<table>
<thead>
<tr>
<th>Title</th>
<th>A fast tree-code algorithm for the vortex method on a sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>SAKAJO, Takashi</td>
</tr>
<tr>
<td>Citation</td>
<td>Hokkaido University Preprint Series in Mathematics, 847, 1-19</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2007-04-23</td>
</tr>
<tr>
<td>DOI</td>
<td>10.14943/83997</td>
</tr>
<tr>
<td>Doc URL</td>
<td><a href="http://hdl.handle.net/2115/69656">http://hdl.handle.net/2115/69656</a></td>
</tr>
<tr>
<td>Type</td>
<td>bulletin (article)</td>
</tr>
<tr>
<td>File Information</td>
<td>pre847.pdf</td>
</tr>
</tbody>
</table>
A fast tree-code algorithm for the vortex method on a sphere *

Takashi Sakajo
Department of mathematics, Hokkaido University
April 23, 2007

Abstract
A fast and accurate algorithm to compute interaction between \( N \) point vortices on a sphere is proposed. It is an extension of the fast tree-code algorithm based on the Taylor expansion developed by Draghicescu for the point vortices in the plane. When we choose numerical parameters in the fast algorithm suitably, the computational cost of \( O(N^2) \) is reduced to \( O(N \log N)^4 \) and the approximation error decreases like \( O(1/N) \) as \( N \to \infty \), which are clearly confirmed in the present article. We also apply the fast method to a long-time evolution of two vortex sheets on the sphere. A key device is to embed the sphere into the three-dimensional space, in which we reformulate the equation of motion for the \( N \) point vortices.

Keywords: fast tree-code algorithms, flows on sphere, vortex method, vortex sheet

AMS subject classification(MSC): 65Y20, 76B47, 76M23, 65C20

1 Introduction
We propose a fast numerical algorithm to compute the evolution of the incompressible and inviscid fluid confined in the surface of a sphere \( S \) with radius \( R \). Since the vorticity, which is denoted by a scalar function \( \omega_0(\theta, \phi) \) in the spherical coordinates \( (\theta, \phi) \), is conserved along the path of the fluid particle, it is possible to approximate the evolution of the vorticity region with that of many point vortices; Let us consider a bounded vorticity region, whose support is denoted by \( \mathcal{A} \subset S \). Then, discretizing the region with \( N \) meshes, we set a point vortex at a

\footnote{Mailing address: Kita 10 Nishi 8, Sapporo Hokkaido, 060-0810 JAPAN, E-mail: sakajo@math.sci.hokudai.ac.jp, TEL: +81-11-706-4660, FAX: +81-11-727-3705}
certain position \((\theta_m, \phi_m)\) in each mesh for \(m = 1, 2, \ldots, N\). The strength of the point vortex \(\Gamma_m\) is represented by

\[
\Gamma_m = \frac{m(A)}{N} \omega_0(\theta_m, \phi_m),
\]

in which \(m(A) \leq 4\pi R\) is a measure of the vorticity region \(A\) on the sphere. When the vorticity field is approximated by

\[
\omega_0(\theta, \phi) \approx \frac{1}{\sin \theta} \sum_{m=1}^{N} \Gamma_m \delta(\theta - \theta_m, \phi - \phi_m),
\]

the Euler equations are reduced to the equations for the discretized \(N\) point vortices\[11\], which are given by

\[
\dot{\theta}_m = -\frac{1}{4\pi R^2} \sum_{j \neq m}^{N} \frac{\Gamma_j \sin \theta_j \sin(\phi_m - \phi_j)}{1 + \sigma^2 - \cos \theta_m \cos \theta_j - \sin \theta_m \sin \theta_j \cos(\phi_m - \phi_j)}, \tag{2}
\]

\[
\dot{\phi}_m = -\frac{1}{4\pi R^2 \sin \theta_m} \sum_{j \neq m}^{N} \frac{\Gamma_j [\cos \theta_m \sin \theta_j \cos(\phi_m - \phi_j) - \sin \theta_m \cos \theta_j]}{1 + \sigma^2 - \cos \theta_m \cos \theta_j - \sin \theta_m \sin \theta_j \cos(\phi_m - \phi_j)}, \tag{3}
\]

for \(m = 1, 2, \ldots, N\). The parameter \(\sigma\) is introduced to regularize the singular behavior of the velocity fields (2) and (3) when one point vortex approaches another point closely. This approximation method is known as the point vortex method for \(\sigma = 0\), and the vortex blob method for \(\sigma \neq 0\). Many references regarding the theory and practice of the methods are found in [2]. Not only the vortex method is of practical use in the numerical investigations for the Euler flows, but also it is mathematically shown that the solution of the eqs. (2) and (3) uniformly converges to that of the Euler equations when \(N \to \infty\) as long as the solution of the Euler equations is smooth\[10\].

Another computational method for the incompressible and inviscid flows on the sphere is the contour method with surgery technique developed by Dritschel\[6\]. In the method, approximating the vorticity region \(A\) by some subregions with piecewise constant vorticity, one has only to track the evolution of their boundaries, since the velocity field induced by the constant vorticity region is represented by the path integral along the boundary owing to Green’s formula. The contour method has been used to compute the evolutions of the constant vorticity strips\[7\] and the polar constant vorticity cap\[18\]. It is also give us an effective way to consider flows on the rotating sphere, since the effect of rotation is approximated by the constant vorticity strips corresponding to the background solid-body rotation\[3, 17\].

