^S = \frac{\tau + \tau^T}{2} \quad \text{and} \quad \tau^{AS} = \frac{\tau - \tau^T}{2},$$

where $\tau^T$ is the transpose of $\tau$. It is obvious that $(\tau^S, \tau^{AS})_T = 0$ and $\|\tau^S\|_{0,T} + \|\tau^{AS}\|_{0,T} \leq \|\tau\|_{0,T}$.

Now we present the main result of this paper which generalizes Theorem 3.4 in [2] to the nonsymmetric $H(\text{div})$-conforming approximation of the stress tensor.

**Theorem 4.2.** Let $\sigma_h$ be defined by Definition 3.2 and let

$$\eta_{CF,T} := \|\mathcal{C}^{-1/2}(\sigma_h^S - \mathcal{C}\epsilon(u_h))\|_{0,T} + \left(\frac{R_T}{2\mu}\right)^{1/2} \frac{h_T}{\pi} \|f + \text{div} \sigma_h\|_{0,T}$$

and

$$\eta_{AS,K} := \left(\frac{R_K}{2\mu}\right)^{1/2} \|\sigma_h^{AS}\|_{0,K}$$

for an element $T$ and a subdomain $K$. Then we have

$$\|u - \xi\| \leq \left\{ \sum_{T \in T_h} (\eta_{CF,T} + \eta_{AS,T})^2 \right\}^{1/2}.$$

Moreover, if $\Omega$ is partitioned into the subdomains $\{\Omega_i\}_{i=1}^N$ such that each $\Omega_i$ is a union of elements in $T_h$, then we have

$$\|u - \xi\| \leq \left( \sum_{T \in T_h} \eta_{CF,T}^2 \right)^{1/2} + \left( \sum_{i=1}^N \eta_{AS,\Omega_i}^2 \right)^{1/2}.$$

**Proof.** We are going to estimate the right-hand side of (4.2). First note that for $v \in H^1_D(\Omega)$,

$$(f, v)_\Omega - (\mathcal{C}\epsilon(u_h), \epsilon(v))_\Omega = \sum_{T \in T_h} r_T(v),$$

where

$$r_T(v) := \int_T f \cdot v \, dx + \int_{\partial T} \sigma_h n_T \cdot v \, ds - \int_T \mathcal{C}\epsilon(u_h) : \epsilon(v) \, dx.$$

Using integration by parts, we get

$$r_T(v) = \int_T \sigma_h : \nabla v \, dx - \int_T \mathcal{C}\epsilon(u_h) : \epsilon(v) \, dx + \int_T (f + \text{div} \sigma_h) \cdot v \, dx$$

$$= \int_T (\sigma_h^S - \mathcal{C}\epsilon(u_h)) : \epsilon(v) \, dx + \int_T (f + \text{div} \sigma_h) \cdot v \, dx + \int_T \sigma_h^{AS} : \nabla v \, dx.$$
The first term is simply bounded by
\[
\int_T (\sigma^S_h - \mathcal{C}e(u_h)) : e(v) \, dx \leq \|\mathcal{C}^{-1/2}(\sigma^S_h - \mathcal{C}e(u_h))\|_{0,T} \|v\|_T.
\]

The second term is handled by using the first result of (3.5) and the estimate (2.5)
\[
\int_T (f + \text{div } \sigma_h) \cdot v \, dx = \int_T (f + \text{div } \sigma_h) \cdot (v - \Pi_T v) \, dx
\leq \|f + \text{div } \sigma_h\|_{0,T} R_T^{1/2} h_T \frac{1}{\pi} \|e(v)\|_{0,T}
\leq \left(\frac{R_T}{2\mu}\right)^{1/2} h_T \frac{1}{\pi} \|f + \text{div } \sigma_h\|_{0,T} \|v\|_T,
\]
where we used the inequality \(\|e(v)\|_{0,T} \leq \frac{1}{2\mu} \|\Pi_T v\|_T^2\). As a result, it follows that
\[
\int_T (\sigma^S_h - \mathcal{C}e(u_h)) : e(v) \, dx + \int_T (f + \text{div } \sigma_h) \cdot v \, dx \leq \eta_{CF,T} \|v\|_T.
\]

For the third term, we use the second result of (3.5) and the estimate (2.4) on each element \(T\) to obtain
\[
\int_T \sigma_h^{AS} : \nabla v \, dx = \int_T \sigma_h^{AS} : \nabla (v - \Pi_T v) \, dx \leq \left(\frac{R_T}{2\mu}\right)^{1/2} \|\sigma_h^{AS}\|_{0,T} \|v\|_T,
\]
which gives
\[
\sum_{T \in \mathcal{T}_h} r_T(v) \leq \sum_{T \in \mathcal{T}_h} (\eta_{CF,T} + \eta_{AS,T}) \|v\|_T \leq \left\{ \sum_{T \in \mathcal{T}_h} (\eta_{CF,T} + \eta_{AS,T})^2 \right\}^{1/2} \|v\|.
\]

Otherwise we can do the same thing on each subdomain \(\Omega_i\) to obtain
\[
\int_{\Omega_i} \sigma_h^{AS} : \nabla v \, dx = \int_{\Omega_i} \sigma_h^{AS} : \nabla (v - \Pi_{\Omega_i} v) \, dx \leq \left(\frac{R_{\Omega_i}}{2\mu}\right)^{1/2} \|\sigma_h^{AS}\|_{0,\Omega_i} \|v\|_{\Omega_i},
\]
which gives
\[
\sum_{T \in \mathcal{T}_h} r_T(v) \leq \sum_{T \in \mathcal{T}_h} \eta_{CF,T} \|v\|_T + \sum_{i=1}^N \eta_{AS,\Omega_i} \|v\|_{\Omega_i}
\leq \left(\sum_{T \in \mathcal{T}_h} \eta_{CF,T}^2 \right)^{1/2} \|v\| + \left(\sum_{i=1}^N \eta_{AS,\Omega_i}^2 \right)^{1/2} \|v\|.
\]
The proof is completed by invoking the estimate (4.2). \(\square\)

**Remark 4.3.** One can readily check that Theorem 4.2 is valid for any \(H(\text{div})\)-conforming approximation \(\sigma_h\) satisfying (3.5). In particular, the symmetric approximation of the stress tensor from \(g_T\) in the Arnold–Winther space [2] or in the Arnold–Douglas–Gupta space [3] yields a simpler error estimator with \(\sigma_h^{AS} = 0\). But the nonsymmetric approximation given in
Definition 3.2 requires much less computation and makes the implementation easier than the symmetric ones.

**Remark 4.4.** In practice, $\eta_{CF,T}$ and $\eta_{AS,K}$ are computed using the upper bounds (2.3). The second result of Theorem 4.2 may give sharper upper bounds, e.g., when every $\Omega_i$ is a square.

### 4.2. Nonconforming error estimator

Based on the estimate (4.3), a computable upper bound on the nonconforming error $\|\xi - u_h\|$ is obtained by choosing a suitable $\chi \in H^1_D(\Omega, u_D)$. Like in [3, 4], we simply choose the continuous piecewise linear or quadratic polynomial $e_{u_h}$ constructed by averaging the nodal values of $u_h$ at Lagrange nodes and enforcing the Dirichlet boundary condition at Dirichlet nodes, i.e., interpolating

$$
\tilde{u}_h(z) = \begin{cases} 
\frac{1}{\text{card}(\omega_z)} \sum_{T \in \omega_z} u_h|_T(z) & \text{for } z \in \mathcal{N}_h \setminus \mathcal{N}_{h,D}, \\
u_D(z) & \text{for } z \in \mathcal{N}_{h,D},
\end{cases}
$$

where $\mathcal{N}_h$ is the set of all linear or quadratic Lagrange nodes of $T_h$, $\mathcal{N}_{h,D} = \mathcal{N}_h \cap \Gamma_D$, and $\omega_z$ is the set of all elements in $T_h$ sharing the node $z$. The proof of the following theorem is trivial and thus omitted.

**Theorem 4.5.** Let $\tilde{u}_h$ be the continuous piecewise linear or quadratic polynomial constructed as above and let

$$
\eta_{NC,T} := \|\tilde{u}_h - u_h\|_T, \quad \text{osc}_T(u_D) := \inf_{\chi} \|\chi\|
$$

where $\tilde{u}_D := \tilde{u}_h|_{\Gamma_D}$ and the infimum is taken over all $\chi \in H^1(\Omega)$ such that

$$
\chi = \begin{cases} 
\nu_D - \tilde{u}_D & \text{on } \partial T \cap \Gamma_D, \\
0 & \text{on } \partial T \setminus \Gamma_D.
\end{cases}
$$

Then we have

$$
\|\xi - u_h\| \leq \left\{ \sum_{T \in T_h} (\eta_{NC,T} + \text{osc}_T(u_D))^2 \right\}^{1/2}.
$$

**Remark 4.6.** Since $\tilde{u}_D$ is the piecewise linear or quadratic Lagrange interpolant of $u_D$ over $\mathcal{E}_D$, the second term $\text{osc}_T(u_D)$ is called the data oscillation of $u_D$ on $\partial T \cap \Gamma_D$ and it can be shown that (cf. [3, 4])

$$
\text{osc}_T(u_D) \leq C \left( \sum_{E \in E_T \cap \mathcal{E}_D} S_E_T \|u_D - \tilde{u}_D\|^2_{1,E} \right)^{1/2}.
$$

This term can be estimated using an extension of the boundary data (4.4) as in [3] or may be regarded as a higher order perturbation and negligible if $u_D$ is piecewise smooth over $\mathcal{E}_D$. 

5. Numerical Results

In this section we present some numerical results to confirm the theory established in the previous section and illustrate the effectiveness of our error estimator. Since the domain \( \Omega \) is either a square (Example 1) or composed of three squares which are unions of elements in \( \mathcal{T}_h \) (Example 2), we can define the total error estimator in two ways (cf. Theorem 4.2)

\[
\eta_0^2 := \sum_{T \in \mathcal{T}_h} \left( \eta_{CF,T} + \eta_{AS,T} \right)^2 + \sum_{T \in \mathcal{T}_h} \eta_{NC,T}^2
\]

and

\[
\eta_1^2 := \left( \sum_{T \in \mathcal{T}_h} \eta_{CF,T}^2 \right)^{1/2} + \left( \frac{R_S}{2\mu} \| \sigma_A^S \|_{0,\Omega} \right)^2 + \sum_{T \in \mathcal{T}_h} \eta_{NC,T}^2,
\]

where the constants \( R_T \) and \( R_S \) are replaced by the upper bounds given in (2.3), respectively, and the data oscillation \( osc_T(u_D) \) is ignored. The nonconforming error estimator \( \eta_{NC,T} \) is computed using the piecewise quadratic polynomial. For comparison we also compute the total error estimator

\[
\eta_{sym}^2 := \sum_{T \in \mathcal{T}_h} \eta_{CF,T}^2 + \sum_{T \in \mathcal{T}_h} \eta_{NC,T}^2
\]

using the symmetric approximation of the stress tensor in the Arnold–Winther space (cf. [2]).

Example 1. Consider the following solution \( u = (u_1, u_2) \) on the unit square \( \Omega = (0, 1)^2 \)

\[
u_1(x, y) = \cos(2\pi x) \sin(2\pi y), \quad u_2(x, y) = -u_1(y, x)
\]

with the Lamé constants \( \mu = 1.0, \lambda = 5.0 \) and the corresponding body force \( f \). We impose the homogeneous Neumann boundary condition

\[\sigma \mathbf{n} = 0 \quad \text{on} \quad \Gamma_N = \{ (x, y) : x = 1, \ 0 \leq y \leq 1 \}\]

and the Dirichlet boundary condition on the remaining part \( \Gamma_D = \partial \Omega \setminus \Gamma_N \).

As the solution \( u \) is smooth, numerical experiments are performed on a sequence of uniform meshes obtained by first partitioning \( \Omega \) into equal squares of size \( h = \frac{1}{2^m} \) \( (m = 3, 4, 5, 6, 7, 8) \) and then dividing every square into two triangles along the diagonal of slope 1.

Numerical results are reported in Tables 1–2, where the three contributions of the total error estimator are defined as

\[
\eta_{CFF}^2 = \sum_{T \in \mathcal{T}_h} \eta_{CF,T}^2, \quad \eta_{AS}^2 = \sum_{T \in \mathcal{T}_h} \eta_{AS,T}^2 \quad \text{or} \quad \frac{R_S}{2\mu} \| \sigma_A^S \|_{0,\Omega}, \quad \eta_{NC}^2 = \sum_{T \in \mathcal{T}_h} \eta_{NC,T}^2
\]

and the effectivity index \( \theta \) is the ratio of the total error estimator to the actual error \( \| u - u_h \| \). We observe that \( \theta \) is always bigger than unity, which means that the total error estimator gives an upper bound on the actual error as predicted by the theory. Although the best result is obtained for the estimator \( \eta_{sym} \), our estimator \( \eta_1 \) seems to be competitive in terms of accuracy and computational cost. This is partly due to the fact that the common contribution \( \eta_{NC} \) is comparable to the sum \( \eta_{CF} + \eta_{AS} \).


Table 1. Mesh size, actual error, three contributions of $\eta_1$ and effectivity index for Example 1.

<table>
<thead>
<tr>
<th>$1/h$</th>
<th>$|u - u_h|$</th>
<th>$\eta_{CF}$</th>
<th>$\eta_{AS}$</th>
<th>$\eta_{NC}$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.802e+0</td>
<td>1.573e+0</td>
<td>1.997e+0</td>
<td>3.025e+0</td>
<td>2.597</td>
</tr>
<tr>
<td>16</td>
<td>8.998e-1</td>
<td>5.631e-1</td>
<td>1.063e+0</td>
<td>1.427e+0</td>
<td>2.404</td>
</tr>
<tr>
<td>32</td>
<td>4.492e-1</td>
<td>2.533e-1</td>
<td>5.424e-1</td>
<td>6.993e-1</td>
<td>2.358</td>
</tr>
<tr>
<td>64</td>
<td>2.244e-1</td>
<td>1.231e-1</td>
<td>2.732e-1</td>
<td>3.462e-1</td>
<td>2.345</td>
</tr>
<tr>
<td>128</td>
<td>1.121e-1</td>
<td>6.112e-2</td>
<td>1.370e-1</td>
<td>1.722e-1</td>
<td>2.341</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the error estimators $\eta_0$ and $\eta_{sym}$ for Example 1.

<table>
<thead>
<tr>
<th>$1/h$</th>
<th>$\eta_0$</th>
<th>$\eta_{CF}$</th>
<th>$\eta_{AS}$</th>
<th>$\theta$</th>
<th>$\eta_{sym}$</th>
<th>$\eta_{CF}$</th>
<th>$\eta_{AS}$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.573e+0</td>
<td>3.918e+0</td>
<td>3.468</td>
<td>2.291</td>
<td>2.809e+0</td>
<td>0.000e+0</td>
<td>2.151</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5.631e-1</td>
<td>2.085e+0</td>
<td>3.333</td>
<td>2.125</td>
<td>1.308e+0</td>
<td>0.000e+0</td>
<td>2.118</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2.533e-1</td>
<td>1.064e+0</td>
<td>3.310</td>
<td>2.125</td>
<td>6.498e-1</td>
<td>0.000e+0</td>
<td>2.118</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>1.231e-1</td>
<td>5.359e-1</td>
<td>3.307</td>
<td>2.119</td>
<td>3.259e-1</td>
<td>0.000e+0</td>
<td>2.117</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>6.112e-2</td>
<td>2.688e-1</td>
<td>3.308</td>
<td>2.118</td>
<td>1.635e-1</td>
<td>0.000e+0</td>
<td>2.117</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>3.050e-2</td>
<td>1.346e-1</td>
<td>3.309</td>
<td>2.117</td>
<td>8.188e-2</td>
<td>0.000e+0</td>
<td>2.117</td>
<td></td>
</tr>
</tbody>
</table>

Example 2. Let $(r, \theta)$ be the polar coordinates and consider the solution $u = (u_1, u_2)$ on the $\Gamma$-shaped domain $\Omega = (-1, 1)^2 \setminus [0, 1] \times [-1, 0]$ (cf. [2, 3])

\[
\begin{align*}
    u_1(r, \theta) &= \frac{1}{2\mu} r^\alpha \left( A \cos(\alpha \theta) - \cos((\alpha - 2)\theta) \right), \\
    u_2(r, \theta) &= \frac{1}{2\mu} r^\alpha \left( B \sin(\alpha \theta) + \sin((\alpha - 2)\theta) \right)
\end{align*}
\]

with the parameters chosen as

\[
\mu = 1.0, \quad \lambda = 5.0, \quad \alpha = 0.6, \quad A = 1.0, \quad B = \frac{2(3\mu + \lambda)}{(\lambda + \mu)\alpha} - A.
\]

The corresponding body force $f$ is then zero and the Dirichlet boundary condition is imposed on the whole boundary $\partial \Omega$.

As the solution $u$ is singular near the origin, we perform adaptive mesh refinement starting with the initial mesh shown in Fig. 1 and using the maximum criterion: mark an element $T \in T_h$ for refinement if

\[
\eta_T > 0.5 \max_{T' \in T_h} \eta_{T'},
\]

where the local error indicator $\eta_T$ is defined with the three contributions of $\eta_1$

\[
\eta_T := \left( \eta_{CF,T}^2 + \frac{R_s}{2\mu} \|\sigma_h \|_{0,T}^2 + \eta_{NC,T}^2 \right)^{1/2}.
\]
Two adapted meshes generated after 7 and 14 refinements are shown in the middle and right figures of Fig. 1. As expected, the mesh refinement is highly concentrated around the origin to resolve the singular behavior of \( u \). Furthermore, numerical results reported in Tables 3–4 confirm the theory and the effectiveness of our estimator \( \eta_1 \) also in the context of adaptive mesh refinement. Finally, it is found that the actual error attains the optimal order of convergence with respect to the number of unknowns \( N \)

\[
\|u - u_h\| = O(N^{-0.50}),
\]

demonstrating the efficiency of adaptive mesh refinement.
TABLE 4. Comparison of the error estimators $\eta_0$ and $\eta_{sym}$ for Example 2.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\eta_{ICF}$</th>
<th>$\eta_{AS}$</th>
<th>$\theta$</th>
<th>$\eta_{ICF}$</th>
<th>$\eta_{AS}$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>1.201e–1</td>
<td>6.461e–1</td>
<td>2.242</td>
<td>6.056e–1</td>
<td>0.000e+0</td>
<td>2.179</td>
</tr>
<tr>
<td>132</td>
<td>1.535e–1</td>
<td>6.612e–1</td>
<td>2.743</td>
<td>5.243e–1</td>
<td>0.000e+0</td>
<td>2.568</td>
</tr>
<tr>
<td>240</td>
<td>1.318e–1</td>
<td>5.259e–1</td>
<td>2.639</td>
<td>4.091e–1</td>
<td>0.000e+0</td>
<td>2.430</td>
</tr>
<tr>
<td>364</td>
<td>1.190e–1</td>
<td>4.299e–1</td>
<td>2.595</td>
<td>3.242e–1</td>
<td>0.000e+0</td>
<td>2.336</td>
</tr>
<tr>
<td>490</td>
<td>1.113e–1</td>
<td>3.767e–1</td>
<td>2.584</td>
<td>2.818e–1</td>
<td>0.000e+0</td>
<td>2.285</td>
</tr>
<tr>
<td>686</td>
<td>1.005e–1</td>
<td>3.382e–1</td>
<td>2.541</td>
<td>2.503e–1</td>
<td>0.000e+0</td>
<td>2.185</td>
</tr>
<tr>
<td>960</td>
<td>9.114e–2</td>
<td>3.025e–1</td>
<td>2.603</td>
<td>2.203e–1</td>
<td>0.000e+0</td>
<td>2.202</td>
</tr>
<tr>
<td>1432</td>
<td>7.667e–2</td>
<td>2.564e–1</td>
<td>2.690</td>
<td>1.827e–1</td>
<td>0.000e+0</td>
<td>2.247</td>
</tr>
<tr>
<td>2102</td>
<td>6.570e–2</td>
<td>2.229e–1</td>
<td>2.814</td>
<td>1.527e–1</td>
<td>0.000e+0</td>
<td>2.307</td>
</tr>
<tr>
<td>3472</td>
<td>5.214e–2</td>
<td>1.777e–1</td>
<td>2.797</td>
<td>1.215e–1</td>
<td>0.000e+0</td>
<td>2.278</td>
</tr>
<tr>
<td>4998</td>
<td>4.403e–2</td>
<td>1.514e–1</td>
<td>2.851</td>
<td>1.021e–1</td>
<td>0.000e+0</td>
<td>2.304</td>
</tr>
<tr>
<td>7400</td>
<td>3.718e–2</td>
<td>1.273e–1</td>
<td>2.915</td>
<td>8.438e–2</td>
<td>0.000e+0</td>
<td>2.336</td>
</tr>
<tr>
<td>11618</td>
<td>2.970e–2</td>
<td>1.013e–1</td>
<td>2.864</td>
<td>6.769e–2</td>
<td>0.000e+0</td>
<td>2.294</td>
</tr>
<tr>
<td>17010</td>
<td>2.468e–2</td>
<td>8.397e–2</td>
<td>2.885</td>
<td>5.581e–2</td>
<td>0.000e+0</td>
<td>2.306</td>
</tr>
<tr>
<td>25886</td>
<td>2.029e–2</td>
<td>6.953e–2</td>
<td>2.941</td>
<td>4.560e–2</td>
<td>0.000e+0</td>
<td>2.336</td>
</tr>
<tr>
<td>39068</td>
<td>1.665e–2</td>
<td>5.680e–2</td>
<td>2.943</td>
<td>3.715e–2</td>
<td>0.000e+0</td>
<td>2.334</td>
</tr>
</tbody>
</table>

REFERENCES


A ROBUST AND ACCURATE PHASE-FIELD SIMULATION OF SNOW CRYSTAL GROWTH

YIBAO LI\textsuperscript{1}, DONGSUN LEE\textsuperscript{1}, HYUN GEUN LEE\textsuperscript{1}, DARAE JEONG\textsuperscript{1}, CHAEYOUNG LEE\textsuperscript{1}, DONGGYU YANG\textsuperscript{2}, AND JUNSEOK KIM\textsuperscript{1}\textsuperscript{1}

\textsuperscript{1}DEPARTMENT OF MATHEMATICS, KOREA UNIVERSITY, SEOUL 136-701, REPUBLIC OF KOREA
\textit{E-mail address:} cfdkim@korea.ac.kr

\textsuperscript{2}SEOUL SCIENCE HIGH SCHOOL, SEOUL 110-530, REPUBLIC OF KOREA

ABSTRACT. In this paper we introduce 6-fold symmetry crystal growth using new phase-field models based on the modified Allen–Cahn equation. The proposed method is a hybrid method which uses both analytic and numerical solutions. We then show this method can be extended to $k$-fold case. The Wulff construction procedure is provided to understand and predict the shape of crystals. We also present a detailed mathematical proof of the validity of the Wulff construction. For computational results, we verify the accuracy and efficiency of the method for snow crystal growth.

