Internship Report

Efficient Preconditioning of DC PSE Operators based on Multivariate Newton Interpolation

Student: Xinjing Jiang *
Supervisor at external research institution:
Prof. Dr. sc. techn. Ivo F. Sbalzarini †
Dr. rer. nat. Michael Hecht ‡
Chair of Scientific Computing for Systems Biology
Faculty of Computer Science, TU Dresden & Center for Systems Biology Dresden,
Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany

June 2019

Abstract

Discretisation-Corrected Particle Strength Exchange (DC PSE) is a mesh–free particle method that numerically solves PDEs & ODEs. Thereby, the differential operator is discretised with respect to certain polynomial kernels ensuring an approximative solution up to specified $r$-th moment $r \in \mathbb{N}$. In this internship we have developed an MATLAB prototype that preconditions the polynomial interpolation problem, whose solution defines the polynomial kernels, in the 2D case. Potentially, this method improves both: runtime and accuracy performances of DC PSE schemes.

1 Introduction

A core application of numerical analysis is the approximation of the solution of Ordinary Differential Equations (ODE) or Partial Differential Equations (PDE). This always involves a (temporal and/or spatial) discretisation scheme and a solver for the resulting equations. There are three classes of methods: spectral methods, Galerkin schemes and collocation schemes.

---

*xinjing.jiang@mailbox.tu-dresden.de
†sbalzarini@mpi-cbg.de, https://mosaic.mpi-cbg.de
‡hecht@mpi-cbg.de, https://mosaic.mpi-cbg.de
Spectral methods are based on Fourier transforms. Since FFTs are only efficient on regular Cartesian grids, spectral methods are hard to apply in complex geometries and on adaptive-resolution discretisation. Galerkin schemes are based on expanding the solution of the differential equation in some basis functions and rely on choosing a suitable basis for given data sets.

Collocation methods can generally be understood as formulations of finite-difference schemes and mesh-free collocation methods [10] as DC PSE schemes. The inversion of the Vandermonde matrix implied in mesh-free methods is the bottleneck and main challenge of these schemes. Despite that issue the DC PSE scheme can handle complex geometries with boundaries in a feasible way. This report aims to give a baseline for developing a preconditioning method for mesh-free particle methods. So far preconditioning methods were not used for mesh-free particle methods. This is due to two reasons: Firstly, general preconditioning methods are much too costly in terms of runtime performance. Secondly, specific methods for this special situation were not developed so far due to the absence of feasible classifications of unisolvent nodes required to solve multivariate polynomial interpolation problems. Here, we use the results of [7], where such a required classification could be given. Our main ideas on how to adapt the provided notion of unisolvent nodes, preliminary validation of numerical experiments and a short survey on DC PSE schemes are presented in the report.

1.1 Outline
To set the stage, we first follow [2] to summarise the essential notions of DC PSE schemes. In section 2 we then briefly review the multivariate Newton interpolation, which is the basis of our method. Then, we introduce our detection approach and present some numerical experiments in section 3 and 4 respectively. Finally, we will discuss the strengths and weaknesses of our approach and give an outlook of further developments.

1.2 DC PSE Operator
DC PSE is a correction of the Particle Strength Exchange method (PSE) on irregular particle distributions. We follow Bourantas et al. [2] and sketch the main facts.

Here we restrict to the 2D case and consider a sufficiently smooth field $f : \Omega \subseteq R^2 \rightarrow R$ on a regular domain $\Omega$. To discretise the spatial differential operator

$$D^{m,n}f(x_p) = \frac{\partial^{m+n}}{\partial x^m \partial y^n} f(x,y) \bigg|_{x=x_p,y=y_p}$$