The contour method has been considered to be more advantageous than the vortex method. This is because we have only to discretize the boundary of the
region in the contour method, whereas the entire vorticity region should be discretized in the vortex method. Furthermore, numerical computation of the vortex method gives rise to another serious difficulty in the computational cost. Namely, we need \( O(N) \) amount of computations to evaluate the velocity fields (2) and (3) for one point vortex. Accordingly, the total of \( O(N^2) \)-operations are required to finish the evaluation for all the point vortices. In spite of the shortcomings, nevertheless, the vortex method still has an benefit in the approximation continuous but non-constant vorticity regions and the singular vorticity distribution such as vortex sheets. Actually, the vortex blob method has been used to compute a complex pattern formation in a single vortex sheet[21]. Thus, if we resolve the difficulty in the computational cost, the vortex method would be a practical choice to compute the incompressible and inviscid flows on the sphere.

The purpose of the present study is to give a fast algorithm to make the \( O(N^2) \) evaluations faster. The same difficulty arises in the numerical computation of the interaction between the \( N \) charges and the \( N \) gravitational bodies, for which the fast algorithms such as the fast tree-code algorithm[1, 4] and the fast multipole method[8, 9] have been developed. The fast methods basically work as follows; Suppose that the \( N \) points are assigned in the computational domain. Then in order to evaluate the velocity field induced by the points at a given position, the \( N \) points are divided into two sets, called the far-field and the near-field. The contribution from the far-field is evaluated by using a certain approximation to the velocity field such as the Taylor expansion and the multipole expansion. As for the points in the near-field, they are computed directly. Owing to the efficient far-field approximation, the fast methods reduce the computational cost to \( O(N \log N) \) or \( O(N^d) \), where \( d \) is the dimension of the computational domain.

Regarding the point vortex approximation for the two-dimensional Euler equations, Draghicescu et al. proposed a fast tree-code algorithm based on the Taylor expansion[4], which was successfully applied to the numerical computations of vortex sheets[5, 19]. The fast method was extended to the three-dimensional Euler equations and used to compute three-dimensional vortex sheets[15, 20]. Note that since the vorticity is no longer a conserved quantity in the three-dimensional case, we need to incorporate the vortex stretching mechanism into the algorithm.

In the present paper, we extend the fast tree-code algorithm to compute the velocity field induced by the point vortices on the sphere. The extension is certainly possible in theory, but it is quite hard to implement the numerical computation in practice, since the expressions of the Taylor coefficients for (2) and (3) are too complicated to compute them efficiently. Since the efficiency of the fast algorithm is canceled out due to this drawback, the vortex method for the Euler equations on the sphere has been regarded as unrealistic so far. In order to resolve the problem, we embed the sphere in the three-dimensional space \( \mathbb{R}^3 \) and rewrite the eqs. (2) and (3) with respect to the three-dimensional coordinates.
system; Let the positions of the point vortices in $\mathbb{R}^3$ be represented by

$$x_m = (x_m(t), y_m(t), z_m(t)) = (R \sin \theta_m \cos \phi_m, R \sin \theta_m \sin \phi_m, R \cos \theta_m).$$

Then the stream function $\psi(x)$ is recovered from the vorticity field with the inversion formula

$$\psi(x) = \int \int_S G(x, x') \omega(x) dA,$$

where the Green function on the sphere is given by $G(x, x') = -(1/4\pi) \log |x-x'|^2$ [11]. With the singular vorticity field approximation (1), the stream function gives rise to the equation for the $N$ point vortices:

$$\dot{x}_m = -\frac{1}{4\pi R} \sum_{j \neq m}^N \Gamma_j \frac{x_m \times x_j}{R^2 + \sigma^2 - x_m \cdot x_j}, \quad m = 1, 2, \ldots, N, \quad (4)$$

for which we develop a fast tree-code algorithm. See the references [16, 17] for the derivation of the equation.

The paper consists of four sections. In the next section, we give a formal description of the fast algorithm and its error estimate. In the third section, some numerical examples are shown to confirm the efficiency of the algorithm. The last section is devoted to summary and discussion on future applications.

## 2 Fast tree-code algorithm

### 2.1 Mesh generation, far-field and near-field

Since we describe the fast algorithm for the sphere embedded in $\mathbb{R}^3$, we define the computational domain $B$ as a three-dimensional box, a little bit larger than the sphere of radius $R$, i.e.,

$$B = \left[-(1+\delta)R, (1+\delta)R\right]^3.$$

The positive parameter $\delta$ is taken sufficiently small but not zero so that the box $B$ contains the sphere completely. Then we construct a tree structure of small boxes in $B$ by dividing the region into two smaller boxes recursively. Since the point vortices are restricted on the sphere, we eliminate the boxes that have no intersection with the sphere from the tree structure, which helps us save the working memory on the computer. In what follows, without loss of generality, let us set the radius of the sphere $R = 0.5$ so that the box $B$ becomes a unit box, and the number of the point vortices $N = 2^l$, $l' = 2l$ so that the finest box size is $h = 1/\sqrt{N} = 2^{-l}$.