1. INTRODUCTION

In nonlinear dynamical systems, the physics of phase transformations has attracted considerable interest. Crystal growth is an essential part of phase transformations from the liquid phase to the solid phase via heat transfer. To simulate crystal growth, cellular automaton [25, 44, 45, 46, 47], Monte-Carlo [29, 33], boundary integral [26, 27, 34, 36], front-tracking [1, 12, 21], level-set [6, 10, 17, 41], and phase-field [4, 5, 7, 8, 11, 14, 15, 16, 18, 28, 30, 31, 32, 35, 37, 39, 40, 43] methods have been developed. Also, many numerical methods such as explicit [12, 13, 16, 31, 39], mixed implicit-explicit [30, 40, 43], and adaptive methods [5, 7, 28, 32, 35] have been proposed for crystal growth problems.

Analysis of this paper using the phase field method extends our study to the various cases $k = 3, 4, 5, 6, \ldots$, and $n$ with $k$-fold symmetry. Beside that, one of the crucial to our crystal growth is a method of the multiple time-step algorithm that uses a larger time step for the flow-field calculations while reserving a finer time step for the phase-field evolution was proposed in [37]. Thus, we show that our scheme can give rise to many shapes with $n$-fold symmetry.

In particular, we focus on six-fold symmetric crystal growth, have some physical meaning that snow crystal while, mathematically, we can extend it to the $n$-fold case. In the six-fold case, water has the unique chemical property known as a hydrogen bond. The attractive interaction
between the hydrogen and oxygen atoms in different water molecules arranges the solid state water molecules to form a hexagonal shape. For such a reason, in specific temperature, snow crystal grows into six-fold symmetric crystal [24]. The thickness and width of snow crystal are in the ratio of 1 : 50, so snow crystal problems can be simplified into two-dimensional problems. We consider here the solidification of a pure substance from its supercooled melt in two-dimensional space.

In addition, we elaborate on the Wulff construction procedure for the equilibrium crystal shapes with a given interface energy function. We also present a detailed mathematical proof of the validity of the Wulff construction.

This paper is organized as follows: in Section 2, we briefly review basic theoretical concepts about the Wulff construction. The governing equations for crystal growth based on the phase-field medal are given in Section 3. In Section 4, we describe the computationally efficient operator splitting algorithm. In Section 5, we present numerical results of snow crystal growth simulations in 2D. Finally, conclusions are given in Section 6.

2. THE WULFF CONSTRUCTION

The equilibrium crystal shape is determined by minimizing the total interfacial free energy. We use the following \( k \)-fold symmetric interfacial energy equation:

\[
\epsilon(\theta) = \epsilon_0 \left[ 1 + \epsilon_k \cos(k\theta) \right],
\]

where \( \epsilon_0 \) is the mean interfacial tension and \( 0 \leq \epsilon_k < 1 \) is the anisotropy parameter.

![Figure 1](image)

**Figure 1.** The Wulff construction. (a) Interfacial free-energy density \( \epsilon(\theta) \) in the polar coordinates. (b) Equilibrium crystal shape (bold line) for \( k = 6 \), \( \epsilon_0 = 1 \), and \( \epsilon_6 = 0.1 \).

In this paper, we focus on \( k = 6 \) case. The equilibrium shape is easily constructed by the Wulff’s theorem [42]. We describe the construction of the equilibrium shape geometrically [3].
Let $M = (\epsilon(\theta), \theta)$ be a point on the interfacial energy function in the polar coordinates (see Fig. 1(a)). The construction starts from the origin $O$ and draw the line segment $OM$ to the point $M$. Draw the perpendicular line $\overrightarrow{AB}$ to the line segment $OM$. Then the inner convex hull made from all such perpendiculars is an equilibrium crystal shape as shown in Fig. 1(b).

Conversely, let us assume the equilibrium shape is known and $(r, \theta)$ be the polar coordinates of a point $T$ of the crystal boundary $S$, that is, $T = (r, \theta)$. And let $T = (x(\phi), y(\phi))$ be the corresponding Cartesian coordinates, where $\phi$ is a parameter and is the angle between $x$-axis and the perpendicular line to the tangent line $\overrightarrow{AB}$ at the point $T$. Let $M$ be the intersection point of the line $\overrightarrow{AB}$ and the perpendicular line containing the origin to $\overrightarrow{AB}$. Let the length of the line segment $OM$ be $p(\phi)$. In Fig. 2, we can see these parameter definitions. Then $p(\phi)$ can be obtained from the right triangle $\triangle OTM$:

\[ p(\phi) = r \cos(\phi - \theta) = r \cos \phi \cos \theta + r \sin \phi \sin \theta = x(\phi) \cos \phi + y(\phi) \sin \phi. \quad (2.1) \]

We can express $(x(\phi), y(\phi))$ in terms of $p(\phi)$. First, take a derivative to $p(\phi)$, then we have

\[ p_\phi(\phi) = x_\phi(\phi) \cos \phi - x(\phi) \sin \phi + y_\phi(\phi) \sin \phi + y(\phi) \cos \phi. \quad (2.2) \]

Since the normal vector $(\cos \phi, \sin \phi)$ and the tangent vector $(x_\phi, y_\phi)$ are orthogonal, that is, $(\cos \phi, \sin \phi) \cdot (x_\phi, y_\phi) = 0$, we can simplify Eq. (2.2) as

\[ p_\phi = -x \sin \phi + y \cos \phi. \quad (2.3) \]
Now, by solving Eqs. (2.1) and (2.3) we have

\[ x(\phi) = p(\phi) \cos \phi - p_\phi(\phi) \sin \phi, \quad y(\phi) = p(\phi) \sin \phi + p_\phi(\phi) \cos \phi. \]  \tag{2.4}

Let \( F \) and \( A \) be the total edge free energy and the area of crystal, respectively and be defined as

\[ F = \int \epsilon(\phi) \sqrt{(x_\phi(\phi))^2 + (y_\phi(\phi))^2} \, d\phi, \]  \tag{2.5}

\[ A = \frac{1}{2} \int (x(\phi)y_\phi(\phi) - y(\phi)x_\phi(\phi)) \, d\phi. \]  \tag{2.6}

Using Eq. (2.4), we can rewrite Eqs. (2.5) and (2.6) in the form

\[ F = \int \epsilon(\phi)(p(\phi) + p_\phi(\phi)) \, d\phi, \]

\[ A = \frac{1}{2} \int p(\phi)(p(\phi) + p_\phi(\phi)) \, d\phi. \]

We want to minimize \( F \) with subject to a constant area constraint of \( A \). Using the Lagrange multiplier \( \lambda \), we seek to minimize

\[ F + \lambda A = \int \left( \epsilon(\phi) + \frac{\lambda}{2} p(\phi) \right) (p(\phi) + p_\phi(\phi)) \, d\phi. \]

And then, the Euler–Lagrange equation is

\[ \frac{\partial Q}{\partial p} - \frac{d}{d\phi} \left( \frac{\partial Q}{\partial p_\phi} \right) + \frac{d^2}{d\phi^2} \left( \frac{\partial Q}{\partial p_\phi} \right) = 0, \]  \tag{2.7}

where

\[ Q = \left( \epsilon + \frac{\lambda}{2} p \right) (p + p_\phi). \]  \tag{2.8}

From these two Eqs. (2.7) and (2.8), we get

\[ p + p_\phi = -\frac{1}{\lambda} (\epsilon + \epsilon_\phi). \]  \tag{2.9}

A solution of differential equation (2.9) is

\[ p(\phi) = -\frac{1}{\lambda} \epsilon(\phi). \]

This result implies that in a crystal at equilibrium, the distances of the faces from the center of the crystal are proportional to their surface free energies per unit area [3].

For large \( \epsilon_6 \) values, the crystal shape will be energy minimizing when certain orientations are missing. Missing orientations occur when the polar plot of \( r = 1/\epsilon(\theta) \) changes convexity [9]. The curvature of a polar plot \( r(\theta) \) is \( \kappa = (r^2 + 2r^3 - rr_\theta^2)/(r^2 + r_\theta^2)^{3/2} \). For \( r(\theta) = 1/\epsilon(\theta) \), the curvature is \( \kappa = (\epsilon + \epsilon_\theta)/[1 + (\epsilon_6)^2]^{3/2} \). So convexity changes whenever

\[ \epsilon + \epsilon_\theta = \epsilon_0(1 - 35\epsilon_6 \cos 6\theta) < 0. \]
If values of $\epsilon_6$ are larger than $1/35$, then missing orientations occur. In other words, some orientations do not appear on the equilibrium shape of a crystal. Figure 3 shows the 6-fold Wulff equilibrium shapes $((x(\phi), y(\phi))$ for $0 \leq \phi \leq 2\pi)$ with two different $\epsilon_6$ values: (a) $\epsilon_6 = 1/50$ and (b) $\epsilon_6 = 1/10$ (which shows the missing orientation). Figure 4 shows the trace of $(x(\phi), y(\phi))$ with different intervals. $\phi_m$ is defined as the smallest non-zero value which satisfies $y(\phi_m) = 0$.

![Figure 3. The 6-fold Wulff equilibrium shapes with two different $\epsilon_6$ values.](image)

![Figure 4. Trace of $(x(\phi), y(\phi))$ with different intervals and $y(\phi_m) = 0$.](image)
3. The phase-field model

The phase-field model for the crystal growth is given by

\[
e^2(c) \frac{\partial c}{\partial t} = \nabla \cdot \left( e^2(c) \nabla c \right) + \left( c - \lambda U(1 - c^2) \right) (1 - c^2)
\]

\[
+ \left( |\nabla c^2| \frac{\partial e(c)}{\partial c_x} \right)_x + \left( |\nabla c^2| \frac{\partial e(c)}{\partial c_y} \right)_y
\]

(3.1)

\[
\frac{\partial U}{\partial t} = D\Delta U + \frac{1}{2} \frac{\partial c}{\partial t},
\]

where \( c \) is the order parameter, \( e(c) \) is the anisotropic function, \( \lambda \) is the dimensionless coupling parameter, and \( U = c_p(T - T_M)/L \) is the dimensionless temperature field. Here \( c_p \) is the specific heat at constant pressure, \( T_M \) is the melting temperature, \( L \) is the latent heat of fusion, \( D = \alpha \tau_0 / \epsilon_0^2 \), \( \alpha \) is the thermal diffusivity, \( \tau_0 \) is the characteristic time, and \( \epsilon_0 \) is the characteristic length. The order parameter is defined by \( c = 1 \) in the solid phase and \( c = -1 \) in the liquid phase. The interface is defined by \( c = 0 \) and \( \lambda \) is given as \( \lambda = D/a_2 \) with \( a_2 = 0.6267 \) [15, 16]. We define a normal vector of \( c \) as \((c_x, c_y)\) and an angle between normal vector and \( x \)-axis as \( \phi \) that satisfies \( \tan \phi = c_y / c_x \). Then by replacing \( e(c) \) with \( e(\phi) = \epsilon_0 (1 + \epsilon_6 \cos(6\phi)) \), we can simplify the following terms in Eq. (3.1):

\[
\left( |\nabla c^2| e(\phi) \frac{\partial e(\phi)}{\partial c_x} \right)_x = \left( e_x^2 + c_y^2 \right) \frac{\partial e(\phi)}{\partial c_x} - \left( c_y^2 \left( 1 + \epsilon_6 \right) \right)_x = - \left( e' \epsilon(\phi) c_y \right)_x.
\]

In a similar way, we get

\[
\left( |\nabla c^2| e(\phi) \frac{\partial e(\phi)}{\partial c_y} \right)_y = \left( e' \epsilon(\phi) c_x \right)_y.
\]

Hence we can rewrite the governing equations of 6-fold symmetric crystal growth as following:

\[
e^2(\phi) \frac{\partial c}{\partial t} = \nabla \cdot \left( e^2(\phi) \nabla c \right) + \left( c - \lambda U(1 - c^2) \right) (1 - c^2)
\]

\[
- \left( e' \epsilon(\phi) c_y \right)_x + \left( e' \epsilon(\phi) c_x \right)_y
\]

(3.2)

\[
\frac{\partial U}{\partial t} = D\Delta U + \frac{1}{2} \frac{\partial c}{\partial t}.
\]

(3.3)

4. Numerical solution

In this section, we propose a robust hybrid numerical method for crystal growth simulation. For simplicity of exposition we shall discretize Eqs. (3.2) and (3.3) in two-dimensional space, i.e., \( \Omega = (-l_1, l_1) \times (-l_2, l_2) \). Let \( N_x \) and \( N_y \) be positive even integers, \( h = 2l_1/N_x \) be the uniform mesh size, and \( \Omega_h = \{(x_i, y_j) : x_i = (i - 0.5)h, y_j = (j - 0.5)h, 1 \leq i \leq N_x, 1 \leq j \leq N_y \} \) be the set of cell-centers. Let \( c^n_{ij} \) be approximations of \( c(x_i, y_j, n\Delta t) \), where \( \Delta t = T/N_t \) is the time step, \( T \) is the final time, and \( N_t \) is the total number of time steps. The discrete differentiation operator is \( \nabla_d c_{ij} = (c_{i+1,j} - c_{i,j}, c_{i,j+1} - c_{i,j-1})/(2h) \). We
then define the discrete Laplacian by $\Delta_d c_{ij} = (c_{i+1,j} + c_{i-1,j} - 4c_{ij} + c_{i,j+1} + c_{i,j-1})/h^2$. We discretize Eqs. (3.2) and (3.3):

\[
e^2(\phi^n) \frac{c^{n+1} - c^n}{\Delta t} = e^2(\phi^n) \Delta_d c^{n+1,2} + 2e(\phi^n) \nabla_d e(\phi^n) \cdot \nabla_d c^n - F'(c^{n+1}) - 4\lambda U^n F(c^{n+1,1}) - (e'(\phi) \cdot e(\phi)c_y)_x + (e'(\phi) \cdot e(\phi)c_x)_y^n,
\]

\[
U^{n+1} - U^n = D\Delta_d U^{n+1} + \frac{c^{n+1} - c^n}{2\Delta t},
\]

where $F(c) = 0.25(c^2 - 1)^2$ and $F'(c) = c(c^2 - 1)$. Here $c^{n+1,k}$ for $k = 1, 2$ are defined in the operator splitting scheme. We propose the following operator splitting scheme:

\[
e^2(\phi^n) \frac{c^{n+1,1} - c^n}{\Delta t} = 2e(\phi^n) \nabla_d e(\phi^n) \cdot \nabla_d c^n - (e'(\phi) \cdot e(\phi)c_y)_x + (e'(\phi) \cdot e(\phi)c_x)_y^n,
\]

\[
e^2(\phi^n) \frac{c^{n+1,2} - c^{n+1,1}}{\Delta t} = e^2(\phi^n) \Delta_d c^{n+1,2} - 4\lambda U^n F(c^{n+1,1}),
\]

\[
e^2(\phi^n) \frac{c^{n+1} - c^{n+1,2}}{\Delta t} = -F'(c^{n+1}). \tag{4.1}
\]

We can solve Eq. (4.1) analytically by the method of separation of variables [22, 23]. The solution is given as follows:

\[
e^{n+1} = \frac{c^{n+1,2}}{\sqrt{e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}} + (c^{n+1,2})^2 \left(1 - e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}}\right)}}.
\]

Finally, the proposed scheme can be written as follows:

\[
e(\phi^n) \frac{c^{n+1,1} - c^n}{\Delta t} = 2e(\phi^n)_x c^n_x + 2e(\phi^n)_y c^n_y - (e'(\phi) \cdot e(\phi)c_y)_x + (e'(\phi) \cdot e(\phi)c_x)_y^n,
\]

\[
e^2(\phi^n) \frac{c^{n+1,2} - c^{n+1,1}}{\Delta t} = e^2(\phi^n) \Delta_d c^{n+1,2} - 4\lambda U^n F(c^{n+1,1}), \tag{4.2}
\]

\[
e^{n+1} = \frac{c^{n+1,2}}{\sqrt{e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}} + (c^{n+1,2})^2 \left(1 - e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}}\right)}}.
\]

\[
U^{n+1} - U^n = D\Delta_d U^{n+1} + \frac{c^{n+1} - c^n}{2\Delta t}. \tag{4.3}
\]

Equations (4.2) and (4.3) can be solved by a multigrid method [2, 38].
5. Numerical results

In this section we perform numerical experiments for two-dimensional solidification to validate that our proposed scheme is accurate, efficient, and robust. Unless otherwise specified, we take the initial state as
\[ c(x, y, 0) = \tanh \left( \frac{R_0 - \sqrt{x^2 + y^2}}{\sqrt{2}} \right) \quad \text{and} \quad U(x, y, 0) = \begin{cases} 0 & \text{if } c > 0 \\ \Delta & \text{else.} \end{cases} \]

The zero level set \((c = 0)\) represents a circle of radius \(R_0\). From the dimensionless variable definition the value \(U = 0\) corresponds to the melting temperature of the pure material, while \(U = \Delta\) is the initial undercooling. The capillary length, \(d_0\), is defined as
\[ d_0 = \frac{a_1}{\lambda} \quad [4, 20, 32] \]
with \(a_1 = 0\).

5.1. Convergence test. To obtain an estimate of the convergence rate, we perform a number of simulations for 6-fold crystal growth problem on a set of increasingly finer grids. The computational domain is \(\Omega = (-100, 100)^2\) and we take \(R_0 = 15d_0\), \(\epsilon = 0.02\), and \(\Delta = -0.55\). The numerical solutions are computed on the uniform grids \(h = \frac{200}{2^n}\) and with corresponding time steps \(\Delta t = \frac{0.6}{2^n-8}\) for \(n = 8, 9, 10,\) and \(11\). The calculations are run up to time \(T = 150\). We define the error to be the discrete of \(l_2\)-norm of the difference between that grid and the average of the next finer grid cells covering it:
\[ e_{h/2} \approx c_{h/2} - \frac{c_h + c_{h/2} + c_{2h} - c_{2h/2}}{4}. \]

The rate of convergence is defined as:
\[ \log_2 \left( \frac{\| e_{h/2} \|_2}{\| e_{h} \|_2} \right). \]

The errors and rates of convergence are given in Table 1. The results suggest that the scheme is indeed second order accurate in space. Figure 5 shows the convergence of numerical results under mesh refinement.

<table>
<thead>
<tr>
<th>(256 - 512)</th>
<th>Rate</th>
<th>(512 - 1024)</th>
<th>Rate</th>
<th>(1024 - 2048)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.477E-4</td>
<td>1.96</td>
<td>1.405E-4</td>
<td>2.01</td>
<td>3.487E-5</td>
</tr>
</tbody>
</table>

Next, we consider the evolution of the interface with different time steps in order to investigate the effect of time step. A \(1024 \times 1024\) mesh is used on the domain \(\Omega = (-200, 200)^2\) with \(R_0 = 50d_0\), \(\epsilon = 0.02\), and \(\Delta = -0.55\). Figure 6(a) shows the interfaces at time \(T = 1200\) with different time steps \(\Delta t = 0.6, 0.3,\) and \(0.15\). Figure 6(b) shows the velocity of the tip versus time. For the calculation of the crystal tip velocity, refer to Ref. [22]. The velocity \(V\) of the tip at time \(T = 1200\) versus time step is shown in Fig. 6(c). Here, we define the error between the fitting velocity \(\tilde{V}\) and \(V\) as \(E_i = |\tilde{V}_i - V_i|/V_i\). In Fig. 6(c), the linear fit \(\tilde{V}\) is done using the MATLAB function “polyfit” and the errors on the index \(i\) are calculated by the MATLAB function “polyval” on the results of the linear fit. In this test, the \(l_2\) error is 0.54%.
Figure 5. Convergence of numerical results under mesh refinement.