of arbitrary order $m,n \in \mathbb{N}$ at a particular collocation point $x_p = (x_p, y_p) \in \Omega$. The inversion of the Vandermonde matrix implied in mesh-free methods is the bottleneck and main challenge of these schemes. Despite that issue the DC PSE scheme can handle complex geometries with boundaries in a feasible way. This report aims to give a baseline for developing a preconditioning method for mesh-free particle methods. So far preconditioning methods were not used for mesh-free particle methods. This is due to two reasons: Firstly, general preconditioning methods are much too costly in terms of runtime performance. Secondly, specific methods for this special situation were not developed so far due to the absence of feasible classifications of unisolvent nodes required to solve multivariate polynomial interpolation problems. Here, we use the results of [7], where such a required classification could be given. Our main ideas on how to adapt the provided notion of unisolvent nodes, preliminary validation of numerical experiments and a short survey on DC PSE schemes are presented in the report.

1.1 Outline
To set the stage, we first follow [2] to summarise the essential notions of DC PSE schemes. In section 2 we then briefly review the multivariate Newton interpolation, which is the basis of our method. Then, we introduce our detection approach and present some numerical experiments in section 3 and 4 respectively. Finally, we will discuss the strengths and weaknesses of our approach and give an outlook of further developments.

1.2 DC PSE Operator
DC PSE is a correction of the Particle Strength Exchange method (PSE) on irregular particle distributions. We follow Bourantas et al. [2] and sketch the main facts.

Here we restrict to the 2D case and consider a sufficiently smooth field $f : \Omega \subseteq R^2 \rightarrow R$ on a regular domain $\Omega$. To discretise the spatial differential operator

$$D^{m,n}f(x_p) = \frac{\partial^{m+n}}{\partial x^m \partial y^n} f(x,y) \bigg|_{x=x_p,y=y_p}$$

of arbitrary order $m,n \in \mathbb{N}$ at a particular collocation point $x_p = (x_p, y_p) \in \Omega$. The inversion of the Vandermonde matrix implied in mesh-free methods is the bottleneck and main challenge of these schemes. Despite that issue the DC PSE scheme can handle complex geometries with boundaries in a feasible way. This report aims to give a baseline for developing a preconditioning method for mesh-free particle methods. So far preconditioning methods were not used for mesh-free particle methods. This is due to two reasons: Firstly, general preconditioning methods are much too costly in terms of runtime performance. Secondly, specific methods for this special situation were not developed so far due to the absence of feasible classifications of unisolvent nodes required to solve multivariate polynomial interpolation problems. Here, we use the results of [7], where such a required classification could be given. Our main ideas on how to adapt the provided notion of unisolvent nodes, preliminary validation of numerical experiments and a short survey on DC PSE schemes are presented in the report.
Ω, we assign its corresponding DC PSE operator as
\[ Q_{m,n}^f(x_p) = \frac{1}{\epsilon(x_p)^{m+n}} \sum_{x_q \in N(x_p)} (f(x_q) \pm f(x_p)) \eta(\frac{x_q - x_p}{\epsilon(x_p)}) , \]
where \( \epsilon(x) \) is a spatially dependent scaling function, \( \eta(x, \epsilon(x)) \) a kernel function, and \( N(x_p) \) the set of nodes in the supporting neighbourhood of the kernel function \( \eta \).

By decreasing the average spacing between nodes, \( h(x_p) \rightarrow 0 \), the DC PSE operator shall converge to the spatial derivative with an asymptotic rate \( r \in \mathbb{N} \), i.e.
\[ Q_{m,n}^f(x_p) = D_{m,n}^f(x_p) + O(h^r) . \] (1)

In [2] \( \eta \) is assumed to be a polynomial kernel of the form
\[ \eta(x) = \begin{cases} \sum_{i+j \leq r+m+n} a_{i,j} x^i y^j e^{-x^2 - y^2} & \sqrt{x^2 + y^2} < r_c \\ 0 & \text{otherwise} \end{cases} \] (2)
with some scalars \( a_{i,j} \) and the cut-off radius \( r_c \), which in general defines the kernel support and was set implicitly by using the nearest \( l - 1 \) neighbours of each node. An additional assumption is that the scaling parameter \( \eta(x_p) \) converges at the same rate as the average spacing between nodes \( h(x_p) \), i.e.
\[ \frac{h(x_p)}{\eta(x_p)} \in \mathcal{O}(1). \] (3)