First, we generate the tree structure with the following recursive algorithm for a given box $\tau = [x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$. The algorithm contains the
initialization of parameters $\rho(\tau), y_\tau, A^k_\tau, B^k_\tau, C^k_\tau$ and $D^k_\tau$. The center point of the mesh $y_\tau$ and its radius $\rho(\tau)$ are defined by
\[
y_\tau = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2}\right), \quad \rho(\tau) = \sup_{y \in \tau} |y - y_\tau|.
\]

The parameters $A^k_\tau, B^k_\tau, C^k_\tau$ are required in the far-field computation and their definitions are provided in the next subsection, while $L(\tau)$ denotes a list of the point vortices contained in the finest box $\tau$, which are used to compute the near-field evaluation.

**Algorithm 1.** *(Generate the tree structure of boxes)*

**Input:** box $\tau$ and integer $k$;
GenerateMesh( $\tau, k$ )
    if $k = 3l$ return;
    $y_\tau = (c_{\tau_1}, c_{\tau_2}, c_{\tau_3}) = \left(\frac{1}{3}(x_1 + x_2), \frac{1}{3}(y_1 + y_2), \frac{1}{3}(z_1 + z_2)\right)$;
    Compute the radius of $\tau$, $\rho(\tau)$;
    For all $k = (k_1, k_2, k_3), |k| \leq \lambda - 1$, initialize $A^k_\tau, B^k_\tau, C^k_\tau$;
    Initialize the list of the near fields, $L(\tau)$;
    if $k \mod 3 = 1$
        $\tau_1 = [x_1, c_{\tau_1}] \times [y_1, y_2] \times [z_1, z_2]; \tau_2 = [c_{\tau_1}, x_2] \times [y_1, y_2] \times [z_1, z_2];$
    else if $k \mod 3 = 2$
        $\tau_1 = [x_1, x_2] \times [y_1, c_{\tau_2}] \times [z_1, z_2]; \tau_2 = [x_1, x_2] \times [c_{\tau_2}, y_2] \times [z_1, z_2];$
    else if $k \mod 3 = 0$
        $\tau_1 = [x_1, x_2] \times [y_1, y_2] \times [z_1, c_{\tau_3}]; \tau_2 = [x_1, x_2] \times [y_1, y_2] \times [c_{\tau_3}, z_2];$
    end
    For each $\tau_i$, if $\tau_i \cap S \neq \emptyset$
        set $\tau_i$ as a child box of $\tau$;
        recursive call GenerateMesh( $\tau_i, k + 1$ );
        return;
    else
        set $\emptyset$ as a child of $\tau$;
        return;
    end
End of Algorithm

When we apply the algorithm to the computational domain $\mathcal{B}$, we obtain the tree structure of small boxes, which is denoted by $\Sigma$ in what follows. The size of box in the finest level $k = 3l$ is equivalent to $h$.

Next, for a given point $x \in S$, we define the set of boxes, called the far field of $x$, $\mathcal{F}(x)$, and the near field of $x$, $N(x)$ as in [4].
Definition 1. (Far field and near field) For $x \in S$, $F(x)$ contains all boxes $\tau$ with center $y_\tau$ such that the following condition is satisfied.

$$\rho(\tau) \leq h^\nu |R^2 - x \cdot y_\tau|,$$

and $\tau$ is maximal. Let $F(x) = \bigcup F(x)$. The parameter $\nu > 0$ is determined to control the efficiency of the fast algorithm and the accuracy of the numerical computation. The near field of $x$ is defined by $N(x) = \Sigma \setminus F(x)$.

The definitions of the far field and near field are almost similar to those given in [4] except for the condition (5) defining the far field. Let us remember that the original far-field condition in [4] is

$$\rho(\tau) \leq h^\nu |x - y_\tau|.$$  

(6)

The change of the far-field condition could affect the efficiency of the fast algorithm. However, as a matter of fact, it is not a serious problem, since $|R^2 - x \cdot y_\tau| \leq |x - y_\tau|$ holds for arbitrary $y_\tau \in \mathbb{R}^3$ and $|x| = R$. Hence, the far field of $x$ in the present algorithm becomes automatically that of the three-dimensional fast tree-code algorithm.

2.2 Far field approximation

The target of the fast algorithm is to evaluate the following velocity field $u_N(x, t)$ in the eq. (4):

$$u_N(x, t) = -\frac{1}{4\pi R} \sum_{j=1}^{N} \frac{x \times y_j}{(R^2 + \sigma^2 - x \cdot y_j)^2} \equiv -\frac{1}{4\pi R} \sum_{j=1}^{N} \Gamma_j \gamma(x, y_j) D(x, y_j),$$

(7)

where $\gamma(x, y) = x \times y$ and $D(x, y) = (R^2 + \sigma^2 - x \cdot y)^{-1}$. The contribution from the near field is computed directly, and that from the far field is approximated by the Taylor expansion of $D$ up to the $(\lambda - 1)$th order. That is to say, the velocity field $u_N$ is approximated by

$$u_N^\lambda(x, t) = -\frac{1}{4\pi R} \sum_{y_j \in N(x)} \Gamma_j \gamma(x, y_j) D(x, y_j) - \frac{1}{4\pi R} \sum_{\tau \in F(x)} u_N^{\lambda\tau}(x, t),$$

(8)

in which

$$u_N^{\lambda\tau}(x, t) = \sum_{y_j \in \tau} \Gamma_j \gamma(x, y_j) \sum_{|k| \leq \lambda-1} a_k(x, y_\tau) (y_j - y_\tau)^k.$$  

(9)