Therefore the results suggest that the convergence rate of the tip velocity is linear with respect to the time step.

5.2. Stability test. In this section, we perform a number of simulations on a set of increasingly finer grids to show that our proposed method is more stable than the previous methods which suffer from time restrictions $\Delta t \leq O(h^2)$ for stability. The computational domain is $\Omega = (-200, 200)^2$ and we take $R_0 = 15d_0$, $\epsilon_6 = 0.02$, and $\Delta = -0.55$. The numerical solutions are computed on the uniform grids $h = 400/2^n$ with corresponding time steps $\Delta t = 3h$ for $n = 8, 9, \text{and } 10$. Figure 7 shows the crystal growth with different time steps at $T = 70.31$. In general, large time steps may cause large truncation errors. However, as can be seen in Fig. 7, we obtain stable solutions with large time steps.

Next, we calculate the maximum $\Delta t$ corresponding to different spatial grid sizes $h$ so that stable solutions can be computed after 20 time step iterations. The results are shown in Table 2 and we obtain stable solutions for all three mesh sizes. Note that there is a linear relation between the time step and mesh sizes. Thus, for finer mesh sizes we may use larger time steps than previous conventional methods.

Table 2. Stability constraint of $\Delta t$ for the proposed scheme.

<table>
<thead>
<tr>
<th>Mesh size</th>
<th>$h = 400/256$</th>
<th>$h = 400/512$</th>
<th>$h = 400/1024$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time step</td>
<td>$\Delta t \leq 12h$</td>
<td>$\Delta t \leq 10h$</td>
<td>$\Delta t \leq 8h$</td>
</tr>
</tbody>
</table>

5.3. Effect of $\epsilon_6$. To investigate the effect of $\epsilon_6$, we consider the evolution of the interface with different $\epsilon_6 = 0.002, 0.02, \text{and } 0.05$. A $1024 \times 1024$ mesh is used on the domain $\Omega =$
\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\includegraphics[width=\textwidth]{figure6a.png}
\caption{(a) The interfaces at $T = 1200$ for different time steps. (b) shows the velocity of the tip versus time. (c) The numerical experimental and linear fitting velocities versus time step.}
\end{subfigure}
\begin{subfigure}{0.45\textwidth}
\includegraphics[width=\textwidth]{figure6b.png}
\end{subfigure}
\caption{Figures 6(a), (b), and (c) are the evolution of crystal growth with $\epsilon_6 = 0.002$, 0.02, and 0.05, respectively. As advised in the previous paper, if $\epsilon_6 < \frac{1}{35}$, all of tangent planes lie outside and all orientations appear on the equilibrium shape. Detail view is drawn in Fig. 8(a). Otherwise, there is missing orientations shown in Fig. 8(c). While if $\epsilon_6$ is not more smaller than \frac{1}{35}, the crystal also works well shown in Fig. 8(b). Thus the Wulff construction is not strictly correlated with $\epsilon_6$ in crystal growth, but provide guidelines for parameter selection.}
\end{figure}

5.4. Effect of undercooling. Now we investigate the effects of undercooling of the initial solid seed. For each test, a $1024 \times 1024$ mesh is used on the domain $\Omega = (-200, 200)^2$ and we choose $R_0 = 15d_0$, $\epsilon_6 = 0.02$, $\Delta t = 0.3$, and $T = 1080$. Figure 9 shows sequences of
Figure 7. The stability of crystal growth with different mesh sizes: (a) 256 × 256 mesh (Δt = 4.68), (b) 512 × 512 mesh (Δt = 2.34), and (c) 1024 × 1024 mesh (Δt = 1.17).

Figure 8. The effect of $\epsilon_6$. (a), (b), and (c) are the evolution of crystal growth with $\epsilon_6 = 0.002, 0.02, \text{ and } 0.05$, respectively. The times are $t = 0, 120, 240, 360, 480, 600, 720, 840, 960, 1080, \text{ and } 1200$.

Interfaces with different undercooling sizes $\Delta = -0.45, \Delta = -0.55, \text{ and } \Delta = -0.65$. We observe that the large initial undercooling causes the dendrite to grow faster.

Figure 9. Sequences of interfaces with different undercooling sizes $\Delta = -0.45, \Delta = -0.55, \text{ and } \Delta = -0.65$. 
5.5. \textit{k-fold symmetric crystal growth}. If we set the energy function by $\epsilon(\phi) = \epsilon_0(1 + \epsilon_k \cos(k\phi))$, then our proposed method can simulate the $k$-fold crystal growth in general. To show this, we simulate sequences of computational experiments of $k$-fold symmetric crystal growth for $k = 4, \ldots, 9$. A $1024 \times 1024$ mesh is used on the domain $\Omega = (-200, -200)^2$ and we take $R_0 = 15d_0$, $\Delta = -0.55$, and $\Delta t = 0.3$. Note that we use $\epsilon_k = 1/(k^2 - 1)$ to respond to the Wulff’s algorithm. The evolutions for each $k$ are shown in Fig. 10.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig10}
\caption{The evolutions of $k$-fold crystal growth after time: (a) $T = 720$, (b) $T = 1200$, (c) $T = 1680$, (d) $T = 2160$, (e) $T = 2520$, and (f) $T = 2880$.}
\end{figure}

5.6. \textbf{Comparison with the previous study}. An isotropic finite-difference scheme for simulating 6-fold symmetric dendritic solidification is presented in [19]. The author showed that the stability criterion becomes $\Delta t \leq (3/8)h^2$. But, as we can see in Section 5.2, the time restriction of our proposed method is $\Delta t \sim O(h)$. In order to show the improvement of our proposed method, we use the same numerical parameters as in [19], e.g., $\lambda = 1.7680$, $\epsilon_0 = 1.1312$, $\epsilon_6 = 0.05$, $D = 2$, and $R_0 = 5$. Note that in [19], the author took the step size as $h = 0.4$ in the progressively increased mesh sizes as $500 \times 500$ for $0 \leq t \leq 150$, to $800 \times 800$ for $150 \leq t \leq 250$, and to $1200 \times 1200$ for $250 \leq t \leq 400$. Here we take a $1280 \times 1280$ mesh size. This simulation is run up to $T = 400$ with $\Delta t = 0.2$. Our proposed method took about only 5 hours of CPU time, which is drastically reduced faster than the CPU time (1000 hours) in [19].
6. Conclusion

In this paper we presented an accurate and efficient numerical method for phase-field models of $k$-fold snow crystal growth. We described the Wulff construction procedure for the equilibrium crystal shapes with a given interface energy function. For the interfacial energy larger than a particular value, convexity changes and missing orientations occur. Focusing on 6-fold symmetric shape, we calculated the particular value. We also provided a detailed mathematical proof of the validity of the Wulff construction. The proposed method is a hybrid method which uses both analytic and numerical solutions. We extended the model to $k$-fold symmetric crystal growth. Computational results showed the accuracy and efficiency of the method.

Acknowledgments

This work was supported by Seoul Science High School R&E program in 2011. The authors wish to thank the reviewers for the constructive and helpful comments on the revision of this article.

References


SURFACE RECONSTRUCTION FROM SCATTERED POINT DATA ON OCTREE

CHANGSOO PARK\textsuperscript{1}, CHOHONG MIN\textsuperscript{2}, AND MYUNGJOO KANG\textsuperscript{1,}\textsuperscript{1}

\textsuperscript{1}\textsc{Department of Mathematical Sciences, Seoul National University, South Korea}
\textit{E-mail address:} winspark, mkang@snu.ac.kr

\textsuperscript{2}\textsc{Department of Mathematics, Ewha Womans University, South Korea}
\textit{E-mail address:} chohong@ewha.ac.kr

\textbf{ABSTRACT.} In this paper, we propose a very efficient method which reconstructs the high resolution surface from a set of unorganized points. Our method is based on the level set method using adaptive octree. We start with the surface reconstruction model proposed in [20]. In [20], they introduced a very fast and efficient method which is different from the previous methods using the level set method. Most existing methods\cite{21, 22} employed the time evolving process from an initial surface to point cloud. But in [20], they considered the surface reconstruction process as an elliptic problem in the narrow band including point cloud. So they could obtain very speedy method because they didn’t have to limit the time evolution step by the finite speed of propagation.

However, they implemented that model just on the uniform grid. So they still have the weakness that it needs so much memories because of being fulfilled only on the uniform grid. Their algorithm basically solves a large linear system of which size is the same as the number of the grid in a narrow band. Besides, it is not easy to make the width of band narrow enough since the decision of band width depends on the distribution of point data. After all, as far as it is implemented on the uniform grid, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically.

We resolve it by adapting octree data structure\cite{12, 11} to our problem and by introducing a new redistancing algorithm which is different from the existing one\cite{19}.

1. INTRODUCTION

Surface reconstruction from a set of unorganized points also known as reverse engineering is a very popular and challenging problem in many area such as medical image, computer graphics, mechanical engineering. Since problem to find the connection between point data is ill-posed, we can just approximate the underlying surface. Although there are so many methodologies to do this, these methods are broadly classified into two categories: explicit representation and implicit representation. The former consists of Delaunay triangulations,
Voronoi diagrams, the power crust[3] and so on. The latter includes Hoppe’s methods[8], moving least squares[1, 6], the radial basis function[4], the level set method[21, 22], etc.

The explicit method is the method which finds a graph to connect every pair of point data. The \( \alpha \)-shape[5], the \( \beta \)-skeleton[2] and the power crust[3] are examples of explicit representation. These graphs are not only based on the Voronoi diagram and the Delaunay triangulation but also the subsets of the Delaunay triangulation.

The implicit representation uses a scalar function which describe a surface as its zero level set.

The first and most famous algorithm using implicit representation is by Hoppe et al.[8]. It firstly approximates the tangent plane on a point using least squares on \( k \) nearest neighboring points. Then it considers the signed distance from the point to its projection onto the tangent plane as the signed distance on the whole domain.

Radial basis function is very well-known function in many fields. In [4], the surface reconstruction using radial basis function was introduced. While it can get the smooth surface, it has some limitations. It should solve the linear system \( Ax = b \) which the matrix \( A \) is dense and ill-conditioned. Although they employed FMM(Fast Multipole Method) for evaluations of energy minimization, it is rather slow compared with other methods. Also, it needs additional information called off-surface points besides given point data.

The first application of the level set method to our problem is the result by Zhao et al. in [21] and [22]. Their models make use of the surface evolving from an closed surface of which inside region includes all point data as the level set method comes originally from computational fluid dynamics. The level set method has the advantages of handling topological changes easily and not requiring the normal information on point data. But the previous models have the bounds of speed from the CFL condition because they use time-dependent PDEs such as nonlinear parabolic PDE and linear advection equation.

In [20], Ye et al. approached our problem from a different standpoint. They considered the surface reconstruction as a Poisson problem. Because they became free from the CFL condition, they could improve the efficiency of the process. However, their algorithm basically solves a large linear system of which size is the same as the number of the grid in a narrow band and is fulfilled only on the uniform grid. Consequently, it requires so much memories. Therefore, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically.

In our problem, the part to be described accurately is the surface passing through points i.e. the interface of the level set function rather than the whole domain. Therefore we have only to capture the details near the interface. That is, we need the multi-resolution approach. The adaptive grid can resolve that problem with refining only near the interface. The adaptive grid is very efficient and prevalent in many fields such as computer graphics and scientific computing and computational fluid dynamics where they need to reduce much computational costs. Also in surface reconstruction, there are some researches[13, 17] using the adaptive grid in order to generate high resolution. We employ the octree which is a sort of the adaptive grid. The Poisson’s equation can be effectively solved on the octree. There are several Poisson solvers on the octree developed up to date.
In [15], Popinet proposed a second order non-symmetric numerical method to solve the incompressible Euler equations on octree. In his method, the pressure is sampled at the center of each cell and the discretization of the Poisson equation requires interpolation procedures involving the pressure values at several adjacent cells. Consequently, this discretization requires finite difference method to use large support.

In [10], Losasso et al. proposed a first order Poisson solver using octree data structures and applied it to the Navier-Stokes equation. They stored data except the pressure at nodes and the pressure at the center of the cell. They perturbed the location of the pressure by a quantity proportional to the size of a grid cell in order to obtain a symmetric linear system. So most approximations in their scheme is second order accurate but approximations to the gradient of the pressure is first order accurate.

In [11], Min et al. proposed a second order accurate finite difference discretization for the variable coefficient Poisson equation on non-graded grids, which yields second order accuracy covering the solution’s gradients. The scheme employs sampling the solution at the nodes of a cell. The discretization at one cell’s node only uses nodes of two or three adjacent cells, producing schemes that are straightforward to implement.

Adaptive level set method was successfully developed in Min’s paper [11]. Ghost nodes had been treated with linear interpolation. Without increasing the support of numerical methods, they showed a possible quadratic interpolation. We apply it to our problem.

The outline of the paper is as follows. In the next section, we explain mathematical models using level set method. In section 3, we present numerical method to reconstruct a surface from point cloud on the octree. In section 4, we show experimental results to demonstrate that our method is efficient.

2. Mathematical models

The most well-known variational model of surface reconstruction using the level set method is the weighted minimal surface model[21]. Let $S$ denote the data set which can include data points, pieces of curves or surfaces. In our problem, $S$ means data points. Define $d(x) = \text{dist}(x, S)$ to be the distance function to $S$, where $\text{dist}(x, S) = \min_{y \in S} \| x - y \|_2$. Then this model begins to define the following surface energy:

$$E(\Gamma) = \left[ \int_{\Gamma} d^p(x) ds \right]^{\frac{1}{p}}, \quad 1 \leq p \leq \infty, \quad (2.1)$$

where $\Gamma$ is an arbitrary surface and $ds$ is the surface area. To find the surface of which energy is minimized, they calculate the gradient descent of the functional (2.1).

$$\frac{d\Gamma}{dt} = - \left[ \int_{\Gamma} d^p(x) ds \right]^{\frac{1}{p}-1} d^{p-1}(x) \left[ \nabla d(x) \cdot n + \frac{1}{p} d(x) \kappa \right] n, \quad (2.2)$$

where $n$ is the unit outward normal and $\kappa$ is the mean curvature. The flow from the above gradient (2.2) rules the movement of the surface. If the surface is initially far from data set, the surface evolves into the final surface with keeping the balance between the potential force
\[ \nabla d(x) \cdot n \] and the surface tension \( d(x) \kappa \). The scalar function \( d(x) \) allows the surface to be more flexible in regions close to the data set and to be more rigid in regions distant from the data set.

Here level set method is used to handle topological changes and to obtain an implicit surface. Let \( \Omega(t) \) be the (generally multiply connected) region enclosed by \( \Gamma(t) \). Let \( \phi(x, t) \) be the level set function associated with \( \Omega(t) \); i.e.,

\[
\begin{align*}
\phi(x, t) & < 0 \quad \text{in} \quad \Omega(t), \\
\phi(x, t) & = 0 \quad \text{on} \quad \Gamma(t), \\
\phi(x, t) & > 0 \quad \text{in} \quad \overline{\Omega}^c(t).
\end{align*}
\]

Then \( \Gamma(t) \) is identical to the zero level set of \( \phi(x, t) \). As [21], level set formulation of (2.1) is

\[
E(\Gamma) = \left[ \int_{\Gamma} d^p(x) ds \right]^{\frac{1}{p}} = E(\phi) = \left[ \int d^p(x) \delta(\phi(x)) |\nabla \phi(x)| d\mathbf{x} \right]^{\frac{1}{p}},
\]

where \( \delta(x) \) is the one-dimensional delta function and \( \delta(\phi(x)) |\nabla \phi(x)| d\mathbf{x} \) is the surface area element at the zero level set of \( \phi \). Also the gradient flow for \( \phi \) corresponding to (2.2) is

\[
\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[ \int d^p(x) \delta(\phi)|\nabla \phi| d\mathbf{x} \right]^{\frac{1}{p} - 1} d^{p-1}(x) \left[ \nabla d(x) \cdot \frac{\nabla \phi}{|\nabla \phi|} + \frac{1}{p} d(x) \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right],
\]

where \( \frac{\nabla \phi}{|\nabla \phi|} \) and \( \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \) are the level set representation of the unit normal and the mean curvature respectively.

In [22], Zhao et al. proposed the following convection model which is similar to the previous minimal surface model in methodology but uses the different physical model. The convection of a flexible surface \( \Gamma \) in a velocity field \( v(x) \) is described by

\[
\frac{d\Gamma(t)}{dt} = v(\Gamma(t)).
\]

If the velocity field is created by a potential field \( F \), then \( v = -\nabla F \). Because the distance function \( d(x) \) to the data set \( S \) means the potential field in this convection model, the convection equation can be represented by

\[
\frac{d\Gamma(t)}{dt} = -\nabla d(x).
\]

The level set formulation of the equation (2.6) is

\[
\frac{\partial \phi}{\partial t} = \nabla d(x) \cdot \nabla \phi.
\]

The evolution equation (2.2) is a nonlinear parabolic equation because it has the mean curvature of the surface. Since the convection equation (2.5) is a first order linear differential equation which has a time step \( \Delta t = O(h) \) where \( h \) is the grid size, it saves the time over parabolic \( \Delta t = O(h^2) \) time step restrictions.

The above two models handle the surface reconstruction problem as the time evolution equation. In [20], Ye et al. considered our problem as Poisson’s equation. They introduced the
Following energy functional of $\phi$:

Given point data $\{x_l\}_{l=1,...,N} \subset S$,

$$E(\phi) = \int G(\phi(x))dx + \sum_{l=1}^{J} \beta_l (P_l \phi)^2,$$

where the projection operator $P_l \phi = \int p_l(x) \phi(x) dx$ and $\int p_l(x) dx = 1$.

In case the data is uniformly distributed, they set $G(\phi(x)) = |\nabla \phi(x)|^2$. Then the first term play a role as the diffusion term which determines the smoothness of the surface and the second term functions as the fidelity term which fits the surface to point data. In the second term the weight parameter $\beta_l$ affects the accuracy of fitting the surface. If enough large $\beta_l$ is chosen, even crude initial boundary conditions result in very successful fitting.

Without getting the Euler-Lagrange equation from the energy functional (2.8) directly, they addressed the equation discretized from (2.8). In the case of two-dimensional, the equation is as follows.

$$\bar{E}(\phi) = \sum_{i} \sum_{j} \left( \frac{\phi_{i+1,j} - \phi_{i,j}}{h^2} \right)^2 + \left( \frac{\phi_{i,j+1} - \phi_{i,j}}{h} \right)^2 + J \sum_{l=1}^{J} \beta_l (\bar{P}_l \phi)^2,$$

where $\bar{P}_l \phi = \sum_{k,n} p^{l}_{k,n} \phi_{k,n}$ and $\sum_{k,n} p^{l}_{k,n} = 1$.

Differentiating the energy functional (2.9) with respect to $\phi(i, j)$ at the grid point $(i, j)$, they obtain the Euler-Lagrange equation as follows

$$\frac{1}{2} \frac{\delta \bar{E}}{\delta \phi_{i,j}} = -\frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{h^2} \sum_{l=1}^{J} \beta_l \bar{P}_l \phi = 0. \tag{2.10}$$

where $J$ is the total number of neighboring points of grid point $(i, j)$, operator $\bar{P}_l$ is a bilinear interpolation operator and $\bar{P}_l \phi$ represents the interpolated value of function $\phi$ at a point $x_l$.

$$P_l \phi = p^{l}_{i,j} \phi_{i,j} + p^{l}_{i,j+1} \phi_{i,j+1} + p^{l}_{i+1,j} \phi_{i+1,j} + p^{l}_{i+1,j+1} \phi_{i+1,j+1},$$

where $p^{l}_{i,j} = \frac{(h - r_1)(h - r_2)}{h^2}$, $p^{l}_{i,j+1} = \frac{(h - r_1)r_2}{h^2}$, $p^{l}_{i+1,j} = \frac{r_1(h - r_2)}{h^2}$, $p^{l}_{i+1,j+1} = \frac{r_1r_2}{h^2}$.

In three dimension, this model can be extended using trilinear interpolation operator instead of bilinear operator.

After they set initial boundary conditions in the narrow band with tagging algorithm such as the Breadth-First Search, they approximated the underlying surface fitting point data by solving the Poisson’s equation (2.10).

This model has the distinct characteristic compared to the previous models as it is a different type of PDE. It doesn’t need to be constrained by the CFL condition and its implementation can be improved because there are many efficient algorithms for solving Poisson’s equation.
However, their method still has the limitation that it needs so much memories because of being fulfilled only on the uniform grid. Their algorithm basically solves a large linear system of which size is the same as the number of the grid in a narrow band. If it is possible to make the width of band narrow enough, a small-sized linear system can be obtained. But in the above model, the decision of band width depends on the distribution of point data, more specifically, the maximum of distances between point data. It means that if point data are ranged sparsely or there is a big hole in the data, the width of band cannot help but become broad. Surely, these cases always exist because most point data from real objects are unorganized. Therefore, as far as it is implemented on the uniform grid, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically. So we needs adaptive grid because it result in the linear system of reasonable size even on high resolution grid. We employ data structures of octree to implement multi-resolution adaptively. Our scheme is based on Min’s[11].