Thereby, the DC PSE operator can be rewritten as
\[ Q_{m,n}^f(x_p) = \sum_{x_q \in N(x_p)} f(x_q) \pm f(x_p) \mathbf{p}(x_p - x_q) \frac{\epsilon(x_p)^{m+n}}{\epsilon(x_p)^{m+n}} D_{i,j} f(x_p) Z_{i,j}(x_p) \]
\[ \times \mathbf{a}^T(x_p) e^{-\frac{(x_p - x_q)^2}{\epsilon(x_p)^2} - \frac{(y_p - y_q)^2}{\epsilon(x_p)^2}} , \]
where \( \mathbf{p}(x) = \{p_1(x), p_2(x), ..., p_l(x)\} \) and \( \mathbf{a}(x) \) are vectors of the terms in the monomial basis and of their coefficients in [2].

To determine these coefficients, we firstly apply Taylor’s theorem with respect to [1] around the node \( x_p \) and can get the following substitution:
\[ Q_{m,n}^f(x_p) = \left( \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \frac{\epsilon(x_p)^{i+j-m-n}(-1)^{i+j}}{i! j!} D_{i,j} f(x_p) Z_{i,j}(x_p) \right) \]
\[ \pm Z^{0,0}(x_p) \epsilon(x_p)^{-m-n} f(x_p) , \] (4)
where
\[ Z_{i,j}(x_p) = \sum_{x_q \in N(x_p)} \frac{(x_p - x_q)^i(y_p - y_q)^j}{\epsilon(x_p)^{i+j}} \eta(\frac{x_p - x_q}{\epsilon(x_p)}) \]

3
are the discrete moments of \( \eta \).

Recall the assumption in (3), the convergence rate \( r \) of the DC PSE operator \( Q^{m,n} \) is thus solely determined by the coefficients of the terms \( \epsilon(x_p)^{i+j-m-n} \). As the coefficients in (1) are required to be 1 when \( i = m \) and \( j = n \), and 0 when \( i+j - m - n < r \), these moment conditions arise:

\[
Z^{i,j}(x_p) = \begin{cases}
  i!j!(1)^{i+j} & i = m, j = n \\
  0 & \alpha_{\min} < i + j < r + m + n \\
  \infty & \text{otherwise}
\end{cases}
\]

In summary, the following linear equation system of kernel coefficients has to be solved:

\[
b = (-1)^{m+n}D^{m,n}p(x)|_{x=0} \in \mathbb{R}^{1 \times 1}
\]

\[
A(x_p)A^T(x_p) = b.
\]

(5)

\( A \) and \( B \) can be further decomposed into

\[
A(x_p) = B(x_p)^T \in \mathbb{R}^{l \times l}
\]

\[
B(x_p) = E(x_p)^T \in \mathbb{R}^{k \times l},
\]

where \( V(x_p) \) is the Vandermonde matrix with respect to the nodes \( z_i \in M(x_p) \subseteq N(x_p) \), \( i = 1, \ldots, k \),

\[
V(x_p) = \begin{pmatrix}
p_1(z_1(x_p)) & p_2(z_1(x_p)) & \cdots & p_l(z_1(x_p)) \\
p_1(z_2(x_p)) & p_2(z_2(x_p)) & \cdots & p_l(z_2(x_p)) \\
\vdots & \vdots & \ddots & \vdots \\
p_1(z_k(x_p)) & p_2(z_k(x_p)) & \cdots & p_l(z_k(x_p))
\end{pmatrix} \in \mathbb{R}^{k \times l}
\]

(6)

and \( E(x_p) \) is a diagonal matrix

\[
E(x_p) = \text{diag} \left( \left\{ e^{-\epsilon(x_p)^2} \right\}^{k}_{q=1} \right) \in \mathbb{R}^{k \times k}.
\]

If one uses this approach to numerically solve ODE’s & PDE’s, the differential operator \( D^{m,n} \) has to be discretised by the DC PSE operator \( Q^{m,n} \), \( m,n \in N \) for every node \( x_p \). Thus, in order to implement a well performing numerical solver we have to provide that the Vandermonde matrix \( V(x_p) \) can be inverted efficiently and numerically accurate for every particle. The problem we thereby address can be stated as follows.