The $k$th order Taylor coefficients of $D$ is given explicitly by

$$a_k(x, y_\tau) = \frac{1}{k!} D_k D(x, y)|_{y=y_\tau} = \frac{|k|!}{k!} (R^2 + \sigma^2 - x \cdot y_\tau)^{-|k|-1} x^k,$$

(10)
where \( k = (k_1, k_2, k_3) \) is the multiple index, with which \( k! = k_1!k_2!k_3! \), \(|k| = k_1 + k_2 + k_3 \), and \( D^k_y = \frac{\partial^{\left|k\right|}}{\partial y_1^{k_1} \partial y_2^{k_2} \partial y_3^{k_3}} \), \( y^k = y_1^{k_1} y_2^{k_2} y_3^{k_3} \) for \( y = (y_1, y_2, y_3) \). The explicit representation of the Taylor coefficients (10) saves the practical computational time greatly. Denoting \( y_j = (y_{j1}, y_{j2}, y_{j3}) \), we reduce the far-field approximation (9)\footnote{\textit{Equation number}} to

\[
\begin{align*}
  u^\lambda_N(x, t) = \sum_{|k| \leq \lambda - 1} a_k(x, y_\tau) \left( x_2 C^k_\tau - x_3 B^k_\tau, x_3 A^k_\tau - x_1 C^k_\tau, x_1 B^k_\tau - x_2 A^k_\tau \right),
\end{align*}
\]

in which \( A^k_\tau, B^k_\tau, C^k_\tau \) are defined by

\[
\begin{align*}
  A^k_\tau &= \sum_{y_j \in \tau} \Gamma_j y_j (y_j - y_\tau)^k, \\
  B^k_\tau &= \sum_{y_j \in \tau} \Gamma_j y_{j2} (y_j - y_\tau)^k, \\
  C^k_\tau &= \sum_{y_j \in \tau} \Gamma_j y_{j3} (y_j - y_\tau)^k.
\end{align*}
\]

These coefficients are associated to the box \( \tau \in F(x) \) that contains the point vortex \( y_j \). They are computed for all the point vortices in advance and reused in the far-field approximation, which allows us to evaluate the velocity field faster.

\subsection*{2.3 Description of the algorithm}

Here we give the formal description of the tree-code algorithm. First, we define the following two recursive algorithms called in the main algorithm.

\textbf{Algorithm 2.} (Compute the coefficients \( A^k_\tau, B^k_\tau, C^k_\tau \) for all the node box \( \tau \) containing the point vortex at \( y \) with the strength \( \Gamma \))

\begin{algorithm}
Input: box \( \tau \), integer \( k \), position of the point vortex \( y \) with the strength \( \Gamma \);
ComputeNodeCoefficients( \( \tau \), \( k \), \( y \), \( \Gamma \) )
  \begin{algorithmic}
    \State if \( \tau = \emptyset \) return;
    \State if \( y_j \in \tau \) then for all \( k \), \( |k| \leq \lambda - 1 \)
      \State add \( \Gamma_j y_1 (y_j - y_\tau)^k \) to \( A^k_\tau \);
      \State add \( \Gamma_j y_2 (y_j - y_\tau)^k \) to \( B^k_\tau \);
      \State add \( \Gamma_j y_3 (y_j - y_\tau)^k \) to \( C^k_\tau \);
    \State if \( k = 3l \) then
      \State add \( y_j \) to the list of the near field, \( L(\tau) \);
    \State else
      \State Recursively call ComputeNodeCoefficients( \( \tau_i, k+1, y, \Gamma \) ) for all the children of \( \tau \), \( \tau_1 \) and \( \tau_2 \);
  \end{algorithmic}
\end{algorithm}
Algorithm 3. *(Compute the velocity field for the point vortex at \( y \))*

*Input:* box \( \tau \), integer \( k \), the position of the point vortex \( y \);

\begin{verbatim}
ComputeFarNearField( \( \tau \), \( k \), \( y \) )
    if \( \tau = \emptyset \) return;
    if \( \rho(\tau) < h^\nu |R^2 - y \cdot y_\tau| \) then
        Compute the far field approximation according to (11);
        return;
    else
        if \( k = 3l \) then
            Compute the contribution from the point vortices in the near field list \( L(\tau) \) directly;
            return;
        else
            Recursively call ComputeFarNearField( \( \tau_i \), \( k + 1 \), \( y \) ) for all the children of \( \tau \), \( \tau_1 \) and \( \tau_2 \);
        end
    end
end
end
end

End of Algorithm

Algorithm 4. *(Fast Tree Code for the point vortices points on sphere)*

*Input:* integer \( n \); real \( \lambda \); real \( \nu \); real \( \Gamma_j \); real \( y_j \), \( j = 1, \ldots, n \);

*Output:* \( u_h^k(x, t) \) for all \( x = y_j \), \( j = 1, \ldots, n \);

*Stage 0 (mesh generation, done only once)*

\begin{verbatim}
GenerateMesh( \( B \), 0 );
end
end
end

*Stage 1 (Compute coefficients \( A^k, B^k, C^k, D^k \))*

\begin{verbatim}
For \( j = 1, \ldots, n \), call ComputeNodeCoefficients( \( B \), 0, \( y_j \), \( \Gamma_j \) );
end
end

*Stage 2 (Compute the velocity field (7))

\end{verbatim}
For \( j = 1, \ldots, n \), call \( \text{ComputeFarNearField}( B, 0, y_j ) \);
end

End of Algorithm

The description of the fast tree algorithm is substantially similar to those given in the paper [4], though it is written with the recursive calls. The definitions of the mesh-generation algorithm, the far-field condition and the coefficients \( A^k, B^k, C^k \) are different. Since the point vortices in the far field satisfying (5) hold the far-field condition (6) for the three-dimensional fast tree-code algorithm, the computational cost becomes \( O(N\lambda^3 \log N) \) as in [4]. Moreover, when we choose the order of the Taylor approximation as \( \lambda = O(\log N) \), it becomes \( O(N(\log N)^4) \). The reason why the we see the similarity in the computational cost is that the present fast algorithm is described by embedding the sphere in the three-dimensional space.