3. NUMERICAL METHOD

We denote a (rectangular) cell in quadtree or octree by $C$ and the node of cell by $v$.

3.1. Tree Generation and Splitting Condition. We use a standard quadtree(resp. octree) data structure to represent the spatial discretization of the two(resp. three)-dimensional domain. Starting from the root of tree corresponding to the whole domain, each cell is split into four(resp. eight) children until the desired level of detail is achieved. We refer the reader to the books of Samet[16] for more details on quadtree/octree data structures.

In our case, the above process stops after the finest resolution for each cell containing at least a point is fulfilled. Details of the process is as follows.
First, for a point we start to split all children cell $C_s$ from a root cell according to the splitting condition

$$\text{dist}(x, \partial C) \leq \frac{1}{2} \min\{\text{width of } C, \text{height of } C\}. \quad (3.1)$$

After finishing the splitting for a point, we split cells satisfying the above condition starting from the root cell for another point. Of course during this process, we don’t split cells which is already split.

In our problem, we need the graded tree, which limits the difference of level between two adjacent cells to at most one, in order to secure the uniform grid near the interface. Otherwise, incorrect information on the nodes of big cells compared with neighboring cells can corrupt values at not only neighboring nodes but distant nodes.

**Lemma 1.** *The splitting condition (3.1) ensures the finest resolution in not only the cell containing the point, but also its neighboring cells. (e.g. 5 cells in quadtree or 7 cells in octree)*

**Proof.** At first, we consider quadtree case. If a point $x$ is inside a cell, it is obvious that the condition (3.1) is satisfied. Thus the finest resolution is guaranteed for the cell containing the point $x$. We denote the finest cell containing the point $x$ by $C_c$. For cells not having the point, we can consider four directional neighboring cells of $C_c$. We call those $C^N_c$ (north), $C^S_c$ (south), $C^E_c$ (east), $C^W_c$ (west), respectively. We have only to show that the levels of $C^N_c, C^S_c, C^E_c, C^W_c$ are equal to the level of $C_c$.

Consider the parent cell $C_p$ of $C_c$. Then $C_c$ is one of four quadrants in $C_p$. In case that $C_c$ is the first quadrant (i.e. north-east quadrant) in $C_p$, The north neighboring cell $C^N_p$ and the east neighboring cell $C^E_p$ of $C_p$ satisfy the condition (3.1). Therefore, there are $C^N_c$ and $C^E_c$ of which level equal the level of $C_c$. Moreover, since $C^S_c$ and $C^W_c$ are children of $C_p$, the levels of $C^S_c$ and $C^W_c$ are evidently the same as the level of $C_c$. For the other quadrants, we can get the same result by symmetry.

We also can extend the above discussion to octree easily. □

### 3.2. Distance Function

After the generation of octree, we calculate the distance from point data $x$ at each node $v_i$. This process progresses in a similar way as generating octree but with a condition different from the splitting condition.

To begin with, for a point $x_1$ chosen arbitrarily, we calculate the distances at nodes of the root cell. Then we check the following condition for each children of the root cell.

$$\text{dist}(x, \partial C) \leq \max\{\sqrt{(2l_1)^2 + l_2^2}, \sqrt{l_1^2 + (2l_2)^2}\} + \min\{l_1, l_2\}, \quad (3.2)$$

where $l_1$ is a half of the width of $C$ and $l_2$ is a half of the height of $C$. If a cell satisfies the condition (3.2) and distance values at nodes of the cell are less than the existing ones, the distances at nodes of the cell are updated. This job continues until it reaches the level of the finest cell containing the point $x_1$ along the hierarchy of the octree. Then with another point
x_2, we again calculate the distances at nodes of the cells satisfying the above condition (3.2). It stops after it is carried out for all points \{x\}.

In the process of updating the distances, it is better to search as many nodes as possible because there are some nodes of which distances are not minimum. Of course, if we choose enough large region, updating process will guarantee the minimum distances at all nodes. But distancing gets too time-consuming chore. Thus we need the least area guaranteeing a complete distance function in the entire domain. The condition (3.2) guarantees it.

The distancing on the octree is more efficient compared to that on the uniform grid. Denote the one side resolution of the uniform grid discretized in the entire domain by \(N\) and the number of point data by \(L\). On the uniform grid, the time required to obtain the distance function in the whole domain is \(O(N^3) + L\). This time came from the fast sweeping algorithm in [23] which is the most efficient algorithm on the uniform grid as far as we know. Meanwhile, we achieved the time of \(O(L \log(N))\) in the octree. We can usually regard \(L \approx N^2\) because \(L\) is the number of points on the interface in some domain. Then the time of distancing on the uniform grid is approximately \(O(LN)\) while \(O(L \log(N))\) in octree. We will demonstrate this results in section 4.

3.3. Initial Guess of Signed Distance Function. We need to initialize boundary conditions in order to solve the Poisson’s equation. In addition, boundary condition should contain the information about whether nodes are inside or outside the underlying surface of point data because we want to represent the surface by the zero level set of the level set function \(\phi\). To achieve it, we employ the fast tagging method in [22] in order to set boundary conditions. Here, we consider just two-dimensional case because it is easy to extend the process to three-dimension. In a way similar to [20], we start from a node \(v_1\) on the boundary of the entire domain to mark the outside region. The node \(v_1\) is unconditionally in the outside region. We check if the distances at neighboring nodes \(\{v_{N_i}\}_{N_i=1,...,8}\) of \(v_1\) satisfies the condition \(d(v_{N_i}) > \epsilon\) where \(\epsilon\) is a bandwidth-related constant. If there is a node satisfying this condition, we denote the node by the outside. After searching all neighboring nodes of \(v_1\), we pass another node \(v_2\) and do the same job again. When finishing this process for all nodes, we can get the outside region. Then, we set the distances for the outside region to \(\epsilon\) and the distances for \(\{x: d(x) < \epsilon\}\) to 0. Finally, the remaining nodes becomes the inside region and the distances for it is set to \(-\epsilon\).

The \(\epsilon\) in the above algorithm is required to satisfy \(\epsilon > \frac{1}{2}\epsilon_{\text{data}}\), where \(\epsilon_{\text{data}} = \max_i \{\min_{j \neq i} |x_i - x_j|\}\), to ensure that the region \(\{x: d(x) > \epsilon\}\) has two topological components. The above choice has some problems. In some data sets, i.e. a set having a large hole, the \(\epsilon_{\text{data}}\) may be large. Then this initial guess can be very rough because of large bandwidth. In this case, our method cannot guarantee appropriate result.

3.4. Basic Finite Difference Methods on Octree. In the case of non uniform Cartesian grids, the main difficulty comes from deriving discretizations at T-junction nodes, i.e. nodes for which there is a missing neighboring node in one of the Cartesian directions. For example, Figure 2 depicts a T-junction node \(v_0\), with three neighboring nodes \(v_1, v_2\) and \(v_3\) aligned in the Cartesian directions and one ghost neighboring node \(v_4\) replacing the missing grid node in
the positive Cartesian direction. The value of a node-sampled function \( \phi : \{ v_i \} \to \mathbb{R} \) at the ghost node \( v_4 \) could for example be defined by linear interpolation:

\[
\phi^G_4 = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6}.
\] (3.3)

However, instead of using this second order accurate interpolation, one can use the following third order accurate interpolation: First, note that a simple Taylor expansion demonstrates that the interpolation error in equation (3.3) is given by:

\[
\phi^G_4 = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} = \phi(v_4) + \frac{s_5 s_6}{2} \phi_{yy}(v_0) + O(\Delta x_{\text{smallest}})^3,
\] (3.4)

where \( \Delta x_{\text{smallest}} \) is the size of the smallest grid cell with vertex \( v_0 \). The term \( \phi_{yy}(v_0) \) can be approximated using the standard first order accurate discretization \( \frac{2}{s_2 + s_3} \left( \frac{\phi_2 - \phi_0}{s_2} + \frac{\phi_3 - \phi_0}{s_3} \right) \) and canceled out in equation (3.4) to give:

\[
\phi^G_4 = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} - \frac{s_5 s_6}{s_2 + s_3} \left( \frac{\phi_2 - \phi_0}{s_2} + \frac{\phi_3 - \phi_0}{s_3} \right).
\] (3.5)

We also point out that this interpolation only uses the node values of the cells adjacent to \( v_0 \), which is particularly beneficial since access to cells not immediately adjacent to the current cell is more difficult and could add on CPU time and/or memory requirement.
In three spatial dimensions, similar interpolation procedures can be used to define the value of $\phi$ at ghost nodes. Referring to Figure 3, a T-junction node $v_0$ has four regular neighboring nodes and two ghost nodes. The values of a node-sampled function $\phi : \{v_i\} \to \mathbb{R}$ at the ghost nodes $v_4$ and $v_5$ can be defined by second order linear and bilinear interpolations as:

$$\phi^G_{v_4} = \frac{s_7 \phi_8 + s_8 \phi_7}{s_7 + s_8},$$

$$\phi^G_{v_5} = \frac{s_{11} s_{12} \phi_{11} + s_{11} s_9 \phi_{12} + s_{10} s_{12} \phi_9 + s_{10} s_9 \phi_{10}}{(s_{10} + s_{11}) (s_9 + s_{12})}. \tag{3.6}$$

As in the case of quadtrees, third order accurate interpolations can be derived by canceling out the second order derivatives in the error term to arrive at:

$$\phi^G_{v_4} = \frac{s_7 \phi_8 + s_8 \phi_7}{s_7 + s_8} - \frac{s_7 s_8}{s_3 + s_6} \left( \frac{\phi_3 - \phi_0}{s_3} + \frac{\phi_6 - \phi_0}{s_6} \right),$$

$$\phi^G_{v_5} = \frac{s_{11} s_{12} \phi_{11} + s_{11} s_9 \phi_{12} + s_{10} s_{12} \phi_9 + s_{10} s_9 \phi_{10}}{(s_{10} + s_{11}) (s_9 + s_{12})} \left( \frac{\phi_1 - \phi_0}{s_1} + \frac{\phi^G_{v_4} - \phi_0}{s_4} \right). \tag{3.7}$$
We emphasize that Figure 3 represents the general configuration of neighboring nodes in the case of an octree as described in Min [12].

The third order interpolations defined above allow us to treat T-junction nodes in a same fashion as a regular node, up to third order accuracy. Here, we refer to a regular node as a node for which all the neighboring nodes in the Cartesian directions exist. Therefore, we can then define finite differences for $\phi_x$, $\phi_y$, $\phi_z$, $\phi_{xx}$, $\phi_{yy}$ and $\phi_{zz}$ at every nodes using standard finite difference formulas in a dimension by dimension framework. For example, referring to Figure 4, we use the standard discretization for $\phi_x$ and $\phi_{xx}$, namely the central difference formulas:

\[
D^0_x \phi_0 = \frac{\phi_2 - \phi_0}{s_2} \cdot \frac{s_1}{s_1 + s_2} + \frac{\phi_0 - \phi_1}{s_1} \cdot \frac{s_2}{s_1 + s_2},
\]
\[
D^0_{xx} \phi_0 = \frac{\phi_2 - \phi_0}{s_2} \cdot \frac{2}{s_1 + s_2} - \frac{\phi_0 - \phi_1}{s_1} \cdot \frac{2}{s_1 + s_2},
\]

(3.8)

the forward and backward first order accurate approximations of the first order derivatives:

\[
D^+ x \phi_0 = \frac{\phi_2 - \phi_0}{s_2},
\]
\[
D^- x \phi_0 = \frac{\phi_0 - \phi_1}{s_1},
\]

(3.9)

and the second order accurate approximations of the first order derivatives:

\[
D^+ x \phi_0 = \frac{\phi_2 - \phi_0}{s_2} - \frac{s_2}{2} \minmod \left( D^0_{xx} \phi_0, D^0_{xx} \phi_2 \right),
\]
\[
D^- x \phi_0 = \frac{\phi_0 - \phi_1}{s_1} + \frac{s_1}{2} \minmod \left( D^0_{xx} \phi_0, D^0_{xx} \phi_1 \right),
\]

(3.10)

where we use the minmod slope limiter [9, 18] because it produces more stable results in region where $\phi$ might present kinks. Similarly, approximations for first and second order derivatives are obtained in the $y$ and $z$ directions.

**Figure 4.** One dimensional adaptive grid

3.5. **Numerical Discretization of Model (2.8) on Octree.** We discretize equation (2.8) based on the scheme in the previous section. While the coefficients of the discretized fitting term on octree differs little from that on the regular nodes, the coefficients of the discretized Laplacian term are distinct compared with those on the uniform grid because of T-junction nodes. We need the following process in order to discretize the Laplacian term on octree. Let us consider the conditions of Figure 3. In Figure 3, only $v_5$ was considered as the ghost node which needs bilinear interpolation. We extend this situation to all neighboring nodes of $v_0$, that is,
Then in a similar way as the case of $v_5$, we can get the values of a node-sampled function at $v_1, v_2, v_3, v_4, v_6$. As follows.

\[
\phi_1^G = \frac{s_{18}s_{20}\phi_{17} + s_{18}s_{19}\phi_{18} + s_{17}s_{20}\phi_{19} + s_{17}s_{19}\phi_{20}}{(s_{17} + s_{18})(s_{19} + s_{20})} (3.11)
\]

\[
\phi_2^G = \frac{s_{13}s_{16}\phi_{13} + s_{14}s_{16}\phi_{14} + s_{13}s_{15}\phi_{15} + s_{14}s_{15}\phi_{16}}{(s_{13} + s_{14})(s_{15} + s_{16})}
\]

\[
\phi_3^G = \frac{s_{28}s_{29}\phi_{27} + s_{27}s_{29}\phi_{28} + s_{28}s_{30}\phi_{29} + s_{27}s_{30}\phi_{30}}{(s_{27} + s_{28})(s_{29} + s_{30})} (3.13)
\]

\[
\phi_4^G = \frac{s_{8}s_{22}\phi_{7} + s_{8}s_{21}\phi_{8} + s_{7}s_{22}\phi_{21} + s_{8}s_{21}\phi_{22}}{(s_{7} + s_{8})(s_{21} + s_{22})} (3.14)
\]

\[
\phi_5^G = \frac{s_{23}s_{26}\phi_{24} + s_{23}s_{25}\phi_{25} + s_{24}s_{26}\phi_{26} + s_{24}s_{25}\phi_{27}}{(s_{23} + s_{24})(s_{25} + s_{26})} (3.15)
\]

Then, we obtain the following equations through simple manipulation.

\[
\left(\frac{\phi_2^G - \phi_0}{s_2} + \frac{\phi_5^G - \phi_0}{s_5}\right) \left(\frac{2}{s_2 + s_5}\right) = \phi_{xx}(v_0) + \left(\frac{s_{9}s_{12}}{(s_2 + s_5)s_5} + \frac{s_{13}s_{14}}{(s_2 + s_5)s_2}\right) \phi_{yy}(v_0) + \left(\frac{s_{10}s_{11}}{(s_2 + s_5)s_5} + \frac{s_{15}s_{16}}{(s_2 + s_5)s_2}\right) \phi_{zz}(v_0) (3.16)
\]

\[
\left(\frac{\phi_4^G - \phi_0}{s_4} + \frac{\phi_1^G - \phi_0}{s_1}\right) \left(\frac{2}{s_1 + s_4}\right) = \phi_{xx}(v_0) + \left(\frac{s_{10}s_{20}}{(s_1 + s_4)s_4} + \frac{s_{21}s_{22}}{(s_1 + s_4)s_1}\right) \phi_{yy}(v_0) + \left(\frac{s_{17}s_{18}}{(s_1 + s_4)s_4} + \frac{s_{7}s_{8}}{(s_1 + s_4)s_4}\right) \phi_{zz}(v_0) (3.17)
\]

\[
\left(\frac{\phi_6^G - \phi_0}{s_6} + \frac{\phi_3^G - \phi_0}{s_3}\right) \left(\frac{2}{s_3 + s_6}\right) = \phi_{xx}(v_0) + \left(\frac{s_{27}s_{28}}{(s_3 + s_6)s_3} + \frac{s_{25}s_{26}}{(s_3 + s_6)s_6}\right) \phi_{yy}(v_0) + \left(\frac{s_{20}s_{20}}{(s_3 + s_6)s_3} + \frac{s_{23}s_{24}}{(s_3 + s_6)s_6}\right) \phi_{zz}(v_0) (3.18)
\]

By multiplying equations (3.16),(3.17),(3.18) by some weights $w_1, w_2, w_3$, respectively, we can get a third order accurate approximation of $\Delta \phi$ in three dimensions.
FIGURE 5. The left picture represents the initial narrow band with the boundary condition. The right picture is the magnified concave part of five-leafed clover. The red thick lines denote the outside boundaries and the blue thick line denote the inside boundaries. The black dots represent point data. The red squares show the grid on the octree. The fine grids cluster near the point data while the coarse grids is far from points.

TABLE 1. The processing time and the required memory for reconstructing the two-dimensional five-leafed clover

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Uniform Grid</th>
<th>Octree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(sec)</td>
<td>Memory(MB)</td>
</tr>
<tr>
<td>512²</td>
<td>0.15</td>
<td>9.7</td>
</tr>
<tr>
<td>1024²</td>
<td>0.52</td>
<td>66.4</td>
</tr>
<tr>
<td>2048²</td>
<td>2.05</td>
<td>234.4</td>
</tr>
</tbody>
</table>

4. EXPERIMENTAL RESULTS

We tested our method and Ye’s[20] with some two-dimensional and three-dimensional examples. These tests were executed on a PC with 3.16GHz Intel Core 2 Duo CPU and 4GB RAM. We adopted the Biconjugate Gradient Stabilized (BICGSTAB) algorithm to the process of minimizing the functional (2.8). The stopping criterion used is that 2-norm of the residual is less than $10^{-8}$.

4.1. Five-leafed clover. First, we generated an artificial point data of which shape is five-leafed clover on two-dimension. The data is made up of 5000 uniformly distributed points. We reconstructed the contour from the data by varying the finest resolution from $512^2$ to $2048^2$. 
Figure 6. The left picture shows the reconstructed five-leafed clover from point cloud. The right picture is the magnified concave part of five-leafed clover. This indicates that the curve is well-fitted to even the structure with high curvature.

Of course, our method can be implemented with quadtree in two-dimension. We compares the processing time and the required memory on quadtree with on the uniform grid in Table 1. Like Table 1 shows, the processing time on quadtree gets shorter than on the uniform grid as the resolution increases. Furthermore, the difference of the memory requirement is much greater than the difference of the time.

As this example was comparatively smooth in shape, the qualities of the curves reconstructed on several different resolutions made no difference. So we show one of those in Figure 6. Figure 5 represents the initial condition for solving the minimization of the functional (2.8), that is, the initial narrow band.

4.2. Bunny, Dragon, Happy Buddha. In three dimensional, we reconstructed the Stanford bunny and the Stanford dragon and happy Buddha. Figure 8 and Figure 9 show results on grids of which maximum resolution is $256^3$ for the Stanford bunny and the Stanford dragon, respectively. Actually, although we obtained the result of $1024^3$ resolution with octree on our PC, the difference of the quality was not noticeable in the case of bunny and dragon. So we demonstrate just $256^3$ case among those. But we could confirm the improvement of detail according to an increase in the resolution from the reconstruction for happy Buddha(Figure 10).

In Table 2, the table on the top shows the processing time and the required memory on the uniform grid and the table on the bottom demonstrates the time and the memory on octree. In case the resolution is higher than $256^3$, we could not execute the algorithm on the uniform grid in our test environment because of memory depletion. Likewise two-dimensional case, octree
structure saved much processing time compared with uniform grid as the resolution get higher. Only in low resolution on octree structure, e.g. $128^3$, the time for calculating the unsigned distances is longer than on the uniform grid, in particular, for dragon and Buddha. As we mentioned it in the last paragraph of section 3.2, it is because the distancing on octree depends on the number of point data while the distancing on the uniform grid depends on the number of grids only. But this weak point is overcome as the resolution increases. Moreover, this process which comprises a large portion of the whole process can be improved. In this test, we didn’t adapt the k-d tree to the storage of the point data. But we expect that this will help to speed up that process.

Finally, one thing that we need to remark is that in the experiment for Stanford bunny, a large amount of memory was needed for the number of point. The reason is the sparse distribution of point data. As we referred in the end of section 2, the sparsity of point data bring about a broad initial band, that is, many grid points which need to be processed. This experimental result for Stanford bunny reflects the fact.
FIGURE 8. Three dimensional surface reconstruction for Stanford bunny on the grid of which maximum resolution is $256^3$. The number of point is 35947.

5. CONCLUSION AND FUTURE WORK

We introduced a new surface reconstruction algorithm using level set method and octree. Although our mathematical model is based on Ye’s [20], ours resolves the drawback that Ye’s has. Furthermore, our algorithm is more efficient than the previous and can generate the high resolution.

Future research will address a new mathematical model which employs not Laplacian term but the term related to curvature for smoothing.

ACKNOWLEDGMENTS

Changsoo Park was supported by the Industrial Strategic Technology Program of the Ministry of Knowledge Economy(project ID:10035474). Chohong Min was supported by Priority Research Centers Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0028298). Myungjoo Kang was supported by the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2011-0002404) and Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2011(211A5020021098501007).