**Problem 1.** Let \( x_p \) be a node and \( N(x_p) \) be neighbourhood of \( x_p \). Find a set of nodes \( M(x_p) \subseteq N(x_p) \) with minimal cardinality such that the Vandermonde matrix \( V(x_p) \) with respect to \( M(x_p) \) is well conditioned.
Thereby, we consider the Vandermonde matrix $V(x_p)$ in [6] to be well conditioned if its inverse is computable within machine accuracy by classical matrix inversion schemes [5][9][12].

1.3 State of the art solutions

The decomposition of $A(x_p)$ in [5] allows to consider a pseudo inverse approach. That is to consider more particles than actually required, i.e., $B(x_p) \in \mathbb{R}^{k \times l}$ with $k \gg l$. The larger $k$ the more likely it is for $B(x_p)$ to be regular or well conditioned. Hence, the pseudo inverse $B^+$ with $B^+B = I$ can be computed accurately.

However, the larger $k$ becomes the larger becomes the runtime costs for computing $B^+ [3]$. Therefore, our aim is to formulate a preconditioning algorithm that chooses only $l$ or $l+1, l+2, l+3$ neighbouring nodes such that the Vandermonde matrix $V(x_p)$ is well conditioned. Since the interpolation problem has to be solved for every particle and their numbers can reach from $10^4 - 10^7$ this effect can make a crucial difference. In fact, classical preconditioning methods were not used currently due to their high runtime costs.

Our approach is based on the characterisation of unisolvent interpolation nodes from [7], which we shortly recapitulate in the next section.

2 The Multivariate Polynomial Interpolation Problem (PIP)

In [7] the general the multivariate polynomial interpolation problem is stated as follows.

**Problem 2 (PIP).** Let $m, n \in \mathbb{N}$ and $f : \mathbb{R}^m \rightarrow \mathbb{R}$ be a computable function.

i) Choose $N(m, n)$ nodes $P_{m,n} = \{p_1, \ldots, p_{N(m,n)}\} \subseteq \mathbb{R}^m$ such that $P_{m,n}$ is unisolvent, i.e., for every $f$ there is exactly one polynomial $Q_{m,n,f} \in \Pi_{m,n}$ fitting $f$ on $P_{m,n}$ as $Q_{m,n,f}(p) = f(p)$ for all $p \in P_{m,n}$.

ii) Determine $Q_{m,n,f}$ once a unisolvent node set $P_{m,n}$ has been chosen.

Here, $\Pi_{m,n}$ is the vector space of polynomials $Q \in \mathbb{R}[x_1, \ldots, x_m]$ in $m$ variables of degree $\text{deg}(Q) \leq n$. Every $Q \in \Pi_{m,n}$ has $N(m, n) := \binom{m+n}{n}$ monomials/coefficients.

In 1D a unisolvent nodes set is simply given by $n + 1$ distinct points $p_0, \ldots, p_n \in \mathbb{R}$. Therefore, the one-dimensional PIP ($m = 1$) can be solved efficiently in $O(N(1,n)^2) = O(n^2)$ and numerically accurately by various algorithms based on Newton or Lagrange Interpolation [1][4][8][11]. The main contribution of [7] is that both; The well performing Newton interpolation
scheme and the approximation results known in dimension one could be generalised to arbitrary dimensions.