2.4 Error estimate

We give an error estimate of the fast algorithm for \( \sigma = 0 \), which is easily extended to the case of \( \sigma \neq 0 \). Let the constant \( C \) be defined by

\[
C = \max_{1 \leq j \leq N} \frac{m(A)}{4\pi R} \omega_0(\theta_j, \phi_j) = N \max_{1 \leq j \leq N} \frac{\Gamma_j}{4\pi R}.
\]

Then, the error estimate becomes

\[
\left| u_N(x, t) - u_N^\sigma(x, t) \right| = \left| \frac{1}{4\pi R} \sum_{\tau \in F(x)} \sum_{y_j \in \tau} \Gamma_j \gamma(x, y_j) \sum_{|k| = \lambda} a_k(x, y_\tau)(y_j - y_\tau)^k \right|
\leq \frac{C}{N} \sum_{\tau \in F(x)} \sum_{y_j \in \tau} |\gamma_j| \sum_{|k| = \lambda} |a_k(x, y_\tau)||y_j - y_\tau|^k. \quad (13)
\]

Let us note that \( |\gamma(x, y_j)| = |x \times y_j| \leq R^2 \) due to \( |x| = |y_j| = R \) and \( |y_j - y_\tau| \leq \rho(\tau) \) for \( y_j \in \tau \), and the Taylor coefficient \( a_k(x, y_\tau) \) with \( |k| = \lambda \) is estimated by

\[
|a_k(x, y_\tau)| \leq \frac{\lambda!}{k!} |R^2 - x \cdot y_\tau|^{-\lambda-1} R^{\lambda+1} \leq \frac{\lambda!}{k!} \rho^{-\lambda-1}(\tau) h^{\rho(\lambda+1)} R^{\lambda+1}.
\]

Now, since \( \sum_{|k| = \lambda} \frac{\lambda!}{k!} = 3^\lambda \), \( \rho(\tau) \geq \frac{\sqrt{7}}{2} h \) for arbitrary \( \tau \), and \( \sum_{\tau \in F(x)} \sum_{y_j \in \tau} = \sum_{\tau \in F(x)} n_\tau \leq N \), where \( n_\tau \) is the number of the point vortices contained in the
box \( \tau \in F(x) \), we obtain

\[
|u_N(x, t) - u_N^\lambda(x, t)| \leq \frac{CR^2}{N} \sum_{\tau \in F(x)} \sum_{y_j \in \tau} \left| \sum_{|k| = \lambda} a_k(x, y_\tau)|\rho^\lambda(\tau) \right|
\]

\[
\leq \frac{CR^{\lambda+3}}{N} \sum_{\tau \in F(x)} \sum_{y_j \in \tau} \sum_{k=\lambda}^{\lambda+1} \frac{\lambda!}{k!} h^{\nu(\lambda+1)}
\]

\[
\leq \frac{CR^{\lambda+3}}{N} \sum_{\tau \in F(x)} \sum_{y_j \in \tau} h^{\nu(\lambda+1)}
\]

\[
\leq C' R^{\lambda+3} h^{\nu(\lambda+1)-1} \leq C'' h^{\nu-1},
\]  

for some constants \( C' \) and \( C'' \). Hence, for \( \nu = O(3/\lambda) \), we finally have

\[
|u_N(x, t) - u_N^\lambda(x, t)| \leq C'' h^2,
\]

which is the same error estimate as the fast algorithm for the planar case[4]. In the actual numerical computation, the parameter \( \nu \) is chosen so that the efficiency of the fast algorithm is balanced with its accuracy.

3 Numerical tests

3.1 Efficiency of the algorithm

We check the efficiency of the fast algorithm. As a test configuration of the point vortices, we consider a layered configuration of \( M \) vortex sheets, which are defined as one-dimensional discontinuous curves of the velocity field, on the sphere with radius \( R = 0.5 \). Suppose that they are located along the latitudes

\[
z^{(i)} = R - \frac{i}{M + 1}, \quad \text{for} \quad i = 1, \ldots, M.
\]

Then the positions of the vortex sheets are specified by

\[
x^{(i)}(\alpha) = \left( \sqrt{R^2 - (z^{(i)})^2} \cos 2\pi \alpha, \sqrt{R^2 - (z^{(i)})^2} \sin 2\pi \alpha, z^{(i)} \right),
\]

for \( 0 \leq \alpha < 1 \). Discretizing each vortex sheet by \( N \) points, we obtain the \( MN \) point vortices, to which we apply the fast algorithm. The layered configuration is suitable for the numerical test to the fast algorithm, since the discretizing point vortices spread almost uniformly on the whole sphere.