REFERENCES

FIGURE 9. Three dimensional surface reconstruction for dragon on the grid of which maximum resolution is $256^3$. The number of point is 460986.

Figure 10. Three dimensional surface reconstruction for happy Buddha. The left one is acquired on the grid of which the maximum resolution is $256^3$ and the right one does on the grid of $512^3$. The number of point is 543652.

TABLE 2. The processing time and the required memory for reconstructing the three-dimensional Stanford bunny and dragon. (a) the time for generating octree. (b) the time for calculating the unsigned distances. (c) the time for initializing the boundary condition. (d) the time for minimizing the functional (2.8).

<table>
<thead>
<tr>
<th>Data</th>
<th>Resolution</th>
<th>Uniform Grid</th>
<th></th>
<th>Time(sec)</th>
<th>Memory(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(a) (b) (c) (d) Total</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bunny</td>
<td>128³</td>
<td>1.03</td>
<td>1.72</td>
<td>0.57</td>
<td>3.32</td>
</tr>
<tr>
<td></td>
<td>256³</td>
<td>16.15</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>512³</td>
<td>207</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dragon</td>
<td>128³</td>
<td>1.02</td>
<td>1.83</td>
<td>2.67</td>
<td>5.52</td>
</tr>
<tr>
<td></td>
<td>256³</td>
<td>16.23</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>512³</td>
<td>205</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>128³</td>
<td>1.02</td>
<td>1.86</td>
<td>4.05</td>
<td>6.93</td>
</tr>
<tr>
<td></td>
<td>256³</td>
<td>15.37</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>512³</td>
<td>204</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>Resolution</th>
<th>Octree</th>
<th></th>
<th>Time(sec)</th>
<th>Memory(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(a) (b) (c) (d) Total</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bunny</td>
<td>128³</td>
<td>1.69</td>
<td>3.54</td>
<td>0.12</td>
<td>2.14</td>
</tr>
<tr>
<td></td>
<td>256³</td>
<td>2.76</td>
<td>4.42</td>
<td>0.14</td>
<td>17.61</td>
</tr>
<tr>
<td></td>
<td>512³</td>
<td>5.42</td>
<td>5.06</td>
<td>0.19</td>
<td>144.28</td>
</tr>
<tr>
<td>Dragon</td>
<td>128³</td>
<td>19.24</td>
<td>44.89</td>
<td>0.18</td>
<td>1.16</td>
</tr>
<tr>
<td></td>
<td>256³</td>
<td>24.31</td>
<td>56.54</td>
<td>0.65</td>
<td>4.89</td>
</tr>
<tr>
<td></td>
<td>512³</td>
<td>31.25</td>
<td>66.44</td>
<td>0.78</td>
<td>45.83</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>128³</td>
<td>21.63</td>
<td>52.22</td>
<td>0.16</td>
<td>1.44</td>
</tr>
<tr>
<td></td>
<td>256³</td>
<td>26.67</td>
<td>65.25</td>
<td>0.54</td>
<td>5.55</td>
</tr>
<tr>
<td></td>
<td>512³</td>
<td>34.35</td>
<td>75.89</td>
<td>0.79</td>
<td>42.29</td>
</tr>
</tbody>
</table>


THE DIMENSION REDUCTION ALGORITHM FOR THE POSITIVE REALIZATION OF DISCRETE PHASE-TYPE DISTRIBUTIONS

KYUNGSUP KIM

1DEPARTMENT OF COMPUTER ENGINEERING, CHUNGNA National University, Yuseong-gu, Daejeon, Korea
E-mail address: sclkim@cnu.ac.kr

ABSTRACT. This paper provides an efficient dimension reduction algorithm of the positive realization of discrete phase type(DPH) distributions. The relationship between the representation of DPH distributions and the positive realization of the positive system is explained. The dimension of the positive realization of a discrete phase-type realization may be larger than its McMillan degree of probability generating functions. The positive realization with sufficient large dimension bound can be obtained easily but generally, the minimal positive realization problem is not solved yet. We propose an efficient dimension reduction algorithm to make the positive realization with tighter upper bound from a given probability generating functions in terms of convex cone problem and linear programming.

1. INTRODUCTION

This paper handles the dimension reduction algorithm of the positive realization with discrete phase-type distributions. Continuous and Discrete phase type(DPH) distributions have been introduced in [1]. Many research activities and applications have been devoted to the field of continuous PH distributions. Recent, a new attention has been devoted to discrete model since they are more closely related to physical observations, the numerical solution of non-Markovian processes, stochastic modeling and applications [2]. The representation of DPH distributions is closely related to the positive realization of the positive systems, which have been researched wide in control and system theory communities. The representation problem finding a Markov chain associated with phase-type distribution has a lot of links with positive realization problem in the control theory [3]. The relationship between the probability generating functions of the probability mass function of a DPH-distribution and a corresponding representation is very similar to the relationship between a transfer function and a corresponding state space realization. The DPH representation are a special form of state space representations with particular constraints on the Markov generators.

Received by the editors December 6 2011; Revised January 2 2012; Accepted in revised form January 25 2012.
2010 Mathematics Subject Classification. 60J05.
Key words and phrases. Discrete phase-type distribution, Discrete Matrix Exponential, Positive realization.
This study was financially supported by research fund of Chungnam National University in 2009.
The parameter estimation algorithm for a DPH distribution has been introduced and formalized in [3]. It is well known that general PH distributions are considerably over-parameterized and have high order representation in fitting or approximation problems[4][5]. The problem of over-parameterizations and high order are due to the constraints of the PH-distribution. The Matrix Exponential (ME) approach can solve the problems and the fitting methods with ME get better results[4][5]. The realization of ME distribution can be converted into the PH realization with higher order bound (so called, positive realization) [6]. From this motivation, we will handle the positive realization of DPH distribution from discrete ME distribution under the assumption that the discrete ME distribution can be obtained.

When a non-negativity constraint is imposed on the realizations, the existence and minimality problems becomes more involved. Problems associated with existence and minimality of positive realization have been widely studied in the last decades and a complete answer is not available. The positive constraint may enforce realizations of larger dimension then McMillan degree[7][8]. Therefore, the minimality problem has been dealt with in a number of papers[9][10]. The minimality problem is solved only for particular classes of transfer functions, and a general solution seems to be out of reach in the current state of art [11]. Therefore, in order to induce a minimal positive realization, we need an efficient dimension algorithm to make a positive realization with tighter dimension bound.

We attempt to propose an efficient general algorithm to reduce the dimension of the positive realization of probability generating functions. We note that a polyhedral cone existence gives a necessary and sufficient condition of the positive realization. Using the dual cone, we formulate a proper polyhedral cone. Furthermore, the novel dimension reduction algorithm of the positive realization will be proposed by using linear programming and convex cone. A new numerical example is given to verify that the proposed algorithm works well.

An outline of the paper is as follows: In Section 2, we introduce the relationship between the DPH distribution and the positive realization of positive system. In section 3, some definitions and preliminary results from convex analysis and linear programming are introduced. Section 4 contains the main results such as the positive realization with sufficiently large upper-bound and an dimension reduction algorithm of the positive realization. Finally, the conclusion section follows.

2. PROBLEMS IN DISCRETE PHASE DISTRIBUTION AND POSITIVE REALIZATION

2.1. DISCRETE PHASE DISTRIBUTION. A discrete phase distribution is the distribution of the time until in a discrete-state discrete-time Markov chain (DTMC) with $n$ transients states and one absorbing state [1][2]. If the transient states are numbered 1, 2, ⋯ , $n$ and the absorbing state is numbered ($n + 1$), the one-step transition probability matrix of the corresponding DTMC can be partitioned as

$$
\hat{B} = \begin{bmatrix}
B & b \\
0 & 1
\end{bmatrix}
$$

where $B = [b_{ij}]$ is the $n \times n$ matrix grouping the transition probabilities among the transient states, $b = [b_i]$ is the $n$-dimensional column vector grouping the probabilities from any state
to the absorbing one. Since \( \hat{B} \) is the transition probability matrix of a DTMC, the following relation holds: \( \sum_{j=1}^{n} b_{ij} = 1 - b_i \).

The initial probability vector of the DTMC is an \( n + 1 \)-dimensional vector \( \hat{\alpha} = [\alpha \ \alpha_{n+1}] \) with \( \sum_{j=1}^{n} \alpha_j = 1 - \alpha_{n+1} \). We only consider the class of DPH distributions for which \( \alpha_{n+1} = 0. \)

Let \( \tau \) be the time till absorption in state \( n + 1 \) in the DTMC. We say that \( \tau \) is a random variable of order \( n \) and representation \( (\alpha, B, b) \) [1]. Let \( f(k) \), \( F(k) \) and \( \mathcal{G}(s) \) be the probability mass, cumulative probability and probability generating function of \( \tau \), respectively, as follows:

\[
\begin{align*}
  \ f(k) &= \Pr\{\tau = k\} = \alpha B^{k-1} b \geq 0, \quad \text{for } k > 0 \\
  \ F(k) &= \Pr\{\tau \leq k\} = \alpha \sum_{i=0}^{k-1} B^i b = \alpha (I - B^k)(I - B)^{-1} b \leq 1 \\
  \ \mathcal{G}(z) &= E(z^\tau) = \alpha (z I - B)^{-1} b = \frac{U(z)}{V(z)}
\end{align*}
\]

where \( e \) is an \( n \)-dimensional column vector with all the entries equal to 1, \( I \) is the \((n \times n)\) identity matrix and the realization \( (\alpha, B, b) \) has constraints such that \( \alpha \geq 0, 0 \leq b_{ij} \leq 1 \) and \( b_i \geq 0 \) for all \( i, j \) (i.e., \( (\alpha, B, b) \) is denoted by a positive realization). In DPH realization, an additional condition \( (I - B)e = b \) should be needed where \( e \) is an \( n \)-dimensional matrix column vector with all the entries equal to 1. \( U(z) \) and \( V(z) \) are polynomials with respect to \( s \). When DPHs are used to approximate general distributions to transform a non-Markovian process into a DTMC, a very crucial feature is to keep the order of the positive DPH realization as low as possible [2].

**Theorem 2.1.** Assume that the realization \( (\alpha, B, b) \) of a probability generating function is a positive realization satisfying Eq. (2.1)(2.2) and (2.3). Then there is a nonsingular matrix \( M \) such that another positive realization \( (\hat{\alpha}, \hat{B}, \hat{b}) \) transformed by \( \hat{\alpha} = \alpha M, \hat{B} = M^{-1}BM \) and \( \hat{b} = M^{-1}b \) satisfies that \( \hat{b} = (I - \hat{B})e \).

**Proof.** Since \( A \) is positive valued matrix and absolute values of its eigenvalues are less than 1, the all entries \( (I - B)^{-1} = \sum_{k=0}^{\infty} B^k \) are positive-valued. The entries of \( x = (I - B)^{-1} b \) are positive. A similarity transform matrix \( M \) is defined by \( M = \text{diag}(x) \). We can verify easily \( (I - B)e = b \).

From Theorem 2.1, if a positive realization of a probability generating function is constructed, we can induce a DPH realization defined in [2]. We introduce the discrete Matrix Exponential distribution (DME) as a general version of DPH distributions.

**Definition 2.1 (Discrete Matrix Exponential Distribution).** The representation \( (\alpha, B, b) \) is called as discrete Matrix Exponential Distribution (DME) if it satisfies the following conditions

1. The induced probability mass function are nonnegative (i.e., \( f(k) = \alpha B^k b \geq 0 \) for all \( k > 0 \))
2. The induced cumulative probability distribution is less than or equal to 1 for \( k < \infty \) (i.e, \( F(k) = \Pr\{\tau \leq k\} \leq 1 \).
(3) The total sum of \( \{f(k)\} \) is 1 (i.e., \( \alpha(I - B)^{-1}b = 1 \)). The triple \((\alpha, B, b)\) is called a representation of the DME distribution.

In the definition of DME, there is no nonnegative constraint in the generator triple \((\alpha, B, b)\). Then we can obtain a realization with minimal dimension. Given a probability generating function corresponding to DME distribution, we can easily find a canonical DME realization \((\alpha, B, b)\) such as

\[
\alpha = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \\
B = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
-b_1 & -b_2 & -b_3 & \cdots & -b_n \end{bmatrix} \\
b = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}^T
\]

where \( U(z) = a_n z^{n-1} + a_{n-1} z^{n-2} + \cdots + a_1, V(z) = z^p + b_n z^{n-1} + \cdots + b_1 \) and \( b^T \) is defined by the transpose of \( b \). The nonnegative conditions of the DME representation \((\alpha, B, b)\) are not necessary. Therefore, we can expect that the fitting or identification problems with DME distributions are more efficient and easier than those of DPH distribution. However, we assume that a general DME distribution can be obtained. The main problem in this paper is to find the minimal positive realization of DPH distribution as possible.

2.2. Positive systems: Definitions and problems. The representation of DPH distributions is closely related to the positive realization of the positive systems. We set \( z = s^{-1} \) and \( H(z) = \bar{F}(s) \). Assume that there exists the rational transfer function of a given positive system with McMillon degree \( n \) such as

\[
H(z) = a_n z^{n-1} + a_{n-1} z^{n-2} + \cdots + a_1, V(z) = z^p + b_n z^{n-1} + \cdots + b_1, \quad b^T
\]

Then the impulse response sequence \( h_k \) of \( H(z) \) are nonnegative (i.e., \( h_k \geq 0 \)). A positive linear system is denoted by a linear dynamical system in which the input \( u \), state \( x \) and output \( y \) are nonnegative real-valued. We deal with time-invariant finite-dimensional positive in discrete time. The linear system will be represented by

\[
x(t + 1) = Ax(t) + bu(t) \\
y(t) = cx(t)
\]

where \( A \in \mathbb{R}^{N \times N}, b \in \mathbb{R}^N, \) and \( c^T \in \mathbb{R}^N \). The rational transfer function has a state space realization \( H(z) = c^T(zI - A)^{-1}b \). The state space realization \((A, b, c)\) is denoted by a positive realization if there exists a triple \( A \in \mathbb{R}_+^{N \times N}, b \in \mathbb{R}_+^N \), and \( c^T \in \mathbb{R}_+^N \) where \( \mathbb{R}_+ \) denote nonnegative real number. It is known that the linear system with positive realization \((A, b, c)\) is a positive system. The positive realization problem is to find a triple \((A, b, c)\) with nonnegative
entries for a given positive system. The positive realization problem has some important issues. Necessary and sufficient conditions for the existence of a positive linear system are completely presented in [12][13]. It is known that the constraint of positivity may force the dimension $N$ to be strictly larger than McMillon degree $n$ [7][10]. The positive minimality problem is to find the lowest possible value of $N$. It seems to be very difficult to give tight lower bounds and upper bounds of the dimension of positive realization [10]. The minimality problems were handled in [9][10][11]. We attempt to propose an efficient general algorithm to find the minimal positive realization for a special class of transfer functions by using the convex polyhedral cone approach and linear programming.

3. Polyhedral cone approach for DPH distribution

3.1. Convex Polyhedral cone. The concepts and results related to convex polyhedral cones are reviewed in this section (refer to references [14] and [15]). A set $K$ is a cone, if $\alpha K \subseteq K$ for all $\alpha > 0$. If $K$ contains an open ball of $\mathbb{R}^n$, then $K$ is solid. A cone that is closed, convex, and solid and pointed is denoted by a proper cone. If $K \cap \{-K\} = \{0\}$, then $K$ is a pointed cone. $cone(v_1, \ldots, v_n)$ denotes the polyhedral closed convex cone consisting of all nonnegative linear combinations of vectors $v_1, \ldots, v_n$. A set $K \subseteq \mathbb{R}^k_+$ is denoted by a polyhedral cone if an $n \in \mathbb{Z}^+$ and a $V = [v_1 \cdots v_n] \in \mathbb{R}^{n \times k}$ exist in order that

$$K = \{V x | x \in \mathbb{R}^n_+\} = cone(V) \quad (3.1)$$

The description in equation (3.1) is a vertex description [14]. A polyhedral cone can also be described by using the intersection of a finite number of half-spaces and hyperplanes such as $K = \{y \in \mathbb{R}^n | Ay \succeq 0, Cy = 0\}$, where $A$ and $C$ denote matrices with finite dimensions. This is denoted by the half-space description.

Given a closed convex cone $K$ in a subspace of $\mathbb{R}^n$, $V$ is called a generator of $K$. For any set $K$, the dual cone is defined by

$$K^* = \{y \in \mathbb{R}^n | x^T y \geq 0, \text{ for all } x \in K\}. \quad (3.2)$$

A finite set of vectors, $\{v_1, \ldots, v_n\} \subset \mathbb{R}^k$, with a columnar matrix, $V = [v_1 \cdots v_n] \in \mathbb{R}^{n \times k}_+$, is said to be positively dependent if there exists a $v_i$ such that it can be written as a nonnegative linear combination of $\{v_j, j \neq i\}$; otherwise, it is positively independent. Consider the extreme direction for a convex cone set (i.e. refer to reference [14]). For a given convex cone, $K$, a nonzero direction $v \in K$ is called an extreme point if $v$ is positively independent of the elements of $K - \{cv | c \geq 0\}$. Recursively, a set of extreme points can be found from $\{v_1, \ldots, v_n\} \subset \mathbb{R}^k$, which is called a frame set. So called, a frame set is an extreme point set spanning the convex cone set $K$. The frame columnar matrix of induced from $V$ is denoted by $V_1 = [v_1 \ v_2 \cdots \ v_{n_1}]$ with $n_1 \leq n$. There are a permutation matrix $P$ and a nonnegative matrix $Q$ such that the columnar matrix $V$ is divided by

$$PV = [V_1 \ V_2], \quad V_2 = V_1Q. \quad (3.3)$$

Using linear programming, An algorithm of a conic independency in Algorithm 1 is derived, which removes all positively dependent points from the given generator. A generator set for a
convex cone is arranged in a matrix columnar, \( V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{m \times n} \). Select an extreme point set \( V_1 \) from \( V \).

**Algorithm 1** Algorithm of conic independency (CI)

**Require:** A generator set for a convex cone is arranged in a matrix columnar, \( V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{m \times n} \).

**Ensure:** Select an extreme point set \( V_1 \) from \( V \).

1. **procedure** CI(\( V, V_1, P, Q \)) ▷ Input: \( V \), Output: \( V_1, P, Q \)
2. Set \( V_1 = [\ ] \) and \( \bar{V} = V \)
3. for \( 1 \leq k \leq n \) do
   4. • The positive independency test may be expressed as a set of the linear feasibility problem:
   
   \[
   \begin{array}{l}
   \text{find} \quad \alpha \in \mathbb{R}^n \\
   \text{subject to} \quad V\alpha = v_k \\
   \quad \alpha \succeq 0 \\
   \quad \alpha_k = 0.
   \end{array}
   \]
   5. if a feasible solution \( \alpha \) exists then
   6. • a direction \( v_k \) must be removed from the generator set \( \bar{V} \) since \( v_k \) is positive dependent.
   7. else
   8. \( V_1 = [V_1 \ v_k] \)
   9. end if
10. end for
11. **return** \( V_1 \).

Through the fundamental theorem of linear programming presented in reference [16], the basic feasible solutions in solving linear programs are established.

**Theorem 3.1** ([16]). Let \( A \) be an \( m \times n \) matrix of rank \( m \) and let \( b \) be an \( m \)-vector. Let \( K \) be the convex polyhedral consisting of all \( n \)-vectors \( x \) satisfying

\[
Ax = b \\
x \geq 0
\]

(3.4)

A vector \( x \) is an extreme point of \( K \) if and only if \( x \) is a basic feasible solution for equation (3.4).

3.2. **Dual cone generator.** It is noted that the concepts of the polyhedral cone and dual polyhedral cone are useful in analyzing the existence conditions of the positive realization. There are two types of descriptions for polyhedral cones: the vertex-description and the half-space-description. It is relatively easy for the half-space-description (i.e., use generator matrix) to be considered in the computation. The half-space-description of the dual cone is transformed from a generator matrix of a given polyhedral cone [14]. Consider a given matrix \( A \) and closed
THE DIMENSION REDUCTION ALGORITHM FOR THE POSITIVE REALIZATION

convex cone $\mathcal{K}$. The membership relationship is \{y | Ay \in \mathcal{K}^*\} = \{A^T x | x \in \mathcal{K}\}^*$. When $\mathcal{K}$ is $\mathbb{R}^n_+$, then \{y | Ay \geq 0\} = \{A^T x | x \geq 0\}^*$ is obtained. A theorem can be arranged for a generator of a dual cone (i.e., to convert the half-space-description) from a given generator matrix of a convex cone. For more detailed proof, refer to reference [14].

**Theorem 3.2 ([14]).** Assume that there exists a half space-description such that $\mathcal{K} = \{x \in \mathbb{R}^n | X x \geq 0\}$ for a pointed polyhedral cone $\mathcal{K}$ where its extreme directions $X_i$ arranged columnar is given by $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix} \in \mathbb{R}^{n \times N} \text{ with } \text{rank}(X) = N$ and $n \geq N$. Define a matrix $X^\perp = \text{basis}(N(X^T))$. A columnar basis $X^\perp$ is the orthogonal complement of a range space $\mathcal{R}(X)$. Then the vertex-description generator $X^*$ of $\mathcal{K}^*$ (i.e. $\mathcal{K}^* = \{X^* b | b \geq 0\}$) is given by

$$X^* = \begin{bmatrix} X^T & X^\perp & -X^\perp \end{bmatrix}$$

where $X^1$ is defined by the pseudoinverse of $X$ with the same dimensions.