In the fast algorithm, there are three parameters \( \nu, \lambda \) and \( l \), which are the parameter in the far-field condition (5), the approximation order of the Taylor expansion, and the level of the tree structure of the computational boxes respectively. Here, they are determined by \( \nu = \frac{1}{n} \) and \( l = n \) from the number of point
Table 1: $L^2$ relative error $E_N^{(2)}$ and the maximum relative error $E_N^{(\infty)}$ between the fast algorithm and the direct summation for the layered vortex-sheets configuration (16).

<table>
<thead>
<tr>
<th>MN</th>
<th>$\lambda = 4$</th>
<th>$\lambda = 6$</th>
<th>$\lambda = 8$</th>
<th>$\lambda = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>3.25e-4</td>
<td>1.84e-5</td>
<td>1.18e-6</td>
<td>8.50e-8</td>
</tr>
<tr>
<td>16384</td>
<td>4.31e-4</td>
<td>2.37e-5</td>
<td>1.49e-6</td>
<td>1.07e-8</td>
</tr>
<tr>
<td>65536</td>
<td>4.51e-4</td>
<td>2.50e-5</td>
<td>1.58e-6</td>
<td>1.15e-7</td>
</tr>
<tr>
<td>262144</td>
<td>4.74e-4</td>
<td>2.65e-5</td>
<td>1.72e-6</td>
<td>1.27e-7</td>
</tr>
<tr>
<td>1048576</td>
<td>4.89e-4</td>
<td>2.74e-5</td>
<td>1.80e-6</td>
<td>1.36e-7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MN</th>
<th>$\lambda = 4$</th>
<th>$\lambda = 6$</th>
<th>$\lambda = 8$</th>
<th>$\lambda = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>4.23e-4</td>
<td>2.32e-5</td>
<td>1.35e-6</td>
<td>8.35e-8</td>
</tr>
<tr>
<td>16384</td>
<td>6.91e-4</td>
<td>3.60e-5</td>
<td>2.09e-6</td>
<td>1.35e-8</td>
</tr>
<tr>
<td>65536</td>
<td>7.78e-4</td>
<td>4.07e-5</td>
<td>2.35e-6</td>
<td>1.48e-7</td>
</tr>
<tr>
<td>262144</td>
<td>8.05e-4</td>
<td>4.17e-5</td>
<td>2.41e-6</td>
<td>1.53e-7</td>
</tr>
<tr>
<td>1048576</td>
<td>8.17e-4</td>
<td>4.20e-5</td>
<td>2.41e-6</td>
<td>1.53e-7</td>
</tr>
</tbody>
</table>

Table 1: $L^2$ relative error $E_N^{(2)}$ and the maximum relative error $E_N^{(\infty)}$ between the fast algorithm and the direct summation for the layered vortex-sheets configuration (16).

We change the approximation order $\lambda$ from 4 to 10 and compute the velocity fields for the layered configuration. Then we compute the $L^2$ relative error $E_N^{(2)}$ and the maximum relative error $E_N^{(\infty)}$ between the velocity field evaluated by the fast tree-code algorithm $u_N^\lambda(x_i)$ and that by the direct summation $u_N(x_i)$, which are defined by

$$E_N^{(2)} = \left( \frac{\sum_{i=1}^N |u_N(x_i) - u_N^\lambda(x_i)|^2}{\left( \sum_{i=1}^N |u_N(x_i)|^2 \right)^{1/2}} \right)^{1/2}, \quad E_N^{(\infty)} = \frac{\max_{1 \leq i \leq N} |u_N(x_i) - u_N^\lambda(x_i)|}{\max_{1 \leq i \leq N} |u_N(x_i)|},$$

for a given configuration of the point vortices $x_i$, $i = 1, \ldots, N$. Table 1 shows the $L^2$ relative error and the maximum relative error for various $MN$ and $\lambda$. The numerical computation is carried out by Opteron 275 processor with 6GB memory. The table indicates that the relative errors decrease as $\lambda$ increases for fixed $MN$. On the other hand, the errors stay in the same order for fixed $\lambda$. Table 2 shows that the computational time of the fast algorithm gets much faster than that of the direct summation for larger $MN$, though the fast algorithm is not effective in the computational time for smaller $MN$. In order to check the accuracy of the fast algorithm given theoretically in the previous section, we rearrange the computational results in Table 3 so that $\lambda\nu$ keeps constant. Both of the error decreases as $O(h^2) = O(N)$, which supports the theoretical estimate (15).
Table 2: Computational time in seconds for the evaluation of the velocity fields for the layer configuration (16).

<table>
<thead>
<tr>
<th>(N)</th>
<th>(\lambda = 4)</th>
<th>(\lambda = 6)</th>
<th>(\lambda = 8)</th>
<th>(\lambda = 10)</th>
<th>direct</th>
<th>(\nu)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>0.7</td>
<td>1.6</td>
<td>3.1</td>
<td>6.5</td>
<td>0.6</td>
<td>(\nu = 0.083)</td>
</tr>
<tr>
<td>16384</td>
<td>4.9</td>
<td>11.9</td>
<td>20.3</td>
<td>34.7</td>
<td>10.2</td>
<td>(\nu = 0.071)</td>
</tr>
<tr>
<td>65536</td>
<td>40.3</td>
<td>71.8</td>
<td>98.0</td>
<td>156.2</td>
<td>164.3</td>
<td>(\nu = 0.0625)</td>
</tr>
<tr>
<td>262144</td>
<td>310.5</td>
<td>433.2</td>
<td>576.5</td>
<td>908.8</td>
<td>2629.3</td>
<td>(\nu = 0.055)</td>
</tr>
<tr>
<td>1048576</td>
<td>2034.4</td>
<td>2694.5</td>
<td>4018.2</td>
<td>6050.5</td>
<td>42060.5</td>
<td>(\nu = 0.05)</td>
</tr>
</tbody>
</table>

Table 3: Error estimate for the evaluation of the velocity field for the layer configuration (16).

<table>
<thead>
<tr>
<th>(N)</th>
<th>4096</th>
<th>65536</th>
<th>1048576</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda)</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>(\nu)</td>
<td>0.083</td>
<td>0.0625</td>
<td>0.05</td>
</tr>
<tr>
<td>(E_N^{(2)})</td>
<td>1.84e-5</td>
<td>1.58e-6</td>
<td>1.34e-7</td>
</tr>
<tr>
<td>(E_N^{(\infty)})</td>
<td>2.32e-5</td>
<td>2.35e-6</td>
<td>1.53e-7</td>
</tr>
</tbody>
</table>

3.2 Long time evolution of two vortex sheets

As another test problem, we compute the long-time evolution of two vortex sheets on the sphere. Since the vorticity exists only in the vortex sheets, we just discretize each of them with the point vortices and track their evolution numerically with the vortex blob method.

We assume that the two vortex sheets lay along the line of latitude \(\theta_1\) and \(\theta_2\). The initial position of the \(i\)-th vortex sheet on the sphere of radius \(R = 0.5\) is given by

\[
\begin{align*}
  z^{(i)}(\alpha) &= R(\cos \theta_i + \epsilon \sin 2\pi \alpha), \\
  x^{(i)}(\alpha) &= \sqrt{R^2 - (z^{(i)})^2} \cos 2\pi \alpha, \\
  y^{(i)}(\alpha) &= \sqrt{R^2 - (z^{(i)})^2} \sin 2\pi \alpha.
\end{align*}
\]

The parameter \(\alpha \in [0, 1]\) is a Lagrangian variable moving with the fluid particle. Small perturbation is imposed to \(z^{(i)}\) in (17), since the two vortex sheets rotate in the longitudinal direction with some constant speeds without the perturbation and thus we observe no complex interaction between them. The amplitude of the disturbance is \(\epsilon = 0.02\). Then we discretize the segment \([0, 1]\) by \(N\) points to obtain the \(N\) point vortices. Since the strengths of the vortex sheets are assumed to be uniform, the strengths of the discretizing point vortices are identical. Accordingly, the initial positions of the point vortices are given by \(x_m^{(i)} \approx x^{(i)}(\frac{m}{N})\) with
the same strength \( \Gamma_m^{(i)} = 1/N \equiv \Gamma \) for \( i = 1, 2 \) and \( m = 1, 2, \ldots, N \). With the initial data, we solve numerically the following \( 2N \)-dimensional ordinary differential equations for \( m = 1, \ldots, N \).

\[
\frac{d\mathbf{x}_m^{(1)}}{dt} = -\frac{\Gamma}{4\pi R} \sum_{j \neq m}^N \frac{\mathbf{x}_m^{(1)} \times \mathbf{x}_j^{(1)}}{R^2 + \sigma^2 - \mathbf{x}_m^{(1)} \cdot \mathbf{x}_j^{(1)}} - \frac{\Gamma}{4\pi R} \sum_{j=1}^N \frac{\mathbf{x}_m^{(1)} \times \mathbf{x}_j^{(2)}}{R^2 + \sigma^2 - \mathbf{x}_m^{(1)} \cdot \mathbf{x}_j^{(2)}},
\]

\[
\frac{d\mathbf{x}_m^{(2)}}{dt} = -\frac{\Gamma}{4\pi R} \sum_{j \neq m}^N \frac{\mathbf{x}_m^{(2)} \times \mathbf{x}_j^{(1)}}{R^2 + \sigma^2 - \mathbf{x}_m^{(2)} \cdot \mathbf{x}_j^{(1)}} - \frac{\Gamma}{4\pi R} \sum_{j=1}^N \frac{\mathbf{x}_m^{(2)} \times \mathbf{x}_j^{(1)}}{R^2 + \sigma^2 - \mathbf{x}_m^{(2)} \cdot \mathbf{x}_j^{(1)}}.
\]

The first summation in the right hand side of the equation represents the self-induced velocity, while the second one is the interaction between the two vortex sheets. The temporal integration is carried out by the fourth order Runge-Kutta method, whose time step size is taken as \( \Delta t = 0.05 \). Each of the vortex sheets is discretized by \( N = 131072 \), i.e., we use the total of 262144 point vortices, for which we set the parameters of the fast algorithm as \( \nu = 0.055 \) and \( \lambda = 10 \). The regularization parameter of the vortex blob method is set to \( \sigma = 0.1 \).

Before showing numerical results, let us remark a numerical difficulty particularly arises in the numerical computation of vortex sheets. According to the linear stability analysis of a single vortex sheet on the sphere[21], a small perturbation grows very rapidly due to the Kelvin-Helmholtz instability and thus the evolution of the vortex sheet defines an ill-posed problem in the sense of Hadamard. Hence, the accumulation of the round-off error inevitably deteriorates the accuracy of the numerical computation. What makes the matter worse, since the fast algorithm gives rise to not only the round off error but also the approximation error, these errors affect the computational result seriously. In order to keep the accuracy of the numerical computation, we adopt the Fourier filtering technique. As for the detailed description of the technique, we would like the readers to refer to [14]. The threshold of the filter is set to \( 1.0 \times 10^{-9} \).