A simplicial convex cone with $\text{rank}(X) = N = n$ is considered [14]. It is known that the dual cone $\mathcal{K}^*$ is pointed and $X^\perp$ is null. Thus $X^*$ is $X^* = X^T$. A proper non-simplicity $\mathcal{K}$ with a generator $X$ fat full-rank is considered. Assume that there exists a set of $N$ conically independent generators of an arbitrary proper polyhedral $\mathcal{K}$ in $\mathbb{R}^n$ with $\text{rank}(X) = N$ and $n \geq N$. A simpler way for locating a vertex-description of the proper cone $\mathcal{K}^*$ is to first decompose $\mathcal{K}$ into the simplicity cones $\mathcal{K}_i$ so that $\mathcal{K} = \bigcup \mathcal{K}_i$. The generator $X^*$ of the dual of $\mathcal{K}$ amounts to each generator of the dual of each simplicial part. Suppose the proper cone $\mathcal{K} \subset \mathbb{R}^n$ equals the union of $M$ simplicial cones $\mathcal{K}_i$ such that $N > n$ and $\text{rank}(X) = n$. A simpler way for locating a vertex-description of the proper dual cone $\mathcal{K}^*$ is to first decompose $\mathcal{K}$ into the simplicity cones $\mathcal{K}_i$ so that $\mathcal{K} = \bigcup \mathcal{K}_i$. The extreme directions of the dual cone are orthogonal to the facets of $\mathcal{K}$. A generator matrix $X^*$ of the dual cone $\mathcal{K}^*$ can be selected as follows:

$$X^* = \text{CI} \{X_i^T(:,j)|X_i^T x_l \geq 0\} \quad (3.5)$$

where $1 \leq i \leq M$, $1 \leq j \leq n$, and $1 \leq l \leq N$. $\text{CI}$ function is defined in Algorithm 1 as the selection method of the columnar matrix of the only conically independent vectors induced from the given set.
4. Dimension Reduction of a Positive Realization

In this section, we discuss the dimension reduction of the positive realization of DPH from the induced generating functions or DME distributions. The existence problems of positive realizations of discrete or continuous Phase-type distributions have been extensively researched [6] [3] [2]. One of the most interesting outstanding questions is to find the minimal order or minimal parameter number of the phase-type distribution. The problem of finding the minimal order of a representation for a given PH distribution is open. The PH representation problem is strongly connected with the positive realization problem which received a great deal of attention in the last theory in control theory [17]. We propose the dimension reduction of the positive realization of DPH distribution by using the concepts of the positive realization in positive system.

4.1. Positive realizations with upper bound. An upper bound on the size of nonnegative realizations for a primitive transfer functions with simple poles was presented in construction method [8]. If \( H(z) = c(I-A)^{-1}b \), then we have \( H(\alpha z) = c(\alpha z I - A)^{-1}b = \alpha^{-1}c(zI - A/\alpha)^{-1}b \). Thus, we can remark that the transfer function \( H(z) \) has a nonnegative realization if and only if \( \alpha H(\alpha z) \) has a nonnegative realization for all \( \alpha > 0 \).

Lemma 4.1. [8] A positive impulse response \( h(k) \) is given by \( h(k) = 1 + c_2 \lambda^2_k \) for \( k \geq 0 \) where \( |\lambda^2_2| < 1 \). Then there always exists a positive realization \((A_2, b_2, c_2)\) of dimensions 2.

We derive a positive realization with upper bound introduced by Hadjicostis [8].

Theorem 4.1. Let \( H(z) = \sum_{j=1}^{\infty} h_j z^{-j} \) be a rational transfer function such that \( h(k) = 1 + \sum_{i=2}^{n} c_i \lambda^i_k \geq 0 \) for \( k \geq 0 \). Let \( H_m(z) \) denote the transfer function corresponding to the shifted sequence \( \{h_m, h_{m+1}, \cdots\} \), i.e., \( H_m(z) = \sum_{j=m}^{\infty} h_j z^{-j} \). \( H_m(z) \) admits a positive realization with dimension \( 2(n-1) \).

Proof. \( h(k) \) is a finite nonnegative impulse response that has an \( N \)-dimensional nonnegative realization \((A_1, b_1, c_1)\) as follows:

\[
A_1 = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \ddots & \vdots & & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
0 & 0 & 0 & \cdots & 0
\end{bmatrix}, \quad b_1 = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}
\]

(4.1)

\[
c_1 = [h(N) \ h(N-1) \ \cdots \ h(2) \ h(1)]
\]

(4.2)

and \((A_2, b_2, c_2)\) is the given positive realization of the truncated transfer matrix \( H_N(z) \). We construct the following nonnegative realization \((A, b, c)\) of dimension \( K + s(N-1) \)

\[
A = \begin{bmatrix}
A_1 \\
b_2
\end{bmatrix}, \quad b = \begin{bmatrix}
b_1 \\
0
\end{bmatrix}, \quad c = [c_1 \ c_2]
\]

(4.3)

\[\square\]
The dimension reduction algorithm for the positive realization with the upper bound for a transfer function with simple pole can be derived. Furthermore, the construction of the positive realization of the transfer function with multiple pole was handled [18]. The construction of the positive realization of the transfer function with complex pole has been discussed in [19]. Thus we expect that we can construct a positive realization for sufficient large dimension. Therefore, without loss of generality, we can remark that there is a positive realization of the positive system for sufficiently large dimension. From now, we will propose a dimension reduction algorithm of a positive realization. First, we consider an important theorem for the necessary and sufficient conditions of the existence of a positive realization introduced in reference [12] and [13].

**Theorem 4.2.** Let $H(z)$ be a rational transfer function, with minimal realization $(F, g, h)$. Then if $H(z)$ has a positive realization $(A_+, b_+, c_+)$, there exists a generator matrix $K$ (with $K = \text{cone}(K)$) such that

1. $FK \subset K$
2. $\mathcal{R} \subset K$ where $\mathcal{R} = \text{span}\{g, Fg, F^2g, \cdots\}$.
3. $h \in K^*$.

and there is a positive realization $\{\tilde{A}_+, \tilde{b}_+, \tilde{c}_+\}$ such that

$$FK = K\tilde{A}_+, \quad g = K\tilde{b}_+, \quad \tilde{c}_+ = hK$$

where $K$ is such that $K = \text{cone}(K)$ and $\text{deg}(\tilde{A}_+) \leq \text{deg}(A_+)$. Here, $\text{deg}(A)$ is defined by the size of matrix $A$.

The first step is to demonstrate that a positive realization of $H(z)$ with sufficient large dimension can be obtained from any minimal realization if a polyhedral proper cone $K$ is found. From a given positive realization, a generator $M$ of a polyhedral cone $M$ is constructed creating a minimal positive realization.

**Lemma 4.2.** Assume that a positive realization of $H(z)$ is given by $(A_+, b_+, c_+)$, there is a nonsingular matrix $M = \begin{bmatrix} M_1 & M_2 \end{bmatrix}$ such that

$$A_+M = \begin{bmatrix} M_1 & M_2 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

(4.4)

$$b_+ = \begin{bmatrix} M_1 & M_2 \end{bmatrix} \begin{bmatrix} e_1 \\ 0 \end{bmatrix}$$

(4.5)

$$c_+ = \begin{bmatrix} M_1 & M_2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}.$$  

(4.6)

and a polyhedral cone is defined by $I = \text{cone}(M_1^T)^*$. Then we have

$$A_+M_1 = M_1A_{11}, \quad b_+ = M_1e_1, \quad c_+M_1 = f_1$$

(4.7)

and $I$ satisfies the following:

$$\text{cone}(e_1, A_{11}e_1, A_{11}^2e_1, \cdots) \subset I$$

(4.8)

$$A_{11}I \subset I$$

(4.9)

$$f_1^T \in I^*.$$  

(4.10)
Theorem 4.2 Proof. Equation (4.7) can be derived easily from equations (4.4)-(4.6) using matrix multiplication computations. Using equation (3.5), a generator of the polyhedral cone $\mathcal{I}$ can be constructed. Set $X = M_+^T$. The generator $X^*$ of $\mathcal{I}$ is created by equation (3.5); thus, $\mathcal{I} = \text{cone}(X^*)$. This ensures that $\mathcal{I}$ is finitely generated. Since $M_1 A_1^t e_1 = A_+^t b_+ \geq 0$ for all $i$ from equation (4.7), $A_i^t e_1 \in \mathcal{I} = \{\text{cone}(M_i^T)\}^*$ is derived for all $i$. Then, obtaining Equation (4.8) is straightforward. From $x \in A_1 \mathcal{I}$, $x = A_2 y$ is obtained for $y \in \mathcal{I}$. For $a \geq 0$, $x^T M_i^T a = y^T A_1^t M_i^T a = yM_i^T A_1^t a \geq 0$ is derived because $A_1 M_1 = M_1 A_1$. Equation (4.9) is obtained. Finally, it is verified $f_i^T = M_i^T c_i^+ \in \mathcal{I}^*$ since $f_i = c_i M_1$.

Lemma 4.3. Assume $\mathcal{I}$ satisfies equations (4.8), (4.9), and (4.10) where $\mathcal{I} = \{\text{cone}(M_1^T)\}^* = \text{cone}(X^*)$, $X = M_1^T$ and the dual cone generator $X^*$ is defined by (3.5). Then there is a positive realization $(\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)$ such that

$$A_1 X^* = X^* \tilde{A}_+, \quad e_1 = X^* \tilde{b}_+, \quad \tilde{c}_+ = f_1 X^*. \tag{4.11}$$

Proof. From equation (4.8), $e_1 \in \mathcal{I} = \text{cone}(X^*)$ is derived. So there exists a nonnegative vector $\tilde{b}_+$ such that $e_1 = X^* \tilde{b}_+$. From equation (4.9), $A X^* \in \mathcal{I} = \text{cone}(X^*)$ is derived. So there exists a nonnegative matrix $\tilde{A}_+^+$ such that $A_1 X^* = X^* \tilde{A}_+^+$. Finally, from equation (4.10), $f_1^T = M_1^T c_i^+$ is derived and $M_1 X^*$ is nonnegative matrix by the definition of $X^*$, i.e., $\{\text{cone}(M_1^T)\}^* = \text{cone}(X^*)$. Therefore, it can be seen that $f_1 X^* = c_i M_1 X^*$ is a nonnegative vector and $\tilde{c}_+$ is denoted by $\tilde{c}_+ = c_i M_1 X^*$. Then $\tilde{c}_+ = f_1 X^*$ is derived.

Theorem 4.2 Proof. Next, the unobservable parts induced from a positive realization are eliminated remaining $A_1, e_1, f_1$. There exists a nonsingular matrix $T = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$ such that

$$T A_1 = \begin{bmatrix} F \\ F_21 \\ F_22 \end{bmatrix},$$

$$T e_1 = \begin{bmatrix} g \\ g_2 \end{bmatrix},$$

$$f_1 = \begin{bmatrix} h \\ 0 \end{bmatrix},$$

where $(F, h)$ is a completely observable pair and $(F, g)$ is a completely controllable pair. Define $\mathcal{K} = \text{cone}(T_1 \mathcal{I})$. Then, the following is obtained

$$FK = F \text{cone}(T_1 \mathcal{I}) = FT_1 I = T_1 A_1 \mathcal{I} \subset T_1 \mathcal{I} = \mathcal{K}$$

Finally, the generator matrix $K = T_1 X^*$ can be constructed. It can be verified that $K$ satisfies the given conditions:

$$FK = K \tilde{A}_+, \quad g = K \tilde{b}_+, \quad \tilde{c}_+ = hK$$

where $K$ is such that $\mathcal{K} = \text{cone}(K)$.

In the next theorem, the dimension reduction problem of the positive realization is discussed from a given positive realization with a sufficiently larger dimension.
**Theorem 4.3.** Let $H(z)$ be a strictly proper rational transfer function. It is assumed that there is a positive realization $(A_+, b_+, c_+)$ of $H(z)$ divided into

$$A_+ = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad b_+ = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad c_+ = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

and \{F, g, h\} is a minimal realization of $H(z)$ (or, not minimal but $\deg(F) < \deg(A_+)$). Assume that there is a polyhedral proper cone $K$ satisfying the condition of Theorem 4.2 where a columnar generator $K$ satisfies $K = \text{cone}(K)$. Then there is a minimal extreme point set with $K_1 \in R_{+}^{n_1 \times n_1}$ such that $\text{cone}(K_1) = K$ for $n_1 \leq n$ and $K = [K_1 \ K_2]$ and there is a reduced positive realization $(\tilde{A}, \tilde{b}, \tilde{c})$ such that

$$FK_1 = K_1 \tilde{A}, \quad g = K_1 \tilde{b}, \quad \tilde{c} = fK_1$$

(4.12)

where there is a matrix $Q$ such that $\tilde{A} = A_1 + QA_{21} \in R_{+}^{n_1 \times n_1}$, $\tilde{b} = b_1 + Q\bar{b}_2$ and $\tilde{c} = cK_1$.

**Proof.** Using Algorithm 1, an extreme point set is located from the generator $K$. Using permutation $P$ as in equation (3.3), $PK = [K_1 \ K_2]$ can be set where $K_1$ is an extreme generator of $K$. There is a $Q \in R_{+}^{n_1 \times n}$ such that $K_2 = K_1 Q$. Thus, following is rewritten:

$$A \begin{bmatrix} K_1 & K_2 \end{bmatrix} = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Then the positive realization reduced from $(A_+, b_+, c_+)$ is given by $(\tilde{A}, \tilde{b}, \tilde{c})$ such that

$$AK_1 = K_1 \tilde{A}, \quad b = K_1 \tilde{b}, \quad \tilde{c} = cK_1.$$

\[ \square \]

**Remark 1.** The dimension of the generator matrix $M_1$ in Lemma 4.2 can be similarly reduced by transposing the matrices in Theorem 4.3 and removing the positive dependent rows of $M_1$. The transposed form method is similar to Theorem 4.3.

The dimension of the positive realization can be reduced depending on the method of choosing the induced positive realization $(\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)$ in equations (4.11) and (4.12). According to Theorem 3.1, the base feasible solutions are extreme points of feasible solutions. All base feasible solutions can be found from equation (4.11). The uncontrollable section in each induced positive realization can be compared and removed. The reduction algorithm is proposed in order to remove the uncontrollable part in positive realization. All feasible basic solutions are chosen using Theorem 3.1. For a certain $k$, $(\tilde{A}_+, \tilde{b}_+)$ with zero rows and a polyhedra generator $X^*$ can be chosen from feasible basic solutions such that

$$\tilde{A}_+ = \begin{bmatrix} A_{11} & a_{1,k} A_{13} \\ 0 & 0 & 0 \\ A_{31} & a_{2,k} A_{33} \end{bmatrix}, \quad \tilde{b}_+ = \begin{bmatrix} b_1 \\ 0 \\ b_3 \end{bmatrix}, \quad X^* = [V_1 \ v_k \ V_3]$$

(4.13)

Then, by removing the $k$-th row and column of $\tilde{A}_+$, $k$-th row of $\tilde{b}_+$ and $k$-column of $X^*$ in equation (4.13), a new reduced positive $(\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)$ and $X^*$ can be reduced. Recursively, call
it again until the dimension cannot be reduced. Finally, a dimension reduction algorithm of positive realization for a given positive system can be summarized in Algorithm 2.

**Algorithm 2** A dimension reduction algorithm of the positive realization

**Require:** Assume that a positive transfer function \( H(z) \) is given and the positive realization within the upper bound can be induced.

**Ensure:** Reduce the dimension of a positive realization \((A_+, b_+, c_+)\) as minimal as possible.

1. Find a positive realization \((A_+, b_+, c_+)\) with a sufficiently large upper bound using Lemma 4.1 and Theorem 4.1 for real value \( \lambda_i \) (\(|\lambda_i| < 1\)).
2. Using the controllability, compute \((A_{11}, e_1, f_1)\) and \(M_1\) in Lemma 4.2.
3. Remove positive dependent components of \(M_1\) using Remark 1 and Algorithm 1, and recreate \((A_{11}, e_1, f_1)\) and \(M_1\).
4. Compute \(X^*\) and \((\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)\) using Lemma 4.2.
5. Reduce a new positive realization \((\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)\) and \(X^*\) as (4.13).
6. Compute a generator \(P\), a minimal realization \((F, g, h)\) and a positive realization \((\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)\) by using the observability in Theorem 4.2.
7. Remove positive dependent components of \(P\) using Theorem 4.3 and Algorithm 1, and recreate a reduced realization \((F, g, h)\), a positive realization \((\tilde{A}_+, \tilde{b}_+, \tilde{c}_+)\) and a generator \(P\).

**Example 4.1.** The probability mass function \( f(k) \) is defined by
\[
 f(k) = \frac{\gamma}{2^k} \left\{ 1 + 5\left(-\frac{1}{8}\right)^k + 3\left(-\frac{1}{9}\right)^k \right\}
\]
for \( k \geq 0 \) and a proper \( \gamma = 0.1047 \), so called, \( f(k) \geq 0 \) and \( \sum_{k=0}^{\infty} f(k) = 1 \). Using Lemma 4.1 and Theorem 4.1, a positive realization for sufficient large dimension (i.e., the degree is 6) is obtained as follows: for \( \lambda_1 = -1/8 \) and \( \lambda_2 = -1/9 \)

\[
 A_+ = \frac{1}{2} \begin{bmatrix}
 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & 0 & 1/8 & 0 & 0 \\
 0 & 0 & 1 & 7/8 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 8/9 \\
\end{bmatrix}, \quad
 b_+ = \begin{bmatrix}
 2f(1) \\
 f(0) \\
 0.5 + 5\lambda_1^2 \\
 0.5 + 5\lambda_1^3 \\
 0.5 + 3\lambda_2^2 \\
 0.5 + 3\lambda_2^3 \\
\end{bmatrix}
\]
\[
 c_+ = \gamma \begin{bmatrix}
 0 & 1 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Using Algorithm 2, the dimension of a positive realization can be reduced and a new positive realization is obtained

\[
 \tilde{A}_+ = \begin{bmatrix}
 0.1761 & 0.0000 & 0.5294 \\
 0.0163 & 0.0000 & 0.0294 \\
 0.1456 & 0.5000 & 0.2059 \\
\end{bmatrix}, \quad
 \tilde{b}_+ = \begin{bmatrix}
 0 \\
 1 \\
 0 \\
\end{bmatrix}
\]
\[
 \tilde{c}_+ = \begin{bmatrix}
 0.0562 & 0.9426 & 0.0044 \\
\end{bmatrix}
\]
By using Theorem 2.1, a positive realization of DPH distribution is obtained as

\[
\hat{B} = \begin{bmatrix}
0.1761 & 0 & 0.8239 \\
0.0075 & 0.0000 & 0.0210 \\
0.0936 & 0.7005 & 0.2059
\end{bmatrix}, \quad \hat{b} = \begin{bmatrix}
0 \\
0.9715 \\
0
\end{bmatrix},
\]

\[
\hat{\alpha} = \begin{bmatrix}
0.0265 \\
0.9702 \\
0.0032
\end{bmatrix},
\]

where \( \hat{\alpha} = \hat{c}_+ M, \hat{B} = M^{-1} \hat{A}_+ M, \hat{\alpha} = M^{-1} \hat{c}_+ \) and \( M = diag(x) \) with \( x = (I - \hat{A}_+)^{-1} \hat{b}_+ \).

It is known that the dimension the minimal realization is larger than or equal to 3 [8]. We can see that the positive realization of the discrete phase type distribution is minimal.

5. Conclusion

We have investigated the dimension reduction algorithm of the positive realization for a given discrete phase type distribution by using the concepts of the positive system theory. The dimension of a discrete phase-type realization may be larger than its McMillan degree of probability generating functions. The dimension reduction algorithm is to find as minimal dimension of the positive realization as possible. We use the convex cone approach to provide positive realization with a lower-bound for a given positive transfer function.

References


THE ISOGEOMETRIC VARIATIONAL MULTISCALE METHOD FOR LAMINAR INCOMPRESSIBLE FLOW

YOURSELF GAFFERS MOULAGE AND HYUNG TAEK AHN

SCHOOL OF NAVAL ARCHITECTURE AND OCEAN ENGINEERING, UNIVERSITY OF ULSAN, SOUTH KOREA

E-mail address: htanh@ulsan.ac.kr

ABSTRACT. We present an application of the variational multiscale methodology to the computation of concentric annular pipe flow. Isogeometric analysis is utilized for higher order approximation of the solution using Non-Uniform Rational B-Splines (NURBS) functions. The ability of NURBS to exactly represent curved geometries makes NURBS-based isogeometric analysis attractive for the application to the flow through the curved channel.