Figure 1 shows the evolution of two vortex sheets up to \( t = 26 \). The base latitudes of the two vortex sheets are given by \( \theta_1 = 0.3\pi \) and \( \theta_2 = 0.5\pi \). Since the two vortex sheets are well separated compared to the amplitude of the initial disturbance \( \epsilon = 0.02 \), each of the vortex sheets becomes unstable independently due to the Kelvin-Helmholtz instability before they interact with each other. Since the linear stability analysis of the single vortex sheet[21] shows that the latitudinal vortex sheets near the pole are more linearly unstable than those near the equator, the upper vortex sheet in the northern hemisphere gets unstable faster than that near the equator. Then they evolve into structures with many rolling-up spirals and very thin filaments. After that, the two vortex sheets interact strongly to show a complex pattern.

Another computational result is shown in Figure 2. The initial configuration of the two vortex sheet is given by (17) with \( \theta_1 = \pi/3 - \pi/20 \) and \( \theta_2 = \pi/3 + \pi/20 \) and \( \epsilon = 0.02 \). After the northern vortex sheet rolls up due to the Kelvin-
Helmholtz instability immediately, they begin interacting with each other before the southern vortex sheet rolls up, since the initial distance between them is closer than the precious result. As a result of the complex interaction, the two vortex sheets gather in the northern polar region and a big coherent spiral structure is split out of the region and stays around the equator.

Both numerical results with $N = 262144$ point vortices keep their fine resolution up to $t = 26$. It is quite difficult to compute the evolution for such a long time without the fast tree-code algorithm. Let us finally mention how the regularization parameter affects the numerical results. As was shown in [14], in some range of $\sigma$, the numerical result is qualitatively similar except that the spirals tend to have infinitely many windings as $\sigma \to 0$. On the other hand, for excessively small $\sigma$, the numerical computation loses its accuracy. In order to keep the accuracy for smaller $\delta$, a larger number of the point vortices should be used.

4 Summary and discussion

A fast tree-code algorithm to evaluate the interaction between the $N$ point vortices on the sphere has been proposed. The fast method is applicable to the numerical computation for the inviscid and incompressible flows on the sphere, since the motion of the $N$ point vortices is naturally derived when the Euler equations are approximated by the vortex method.

The tree-code algorithm is an extension of the fast algorithm originally developed by Draghicescu[4] for the two-dimensional and the three-dimensional Euler equations. However, a naive implementation of the algorithm to the $N$ vortex motion on the sphere is practically ineffective, since the high-order Taylor coefficients of the velocity field in the equations are expressed in a too complicated way to evaluate efficiently. We resolve the difficulty by rewriting the equations of the $N$ point vortices in the three-dimensional space, for which the Taylor coefficients are easily computed with a simple formula. Owing to its three-dimensional representation of the equations, the fast algorithm works very effective. The error estimate shows that the approximation error decreases as $O(1/N)$ with an appropriate choice of the parameters in the algorithm. The computational cost is largely reduced to $O(N(\log N)^4)$, which is the same order as the three-dimensional fast algorithm[4].

The efficiency of the fast method is examined by applying it to the numerical computation of vortex sheets. The theoretical error estimate and the reduction of computational cost are clearly confirmed. We also compute the long-time evolution of the two vortex sheets, which show very complex interactions between them.

The algorithm is formulated for the three-dimensional representation of the velocity field (4), which depends on nothing but the relative distances between
Figure 1: Long time evolution of two vortex sheets for $\theta_1 = 0.3\pi$ and $\theta_2 = 0.5\pi$ up to $t = 26$. Each vortex sheet is approximated by the 131072 point vortices.
Figure 2: Long time evolution of two vortex sheets up to $t = 26$. The initial base latitudes for them are given by $\theta_1 = \pi/3 - \pi/20$ and $\theta_2 = \pi/3 + \pi/20$ respectively.
the point vortices. Therefore, it is unnecessary to take into consideration the singular pole effect, which sometimes gives rise to serious problems in the numerical simulation on the sphere. On the contrary, because of the three-dimensional formulation, the point vortices might detach from the sphere, if the accuracy of the computation is too poor. In the numerical tests given in the paper, we check if the point vortices stay in the surface of the sphere every time steps.

Let us finally remark possible future directions. In the present article, the maximum number of the point vortices we used is less than a million, since the fast algorithm was carried out on a comparatively inexpensive processor with a few working memory. However, in fact, as was confirmed in the numerical computation of the three-dimensional vortex sheets[20], the fast tree-code algorithm is easily adjustable to a parallel processor. Hence, implementation of the fast algorithm to fast parallel processors enables us to use more than ten million point vortices. Thus the fast algorithm gives us a new way to challenge many interesting problems such as the statistical theory of the point vortices on the sphere and numerical simulations of the atmospheric phenomena on Earth.

Acknowledgments

This work is partially supported by Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (A) #17684002 2007, Grant-in-Aid for Exploratory # 17654018 2007, and Grant-in-Aid for formation of COE at Hokkaido University.

References