1. INTRODUCTION

Isogeometric analysis (IGA) is a computational mechanics technology based on functions used to represent geometry [1,2,3,8,9,14]. This idea is that the functions used for the geometric representation are directly utilized for analysis. In modern Computer Aided Design (CAD) systems, NURBS are the dominant technology. When a NURBS model is constructed, the basis functions used to define the geometry can be systematically enriched by h-, p-, k-refinement, without altering the geometry or its parameterization. Hence an adaptive mesh refinement techniques can be utilized independently without a link to the CAD database, in contrast with finite element methods. A distinguishing feature of isogeometric analysis is so-called k-refinement, in which the order of functions is increased together with their continuity. As a result, isogeometric analysis allows for higher-order and higher-continuity discretizations on complex geometries.

The incompressible Navier-Stokes equations represent the mathematical model for both laminar and turbulent flow states. Two parallel lines have been flown in the past years to simulate incompressible turbulent flows that can be of engineering interest. On one side, the drawbacks of Reynolds Averaging Navier-Stokes (RANS) models combined with the impracticality to use Direct Numerical Simulation (DNS) computations for high Reynolds number problems led to...
the development of Large Eddy Simulation (LES) strategies. On the other side the numerical problems that arise when trying to solve convection dominated flow problems have motivated the development of several stabilization strategies. A key point in the development of the these stabilization methods was the appearance of the subgrid scale stabilization approach or, as originally termed, the VMS (Variation Multiscales) method [12,15]. Both approaches, LES and VMS applied to fluid dynamics, share some features like being based on a scale decomposition of the continuous velocity and pressure fields of the Navier-Stokes. In this work we employ the residual-based variational multiscale (RBVMS) turbulence modeling approach recently proposed in [3] (also see [5] for earlier reference). The modeling paradigm is based on the variational multiscale theory of turbulence [4,7,8,9,11] and the numerical experience of stabilized methods [4,18] that are residual-based. The VMS provides a theoretical framework for general multiscale problems in computational mechanics by separating the scales of interest in a predetermined number of groups, usually two, coarse-scale and fine-scale, three groups have been considered as well, coarse resolved scales, fine resolved scales, and unresolved scales (i.e. the resolved scales are further distinguished ). Another interpretation for the VMS can considered as a new technique to take account the effect of neglected unresolved fine-scale onto the behavior of coarse-scale. It also has prepared a logical proof for the stabilized methods and a platform for the development of new computational technologies ( see [3,7,11,17] for application to turbulence modeling and simulation). In VMS approach to solve turbulent flow, the scale separation is carried out by means of a projection onto the finite element space. Two equations are then obtained respectively governing the dynamics of the coarse and fine-scales. Resolved-scales are those that can be captured by the computational mesh, while unresolved-scales or subgrid scales are those not captured by the mesh. The unresolved-scale component can then be represented as the fine-scale Green’s functions of the coarse-scale residual which is to be inserted in the resolved scales equation to account for their effects. The structure of the fine-scale Green’s function was studied extensively in [17].

VMS can be utilized to solve laminar and turbulent flow. We performed an residual-based isogeometric VMS to solve laminar and turbulent channel flow (for more details see [10]). In present paper we apply IGA to construct an exact geometrical model of concentric annular channel and VMS to solve laminar flow to preserve the optimal rates of convergence regarding to curvature effect.

The structure of the paper is the following. In Section 2 a brief introduction to IGA and our general approach to build an exact geometric model are presented. In Section 3 the strong and weak forms of the incompressible Navier-Stokes equations and the discrete VMS formulation are presented. In Section 4 we present our numerical results. In Section 5 we summarize and conclude.

2. ISOGEOGRAPHIC ANALYSIS (IGA)

This section gives a very brief overview of isogeometric analysis based on NURBS and model construction as well. A more detailed description of the isogeometric approach may be found in [20].
2.1. **One-dimensional B-splines and NURBS.** From a mathematical point of view generating curves, surfaces or volumes using control points depends on some approximation or interpolation scheme which relies on the choice of basis function. NURBS basis functions are a generalization of non-rational B-spline basis functions which may be defined recursively using the Cox-deboor formulas in the one-dimensional case. Let \( U = \{u_0, ..., u_m\} \) be a nondecreasing sequence of real numbers, i.e., \( u_i \leq u_{i+1}, i = 0, ..., m - 1 \). The \( u_i \) are called knots, and \( U \) is the knot vector. The \( i^{th} \) B-spline basis function of \( p^{th} \) degree (order \( p + 1 \), denoted by \( N_{i,p}(u) \), is defined as

\[
N_{i,0}(u) = \begin{cases} 
1 & \text{if } u_i \leq u < u_{i+1} \\
0 & \text{otherwise} 
\end{cases} 
\]  

\( N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) 
\]
For the definition of a one dimension entity, i.e. a curve, a knot vector $U$ consists of $n + p + 1$ knot values, where $n$ is the number of basis functions and again the same as the number of control points. Repeated knots reduce continuity by one at the corresponding location per each knot repetition. In an open knot vector, $p + 1$ knots with the same coordinates are defined at either end of the patch. Rational B-spline entities are obtained by projective transformation of the non-rational B–spline data set, where $w_i$ is referred to as the $i^{th}$ weight.

$$R_{i,p}(u) = \frac{N_{i,p}w_i}{\sum_{j=0}^{n} N_{j,p}w_j} \quad (2.3)$$

Weights, associated with each control point is required for the above mentioned projective transformation of non-rational B-spline control point data. If all weights are equal to one non-rational B-spline are obtained. Higher-dimensional basis functions are calculated by the tensor product of the shape functions of the one-dimensional case.

2.2. Model construction. In this section a brief description of the model construction will be presented. The cylinder shown in Figure 1 has an inner radius of $R_i = 2$, and an outer radius of $R_o = 4$. The length of the cylinder is $L = 9$; for more details please see Appendix A. Our first goal in using IGA is the construction of the exact model. Toward this goal, NURBS quadratic and cubic basis functions are employed. The second important feature of IGA will be achieving $C^1$- and $C^2$-continuity across element interfaces corresponding to quadratic and cubic NURBS elements, respectively. The coarse mesh uses quadratic basis function in all three parametric directions. In order to preserve $C^1$-continuity across element interfaces when quadratic NURBS elements are employed, a model with 8 patches is constructed; see Appendix B for further details. The approach ensures $C^1$-continuity in the solution (velocity, pressure, etc.) across element interfaces inside every patches and $C^0$-continuity across the patch interfaces. $C^1$-continuity of the solution across the patch boundary is enforced by applying a constraint equation. Figure 2 illustrates the idea. Figure 2-a shows two contiguous patches with open knot vector construction. Figure 2-b shows them assembled with $C^0$-continuity achieved at the interface of the two patches. Figure 2-c depicts a $C^1$-continuous basis, which may be constructed through appropriate constraints applied to the model of Figure 2-b. Note, the end result is identical to a periodic spline basis.
Figure 2. Improving continuity of quadratic NURBS between adjacent patches. (a) Adjacent patches constructed from open knot vectors. (b) Assembly of the patches into one patch that is only $C^0$-continuous at the interface of the two original patches. (c) $C^1$-continuous basis functions, constructed by applying appropriate constraints to (b). The end result is a $C^1$-continuous periodic basis.
3. INCOMPRESSIBLE NAVIER-STOKES EQUATIONS AND RESIDUAL-BASED VARIATIONAL MULTISCALE METHOD

3.1. Strong and weak formulations. Let us start by recalling the incompressible Navier-Stokes equations. Let \( \Omega \subset \mathbb{R}^d, d = 2, 3 \), denote the spatial domain occupied by the fluid, and let \( \Gamma = \partial \Omega \) be its boundary. Then

\[
\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla p = \nabla \cdot (2\nu \nabla^s \mathbf{u}) + \mathbf{f} \text{ in } \Omega, t \in [0, T],
\]

\[
\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega, t \in [0, T],
\]

where \( \mathbf{u} \) is the velocity, \( \mathbf{f} \) is the force (per unit mass), \( \nu \) is the kinematic viscosity, \( p \) is the pressure divided by the density and \( \otimes \) denotes the tensor product (e.g. in component notation, \( [u \otimes v]_j = u_i v_j \)). Equations (3.1) and (3.2) are the balance of linear momentum and incompressibility constraint. These equations must be supplied with an initial condition of the form \( \mathbf{u} = \mathbf{u}_0 \) in \( \Omega \), \( t = 0 \) and a boundary condition which, for simplicity, will be taken as \( \mathbf{u} = 0 \) on \( \Gamma \), \( t \in [0, T] \). We also define \( \nabla^s \mathbf{u} \) as follows:

\[
\nabla^s \mathbf{u} = \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)
\]

To formulate the weak statement of the problem, \( \mathcal{Y} \) denotes both the trial solution and weighting function spaces, which are assumed to be identical. We assume \( U = \{\mathbf{u}, p\} \in \mathcal{Y} \) implies \( \mathbf{u} = 0 \) on \( \Gamma \) and \( \int_\Omega p(t)d\Omega = 0 \) for all \( t \in [0, T] \). The variational formulation is stated follows: Find \( U \in \mathcal{Y} \) such that

\[
B(\mathbf{W}, U) = L(\mathbf{W})
\]

where

\[
B(\mathbf{W}, U) = (\mathbf{w}, \frac{\partial \mathbf{u}}{\partial t})_\Omega - (\nabla \mathbf{w}, \mathbf{u} \otimes \mathbf{u})_\Omega + (q, \nabla \cdot \mathbf{u})_\Omega - (\nabla \cdot \mathbf{w}, p)_\Omega + (\nabla^s \mathbf{w}, 2\nu \nabla^s \mathbf{u})_\Omega
\]

\[
L(\mathbf{W}) = (\mathbf{w}, \mathbf{f})_\Omega
\]

3.2. Residual-based variational multiscale method. We consider a direct-sum decomposition of \( \mathcal{Y} \) into coarse-scale and fine-scale subspace, \( \mathcal{Y}^h \) and \( \mathcal{Y}' \) respectively.

\[
\mathcal{Y} = \mathcal{Y}^h \oplus \mathcal{Y}'
\]

\( \mathcal{Y}^h \) is assumed to be a finite-dimensional space, while \( \mathcal{Y}' \) is infinite-dimensional. The first step consists of the multiscale decomposition of the original fields,

\[
B(\mathbf{W}^h, \mathbf{U}^h + \mathbf{U}') = L(\mathbf{W}^h)
\]

\[
B(\mathbf{W}', \mathbf{U}^h + \mathbf{U}') = L(\mathbf{W}')
\]

where \( \mathbf{U}^h = \{\mathbf{u}^h, p^h\} \) and \( \mathbf{U}' = \{\mathbf{u}', p'\} \) stand for the coarse-scales (resolved-scale) and fine-scale (unresolved-scale) components of the solution, respectively. Equations (3.8) and (3.9) designate coarse-
and fine-scale equations, respectively. The left-hand side of equation (3.8) consists of the following terms:

\[
B(W^h, U^h + U') = (w^h, \frac{\partial u^h}{\partial t})_\Omega + (w^h, \frac{\partial u'}{\partial t})_\Omega \\
- (\nabla w^h, u^h \otimes u^h)_\Omega - (\nabla w^h, u^h \otimes u')_\Omega \\
- (\nabla w^h, u' \otimes u^h)_\Omega - (\nabla w^h, u' \otimes u')_\Omega \\
+ (q^h, \nabla \cdot u^h)_\Omega + (q^h, \nabla \cdot u')_\Omega \\
- (\nabla \cdot w^h, p^h)_\Omega - (\nabla \cdot w^h, p')_\Omega \\
+ (\nabla s w^h, 2\nu \nabla s u^h)_\Omega + (\nabla s w^h, 2\nu \nabla s u')_\Omega \\
\]

(3.10)

Also, we can rewrite (3.10) as follows:

\[
B(W^h, U^h + U') = B(W^h, U^h) \\
+ (w^h, \frac{\partial u'}{\partial t})_\Omega - (\nabla w^h, u^h \otimes u')_\Omega \\
- (\nabla w^h, u' \otimes u^h)_\Omega - (\nabla w^h, u' \otimes u')_\Omega \\
+ (q^h, \nabla \cdot u')_\Omega - (\nabla \cdot w^h, p')_\Omega + (\nabla s w^h, 2\nu \nabla s u')_\Omega \\
\]

(3.11)

where

\[
B(W^h, U^h) = (w^h, \frac{\partial u^h}{\partial t})_\Omega - (\nabla w^h, u^h \otimes u^h)_\Omega \\
+ (q^h, \nabla \cdot u^h)_\Omega - (\nabla \cdot w^h, p^h)_\Omega + (\nabla s w^h, 2\nu \nabla s u^h)_\Omega \\
\]

(3.12)

The first term on the right-hand side of equation (3.11) is referred to as the Galerkin term, which is defined in (3.12); the second term is assumed to be equal to zero because the time dependence of \( u' \) is neglected, leading to a quasi-static modeling of the fine-scales. The third and fourth terms represent cross stress and the fifth term is the Reynolds stress. The fourth and fifth terms produced by the variational multiscale method are not accounted for in classical stabilization methods, such as SUPG and GLS, which only include the third term. The last term on the right-hand side of (3.11) is assumed to be zero due to an orthogonality condition induced by the Dirichlet projector. See [3] for further details and elaboration.

Now we focus on equation (3.9) which is the fine-scale equation. The left-hand side of (3.9) consists of the following terms:

\[
B(W', U^h + U') = B(W', U^h) \\
+ (w', \frac{\partial u'}{\partial t})_\Omega - (\nabla w', u^h \otimes u')_\Omega \\
- (\nabla w', u' \otimes u^h)_\Omega - (\nabla w', u' \otimes u')_\Omega \\
+ (q', \nabla \cdot u')_\Omega - (\nabla \cdot w', p')_\Omega + (\nabla s w', 2\nu \nabla s u')_\Omega \\
\]

(3.13)
where
\[
B(W', U^h) = \left( w', \frac{\partial u^h}{\partial t} \right)_\Omega - (\nabla w', u^h \otimes u^h)_\Omega \\
+ \left( q', \nabla \cdot u^h \right)_\Omega - (\nabla \cdot w', p^h)_\Omega + (\nabla^2 w', 2\nu \nabla u^h)_\Omega
\]
(3.14)

We can rearrange equation (3.9) by considering equation (3.13) in the following form:
\[
B(W', U^h) - (W', f) + (w', L u^h u' + (\nabla \cdot u') (u^h + u'))_\Omega \\
(w', \nabla \cdot u'(u^h + u'))_\Omega + (q', \nabla \cdot u')_\Omega - (\nabla \cdot w', p')_\Omega = 0
\]
(3.15)

where
\[
L u^h u' = \frac{\partial u'}{\partial t} + u^h \cdot \nabla u' + (\nabla \cdot u^h) u' - \nabla \cdot (2\nu \nabla u')
\]
(3.16)

The first two terms in equation (3.15) can be expressed as
\[
B(W', U^h) - (W', f) = (W', \{ r_M, r_C \})
\]
(3.17)

where
\[
r_M(u^h, p^h) = \frac{\partial u^h}{\partial t} + u^h \cdot \nabla u^h \\
+ \nabla p^h - \nu \Delta u^h - f
\]
(3.18)

Finally, we can rewrite equation (3.15) as follows:
\[
(w', \mathcal{L} u^h u' + u' \cdot \nabla (u^h + u') + r_M)_\Omega + \\
(w', \nabla \cdot u'(u^h + u'))_\Omega + (q', \nabla \cdot u' + r_C)_\Omega - (\nabla \cdot w', p')_\Omega = 0
\]
(3.20)

Since solving equation (3.20) is almost as daunting as solving the original Navier-Stokes system, several simplifying assumptions are considered [3]. The assumptions are that \( \nabla \cdot u' \sim 0 \) and \( \nabla \cdot w' \sim 0 \). Thus, equation (3.20) is reduced to:
\[
(w', \mathcal{L} u^h u' + u' \cdot \nabla (u^h + u') + r_M)_\Omega = 0
\]
(3.21)

Equation (3.21) may be interpreted as the fine-scales being driven by the residual of the coarse-scale equation, \( r_M \). In addition to the above simplifying assumptions, \( u' \) is approximated through an algebraic model. We model the fine scales as in [7]:
\[
U' \approx -\tau R(U^h)
\]
(3.22)

where \( \tau \) is a \( 4 \times 4 \) matrix and \( R(U^h) \) is a \( 4 \times 1 \) vector that collects momentum and continuity residual of the Navier-Stokes equations,
\[
R(U^h) = \{ r_M(u^h, p^h), r_C(u^h) \}^T
\]
(3.23)

We define \( \tau \) as follows:
\[
\tau = \begin{bmatrix} \tau_M I_{3 \times 3} & O_3 \\ O_3 & \tau_C \end{bmatrix}
\]
(3.24)
\[
\tau_M = \left( \frac{C_t}{\Delta t^2} + u^h \cdot Gu^h + C_I \nu^2 G : G \right)^{-1/2}
\] (3.25)

where \(M = \left( C_t \Delta t^2 + u^h \cdot Gu^h + C_I \right)^{-1} \) (3.26)

\[
G = \frac{\partial \xi^T}{\partial x} \frac{\partial \xi}{\partial x}
\] (3.27)

and \(G\) a second rank metric tensor

\[
g = \sum_{j=1}^{3} \left( \frac{\partial \xi_j}{\partial x} \right)
\] (3.28)

\(x\) and \(\xi\) denote the coordinates of elements in physical and parametric space, respectively. Also, \(\Delta t\) is the time step size and \(C_t\) and \(C_I\) are non-dimensional positive constants, independent of the mesh size. \(C_t\) is set to 4 and \(C_I\) is considered \(36, 36 \times 4\) and \(36 \times 9\) for linear, quadratic and cubic elements respectively. Combining (3.8), (3.11) and (3.22), we obtain our discrete formulation: Find \(U^h\) such that

\[
B^{MS}(W^h, U^h) = L^{MS}(W^h)
\] (3.29)

where

\[
B^{MS}(W^h, U^h) = B^G(W^h, U^h) + (u^h \cdot \nabla w^h + \nabla \cdot q^h, \tau_M \cdot \nabla M)_{\Omega} + (\nabla \cdot w^h, \tau_C \cdot \nabla C)_{\Omega} + (u^h, (\nabla w^h)^T, \tau_M \cdot \nabla M)_{\Omega} - (\nabla w^h, \tau_M \cdot \nabla M)_{\Omega}
\] (3.30)

\[
L^{MS}(W^h) = (w^h, f)_{\Omega}
\] (3.31)

and

\[
B^G(W^h, U^h) = (w^h, \frac{\partial u^h}{\partial t})_{\Omega} + (\nabla \cdot \nabla \cdot w^h + 2\nu \nabla \cdot u^h)_{\Omega} - (\nabla \cdot w^h, u^h \cdot \nabla u^h)_{\Omega} + (q^h, \nabla \cdot u^h)_{\Omega} - (\nabla \cdot w^h, p^h)_{\Omega}
\] (3.32)

The superscripts MS and G stand for multiscale and Galerkin, respectively. Now, we consider the roles of the different terms in equation (3.30). The first term on the right-hand side of (3.30), defined in (3.32), is the Galerkin term; the next two terms are classical stabilization terms; and the last two terms are the additional terms produced by the variational multiscale method. From this perspective, classical stabilization methods, such as SUPG and GLS (see [18]), are viewed as only stepping stones toward the full variational multiscale method.

In the sequel, \(A\) is the nodal index in standard finite element analysis, and the control point index in NURBS-based isogeometric analysis, \(\{e_i\}\) is the \(i^{th}\) Cartesian basis vector and \(V\) and \(P\) denote the vectors of control-variable degrees-of-freedom of velocity and pressure, respectively. It is assumed
that velocity and pressure are expanded in terms of the same basis, denoted by \( \{ N_A \}_{A=1}^{n_b} \), where \( n_b \) is the number of basis functions. Two residual vectors corresponding to the momentum and continuity equations by substituting in (3.29) \( N_A e_i \) and \( N_A \) for \( w^h q^h \), respectively, which yields

\[
R^M = [R^M_A, i]
\]

(3.33)

\[
R^M_{A,i} = B^{MS}(\{N_A e_i, 0\}, \{u^h, p^h\}) - L^{MS}(\{N_A e_i, 0\})
\]

(3.34)

\[
R^C = [R^C_A]
\]

(3.35)

\[
R^C_{A} = B^{MS}(\{0, N_A\}, \{u^h, p^h\}) - L^{MS}(0, \{N_A\})
\]

(3.36)

We employ the generalized \(-\alpha \) method [6, 19]. to advance equations (3.33)-(3.36) in time. This leads to a nonlinear system of equations to be solved at each time step for which we employ a Newton-Raphson procedure. During each Newton step we solve the linear system

\[
\begin{bmatrix}
\partial R^M / \partial \mathbf{V} & \partial R^M / \partial \mathbf{P} \\
\partial R^C / \partial \mathbf{V} & \partial R^C / \partial \mathbf{P}
\end{bmatrix}
\begin{bmatrix}
\triangle \mathbf{V} \\
\triangle \mathbf{P}
\end{bmatrix}
= 
\begin{bmatrix}
-R^M \\
-R^C
\end{bmatrix}
\]

(3.37)

where \( \triangle \mathbf{V} \) and \( \triangle \mathbf{P} \) denote the solution increments of velocity and pressure. For the details of the time integration and linearization see [3].

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure3.png}
\caption{Schematic diagram, \( R_i = 2 \) \( , R_o = 4 \) and \( L = 9 \) are for the all cases.}
\end{figure}
4. LAMINAR FLOW

This section describes a laminar test case verification of the numerical formulations. The flow is chosen as test case for the following reasons: (1) The code is precisely the same one we apply to the turbulent cases; (2) The exact solution for this case is available to compare with our the numerical results; (3) NURBS are capable of exactly modeling the geometric configurations. Mesh refinement studies are performed for each order of solution approximation, namely linear, quadratic, and cubic basis functions.

![Figure 4. Laminar flow, snapshot of velocity contours for Re = 0.004, based on bulk-flow Reynolds number.](image)

4.1. Problem setup. The flow domain is described in Figure 3. A no slip Dirichlet boundary condition is set at the walls, while the axial and the circumferential directions are assigned periodic boundary conditions for every patch. The flow is driven by a constant pressure gradient, \( f_x \), acting in the stream-wise direction. The values of the kinematic viscosity \( \nu \) and the forcing \( f_x \) are set to \( 10^3 \) and 3.0, respectively.

4.2. Numerical results. The computations were performed on 16\(^3\), 32\(^3\) and 64\(^3\) elements. For all meshes we employ \( C^0 \)-continuous linear, \( C^1 \)-continuous quadratic, and \( C^2 \)-continuous cubic NURBS. For all orders, in the stream-wise and the circumferential directions the number of basis functions is equal to the number of elements in these directions. On the other hand, due to the open knot vector construction, the number of basis functions in the wall-normal direction is \( n + p \), where \( n \) is the number of elements in wall-normal the direction and \( p \) is the polynomial order. As an exact solution, the steady axial laminar flow is considered in the annular space between two concentric cylinders. Our numerical results are compared with the analytical solution, which is given in [21], namely.
The rates of convergence of the error measured in the $L^2$-norm of velocity vs. mesh parameter, are presented in Figure 5. As may be seen in Figure 5, the slopes of the lines related to linear, quadratic and cubic approximations are almost two, three and four, respectively indicating second-, third- and fourth-order accuracy respectively. This means that the VMS formulation can successfully resolve the laminar flow as well as turbulent flow, without any modifications. In other words, the same formulation can be applied to any flow problem without any a priori information about the flow regime, which is another important advantage of the current residual-based VMS method.

5. Conclusions

The variational multiscale method has been utilized for solving the incompressible Navier-Stokes equations for the laminar. The residual-based VMS formulation has been shown to perform very well for fully resolved laminar flow. Furthermore, it has been shown that the current VMS formulation preserved the optimal order of accuracy and the fine-scale modeling is consistent with the resolved coarse-scale solution. Isogeometric analysis using $C^0$-continuous linear, $C^1$-continuous quadratic and $C^2$-continuous cubic NURBS were employed to construct the model and for solution approximation.
In context of isogeometric concept, NURBS are capable of precisely modeling complex geometric configurations, we successfully employed in the complex laminar computations. Although classical finite elements are capable of approximating complex geometries, they are not capable of high-precision geometric modeling because curved geometries are modeled with piece-wise polynomial facets.

ACKNOWLEDGMENTS

This research was supported by WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology(#R33-10150), and also Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology(#2009-0065381,#2010-0004606). The authors would like to acknowledge the support from KISTI supercomputing center through the strategic support program for the supercomputing application research(#KSC-2010-C2-0013).

APPENDIX A

The cylinder shown in Figure 1 has an inner radius of \( R_i = 2 \), and an outer radius of \( R_o = 4 \). The length of the cylinder is \( L = 9 \). The coarse mesh uses quadratic NURBS basis functions in the circumferential direction, while linear in the radial and axial directions. The \( u_3 \) coordinate traverses the circumferential direction; the \( u_2 \) coordinate traverses the thickness, the \( u_1 \) coordinate the length. Table 1 represents the knot vectors for 8 patches.

<table>
<thead>
<tr>
<th>Patch No</th>
<th>( U_1 )</th>
<th>( U_2 )</th>
<th>( U_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{0,0,1,1}</td>
<td>{0,0,1,1}</td>
<td>{0,0,0,1,1,1}</td>
</tr>
<tr>
<td>2</td>
<td>{1,1,2,2}</td>
<td>{0,0,1,1}</td>
<td>{0,0,0,1,1,1}</td>
</tr>
<tr>
<td>3</td>
<td>{0,0,1,1}</td>
<td>{0,0,1,1}</td>
<td>{1,1,1,2,2,2}</td>
</tr>
<tr>
<td>4</td>
<td>{1,1,2,2}</td>
<td>{0,0,1,1}</td>
<td>{1,1,1,2,2,2}</td>
</tr>
<tr>
<td>5</td>
<td>{0,0,1,1}</td>
<td>{0,0,1,1}</td>
<td>{2,2,2,3,3,3}</td>
</tr>
<tr>
<td>6</td>
<td>{1,1,2,2}</td>
<td>{0,0,1,1}</td>
<td>{2,2,2,3,3,3}</td>
</tr>
<tr>
<td>7</td>
<td>{0,0,1,1}</td>
<td>{0,0,1,1}</td>
<td>{3,3,3,4,4,4}</td>
</tr>
<tr>
<td>8</td>
<td>{1,1,2,2}</td>
<td>{0,0,1,1}</td>
<td>{3,3,3,4,4,4}</td>
</tr>
</tbody>
</table>

The control net and weights are given in Tables 2-9. Rational solid basis functions are defined by combining the weights and one-dimensional basis functions using (A.1).

\[
R_{i,j,k}^{p,q,r}(u_1, u_2, u_3) = \frac{N_{i,p}(u_1)M_{j,q}(u_2)L_{k,r}(u_3)w_{i,j,k}}{\sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} N_{i,p}(u_1)M_{j,q}(u_2)L_{k,r}(u_3)w_{i,j,k}} \tag{A.1}
\]

where \( u_1, u_2, u_3 \) are knots in the axial, radial and circumferential directions, respectively. The number of control points are \( n, m, \) and \( l \) in the axial, radial and circumferential directions respectively. The \( N, M, L, w \) represent the basis functions in each direction and the weight and \( p, q, r \) are the polynomial orders in each direction.
Table 2. Control points and weights for cylindrical solid, patch No.1

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,0,2)</td>
<td>(9/2,0,2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,0,4)</td>
<td>(9/2,0,4)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,2,2)</td>
<td>(9/2,2,2)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,4,4)</td>
<td>(9/2,4,4)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,2,0)</td>
<td>(9/2,2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,4,0)</td>
<td>(9/2,4,0)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3. Control points and weights for cylindrical solid, patch No.2

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,0,2)</td>
<td>(9,0,2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,0,4)</td>
<td>(9,0,4)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,2,2)</td>
<td>(9,2,2)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,4,4)</td>
<td>(9,4,4)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9,2,2,0)</td>
<td>(9,2,2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9,2,4,0)</td>
<td>(9,2,4,0)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4. Control points and weights for cylindrical solid, patch No.3

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,2,0)</td>
<td>(9/2,2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,4,0)</td>
<td>(9/2,4,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,2,-2)</td>
<td>(9/2,2,-2)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,2,-4)</td>
<td>(9/2,4,-4)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,0,-2)</td>
<td>(9/2,0,-2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,0,-4)</td>
<td>(9/2,0,-4)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5. Control points and weights for cylindrical solid, patch No.4

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,2,0)</td>
<td>(9,2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,4,0)</td>
<td>(9,4,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,2,-2)</td>
<td>(9,2,-2)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,4,-4)</td>
<td>(9,4,-4)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,0,-2)</td>
<td>(9,0,-2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,0,-4)</td>
<td>(9,0,-4)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 6. Control points and weights for cylindrical solid, patch No.5

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,0,-2)</td>
<td>(9/2,0,-2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,0,-4)</td>
<td>(9/2,0,-4)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,-2,-2)</td>
<td>(9/2,-2,-2)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,-4,-4)</td>
<td>(9/2,-4,-4)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,-2,0)</td>
<td>(9/2,-2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,-4,0)</td>
<td>(9/2,-4,0)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 7. Control points and weights for cylindrical solid, patch No.6

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,0,-2)</td>
<td>(9,0,-2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,0,-4)</td>
<td>(9,0,-4)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,-2,-2)</td>
<td>(9,-2,-2)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,-4,-4)</td>
<td>(9,-4,-4)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,-2,0)</td>
<td>(9,-2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,-4,0)</td>
<td>(9,-4,0)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 8. Control points and weights for cylindrical solid, patch No.7

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,-2,0)</td>
<td>(9/2,-2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,-4,0)</td>
<td>(9/2,-4,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,-2,2)</td>
<td>(9/2,-2,2)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,-4,4)</td>
<td>(9/2,-4,4)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,0,2)</td>
<td>(9/2,0,2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,0,4)</td>
<td>(9/2,0,4)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 9. Control points and weights for cylindrical solid, patch No.8

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,-2,0)</td>
<td>(9,-2,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,-4,0)</td>
<td>(9,-4,0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,-2,2)</td>
<td>(9,-2,2)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,-4,4)</td>
<td>(9,-4,4)</td>
<td>1/$\sqrt{2}$</td>
<td>1/$\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,0,2)</td>
<td>(9,0,2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,0,4)</td>
<td>(9,0,4)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
APPENDIX B

The cylinder mentioned in section 2.2 used for our computations has an inner radius of $R_i = 2$, and an outer radius of $R_o = 4$. The length of the cylinder is $L = 9$. The coarse mesh uses quadratic in the all direction. The $u_3$ coordinate traverses the circumferential direction; the $u_2$ coordinate traverses the thickness, the $u_1$ coordinate the length. Table 10 represents the knot vectors for 8 patches.

TABLE 10. Knot vectors for cylindrical solid

<table>
<thead>
<tr>
<th>Patch No</th>
<th>$U_1$</th>
<th>$U_2$</th>
<th>$U_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{0.0,0.1,1,1}</td>
<td>{0.0,0.1,1,1}</td>
<td>{0.0,0.1,1,1}</td>
</tr>
<tr>
<td>2</td>
<td>{1.1,1.2,2,2}</td>
<td>{0.0,0.1,1,1}</td>
<td>{0.0,0.1,1,1}</td>
</tr>
<tr>
<td>3</td>
<td>{0.0,0.1,1,1}</td>
<td>{0.0,0.1,1,1}</td>
<td>{1.1,1.2,2,2}</td>
</tr>
<tr>
<td>4</td>
<td>{1.1,1.2,2,2}</td>
<td>{0.0,0.1,1,1}</td>
<td>{1.1,1.2,2,2}</td>
</tr>
<tr>
<td>5</td>
<td>{0.0,0.1,1,1}</td>
<td>{0.0,0.1,1,1}</td>
<td>{2.2,2.3,3,3}</td>
</tr>
<tr>
<td>6</td>
<td>{1.1,1.2,2,2}</td>
<td>{0.0,0.1,1,1}</td>
<td>{2.2,2.3,3,3}</td>
</tr>
<tr>
<td>7</td>
<td>{0.0,0.1,1,1}</td>
<td>{0.0,0.1,1,1}</td>
<td>{3.3,3.4,4,4}</td>
</tr>
<tr>
<td>8</td>
<td>{1.1,1.2,2,2}</td>
<td>{0.0,0.1,1,1}</td>
<td>{3.3,3.4,4,4}</td>
</tr>
</tbody>
</table>

The control net and weights are given in Tables 11-18. Rational solid basis functions are defined by combining the weights and one-dimensional basis functions using (A.1).

TABLE 11. Control points and weights for cylindrical solid, patch No.1

<table>
<thead>
<tr>
<th>$j$</th>
<th>$k$</th>
<th>$B_{0,j,k}$</th>
<th>$B_{1,j,k}$</th>
<th>$B_{2,j,k}$</th>
<th>$w_{0,j,k}$</th>
<th>$w_{1,j,k}$</th>
<th>$w_{2,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,0,2)</td>
<td>(9/4,0,2)</td>
<td>(9/2,0,2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,0,3)</td>
<td>(9/4,0,3)</td>
<td>(9/2,0,3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(0,0,4)</td>
<td>(9/4,0,4)</td>
<td>(9/2,0,4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,2,2)</td>
<td>(9/4,2,2)</td>
<td>(9/2,2,2)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,3,3)</td>
<td>(9/4,3,3)</td>
<td>(9/2,3,3)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(0,4,4)</td>
<td>(9/4,4,4)</td>
<td>(9/2,4,4)</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,2,0)</td>
<td>(9/4,2,0)</td>
<td>(9/2,2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,3,0)</td>
<td>(9/4,3,0)</td>
<td>(9/2,3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(0,4,0)</td>
<td>(9/4,4,0)</td>
<td>(9/2,4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 12. Control points and weights for cylindrical solid, patch No.2

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>B₀,j,k</th>
<th>B₁,j,k</th>
<th>B₂,j,k</th>
<th>w₀,j,k</th>
<th>w₁,j,k</th>
<th>w₂,j,k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,0,2)</td>
<td>(27/4,0,2)</td>
<td>(9,0,2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,0,3)</td>
<td>(27/4,0,3)</td>
<td>(9,0,3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(9/2,0,4)</td>
<td>(27/4,0,4)</td>
<td>(9,0,4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,2,2)</td>
<td>(27/4,2,2)</td>
<td>(9,2,2)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,3,3)</td>
<td>(27/4,3,3)</td>
<td>(9,3,3)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(9/2,4,4)</td>
<td>(27/4,4,4)</td>
<td>(9,4,4)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,2,0)</td>
<td>(27/4,2,0)</td>
<td>(9,2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,3,0)</td>
<td>(27/4,3,0)</td>
<td>(9,3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(9/2,4,0)</td>
<td>(27/4,4,0)</td>
<td>(9,4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 13. Control points and weights for cylindrical solid, patch No.3

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>B₀,j,k</th>
<th>B₁,j,k</th>
<th>B₂,j,k</th>
<th>w₀,j,k</th>
<th>w₁,j,k</th>
<th>w₂,j,k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,2,0)</td>
<td>(9/4,2,0)</td>
<td>(9/2,2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,3,0)</td>
<td>(9/4,3,0)</td>
<td>(9/2,3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(0,4,0)</td>
<td>(9/4,4,0)</td>
<td>(9/2,4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,2,-2)</td>
<td>(9/4,2,-2)</td>
<td>(9,2,-2)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,3,-3)</td>
<td>(9/4,3,-3)</td>
<td>(9,3,-3)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(0,4,-4)</td>
<td>(9/4,4,-4)</td>
<td>(9,4,-4)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,0,-2)</td>
<td>(9/4,0,-2)</td>
<td>(9,0,-2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,0,-3)</td>
<td>(9/4,0,-3)</td>
<td>(9,0,-3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(0,0,-4)</td>
<td>(9/4,0,-4)</td>
<td>(9,0,-4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 14. Control points and weights for cylindrical solid, patch No.4

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>B₀,j,k</th>
<th>B₁,j,k</th>
<th>B₂,j,k</th>
<th>w₀,j,k</th>
<th>w₁,j,k</th>
<th>w₂,j,k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,2,0)</td>
<td>(27/4,2,0)</td>
<td>(9,2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,3,0)</td>
<td>(27/4,3,0)</td>
<td>(9,3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(9/2,4,0)</td>
<td>(27/4,4,0)</td>
<td>(9,4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,2,-2)</td>
<td>(27/4,2,-2)</td>
<td>(9,2,-2)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,3,-3)</td>
<td>(27/4,3,-3)</td>
<td>(9,3,-3)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(9/2,4,-4)</td>
<td>(27/4,4,-4)</td>
<td>(9,4,-4)</td>
<td>1/√2</td>
<td>1/√2</td>
<td>1/√2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,0,-2)</td>
<td>(27/4,0,-2)</td>
<td>(9,0,-2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,0,-3)</td>
<td>(27/4,0,-3)</td>
<td>(9,0,-3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(9/2,0,-4)</td>
<td>(27/4,0,-4)</td>
<td>(9,0,-4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 15. Control points and weights for cylindrical solid, patch No.5

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>(B_{0,j,k})</th>
<th>(B_{1,j,k})</th>
<th>(B_{2,j,k})</th>
<th>(w_{0,j,k})</th>
<th>(w_{1,j,k})</th>
<th>(w_{2,j,k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,0,-2)</td>
<td>(9/4,0,-2)</td>
<td>(9/2,0,-2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,0,-3)</td>
<td>(9/4,0,-3)</td>
<td>(9/2,0,-3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(0,0,-4)</td>
<td>(9/4,0,-4)</td>
<td>(9/2,0,-4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,-2,-2)</td>
<td>(9/4,-2,-2)</td>
<td>(9/2,-2,-2)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,-3,-3)</td>
<td>(9/4,-3,-3)</td>
<td>(9/2,-3,-3)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(0,-4,-4)</td>
<td>(9/4,-4,-4)</td>
<td>(9/2,-4,-4)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,-2,0)</td>
<td>(9/4,-2,0)</td>
<td>(9/2,-2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,-3,0)</td>
<td>(9/4,-3,0)</td>
<td>(9/2,-3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(0,-4,0)</td>
<td>(9/4,-4,0)</td>
<td>(9/2,-4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 16. Control points and weights for cylindrical solid, patch No.6

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>(B_{0,j,k})</th>
<th>(B_{1,j,k})</th>
<th>(B_{2,j,k})</th>
<th>(w_{0,j,k})</th>
<th>(w_{1,j,k})</th>
<th>(w_{2,j,k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,0,-2)</td>
<td>(27/4,0,-2)</td>
<td>(9,0,-2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,0,-3)</td>
<td>(27/4,0,-3)</td>
<td>(9,0,-3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(9/2,0,-4)</td>
<td>(27/4,0,-4)</td>
<td>(9,0,-4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,-2,-2)</td>
<td>(27/4,-2,-2)</td>
<td>(9,-2,-2)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,-3,-3)</td>
<td>(27/4,-3,-3)</td>
<td>(9,-3,-3)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(9/2,-4,-4)</td>
<td>(27/4,-4,-4)</td>
<td>(9,-4,-4)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,-2,0)</td>
<td>(27/4,-2,0)</td>
<td>(9,-2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,-3,0)</td>
<td>(27/4,-3,0)</td>
<td>(9,-3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(9/2,-4,0)</td>
<td>(27/4,-4,0)</td>
<td>(9,-4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 17. Control points and weights for cylindrical solid, patch No.7

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>(B_{0,j,k})</th>
<th>(B_{1,j,k})</th>
<th>(B_{2,j,k})</th>
<th>(w_{0,j,k})</th>
<th>(w_{1,j,k})</th>
<th>(w_{2,j,k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0,-2,0)</td>
<td>(9/4,-2,0)</td>
<td>(9/2,-2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(0,-3,0)</td>
<td>(9/4,-3,0)</td>
<td>(9/2,-3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(0,-4,0)</td>
<td>(9/4,-4,0)</td>
<td>(9/2,-4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(0,-2,2)</td>
<td>(9/4,-2,2)</td>
<td>(9/2,-2,2)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,-3,3)</td>
<td>(9/4,-3,3)</td>
<td>(9/2,-3,3)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(0,-4,4)</td>
<td>(9/4,-4,4)</td>
<td>(9/2,-4,4)</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
<td>(1/\sqrt{2})</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(0,0,2)</td>
<td>(9/4,0,2)</td>
<td>(9/2,0,2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(0,0,3)</td>
<td>(9/4,0,3)</td>
<td>(9/2,0,3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(0,0,4)</td>
<td>(9/4,0,4)</td>
<td>(9/2,0,4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 18. Control points and weights for cylindrical solid, patch No.8

<table>
<thead>
<tr>
<th>j</th>
<th>k</th>
<th>( B_{0,j,k} )</th>
<th>( B_{1,j,k} )</th>
<th>( B_{2,j,k} )</th>
<th>( w_{0,j,k} )</th>
<th>( w_{1,j,k} )</th>
<th>( w_{2,j,k} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(9/2,-2,0)</td>
<td>(27/4,-2,0)</td>
<td>(9,-2,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(9/2,-3,0)</td>
<td>(27/4,-3,0)</td>
<td>(9,-3,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(9/2,-4,0)</td>
<td>(27/4,-4,0)</td>
<td>(9,-4,0)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(9/2,-2,2)</td>
<td>(27/4,-2,2)</td>
<td>(9,-2,2)</td>
<td>1/( \sqrt{2} )</td>
<td>1/( \sqrt{2} )</td>
<td>1/( \sqrt{2} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(9/2,-3,3)</td>
<td>(27/4,-3,3)</td>
<td>(9,-3,3)</td>
<td>1/( \sqrt{2} )</td>
<td>1/( \sqrt{2} )</td>
<td>1/( \sqrt{2} )</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>(9/2,-4,4)</td>
<td>(27/4,-4,4)</td>
<td>(9,-4,4)</td>
<td>1/( \sqrt{2} )</td>
<td>1/( \sqrt{2} )</td>
<td>1/( \sqrt{2} )</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>(9/2,0,2)</td>
<td>(27/4,0,2)</td>
<td>(9,0,2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(9/2,0,3)</td>
<td>(27/4,0,3)</td>
<td>(9,0,3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(9/2,0,4)</td>
<td>(27/4,0,4)</td>
<td>(9,0,4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### References