**Theorem 1** (Hecht et al.). Let \( m, n \in \mathbb{N} \) and \( f : \mathbb{R}^m \rightarrow \mathbb{R} \) be a given function. Then, there exists an algorithm with runtime complexity \( \mathcal{O}(N(m,n)^2) \) requiring \( \mathcal{O}(N(m,n)) \) memory that computes:

i) A unisolvent node set \( P_{m,n} \subseteq \mathbb{R}^m \) and the coefficients of the corresponding interpolation polynomial \( Q_{m,n,f} \in \Pi_{m,n} \) in normal form.

ii) A unisolvent set \( P_{m,n} \subseteq \mathbb{R}^m \) of multidimensional Newton nodes and the coefficients of the corresponding interpolation polynomial \( Q_{m,n,f} \in \Pi_{m,n} \) in multivariate Newton form.

The notion of \( mD \)–Newton nodes is given by choosing \( m \) sets \( P_k \) of cardinality \( \#P_k = n + 1, k = 1,\ldots,m \) and generating \( N(m,n) \) nodes \( P_{m,n} \subseteq \bigoplus_{k=1}^{m} P_k \subseteq \Omega \) as a sparse \( T \)-grid contained in the grid \( \bigoplus_{k=1}^{m} P_k \) spanned by the \( P_k \). The construction is based on the tree \( T_{m,n} \), which is sketched in Figure 1 and is explained in [7] in detail. Further, the tree \( T_{m,n} \) and the \( mD \)–Newton-nodes are used to formulate a \( mD \)–divided difference scheme yielding the solutions of the PIP in the runtime and storage complexity stated above.

As it turns out \( mD \)–Newton nodes yield a well conditioned \( mD \)–Vandermonde matrix.

### 2.1 Unisolvent Interpolation nodes in dimension 2

In the special case of dimension 2 the construction of 2D unisolvent nodes can be done as follows.

Choose \( n + 1 \) lines...\( L_i, i = 0,\ldots,n \) and \( i + 1 \) distinct nodes \( p_{i,k} \subseteq L_i \), \( k = 0,\ldots,i \) on each line such that no node \( p_{i,j} \) is contained in more than one
For simplifying the construction we assume that all lines are parallel and arranged such that the nodes $P$ can be chosen from a regular grid $G$. In the special case where the nodes are nested with respect to the line $L_i$ the nodes are called 2D-Newton Nodes. An example for $n=4$ is illustrated in Figure 2 where 2a shows 2D Unisolvent nodes and 2b the 2D Newton nodes.

We will use this fact to derive several templates of well conditioned node patterns in the special case of dimension 2. A set of neighbours $M(x_p) \subseteq N(x_p)$ of a particle $p$ is then chosen if $M(x_p)$ is close to one of these templates.

3 Our Approach

We have developed a fast template detection scheme that solves Problem 1. In order to assert this scheme we first introduce our notions of templates.

3.1 Interpolation Node Templates

Let us first consider the following 2 types of templates:

**Definition 1 (Strong and Relaxed Template).** For $m = 2$ and $n \in \mathbb{N}$ we call a set of nodes $\mathcal{M}_T \subseteq [-1, 1]^2$ a strong template if and only if $\mathcal{M}_T$ are equidistantly distributed 2D-Newton nodes and relaxed template if and only if $\mathcal{M}_T$ is a unisolvent node set with respect to parallel and equidistantly distributed lines as described in section 2.1.

Definition 1 formalises the desired spatial distribution of the nearby particles. However, in practice we do not want to move the particles and are therefore only allowed to pick some of the particles in the local supporting domain of the kernel. In the next section we assert how our detection or recognition scheme of such a feasible set of particles is implemented.
3.2 Occupation matrix

The starting point is to find a better expression of the individual position of actual particles than by their coordinates. Then a fast evaluation of their geometry yields an efficient comparison with all allowed templates. Consequently, we can identify whether a template is already included or not. In the latter case, the mismatch with the templates could be fixed in 2 ways: enlarge the neighbourhood or choose some suboptimal nodes as a compromise.

3.2.1 Occupation matrix as flag

We first split the domain into $g \times g$ sub domains, where $g = n + 1$, w.r.t. degree $n$. For given particle $p$ we center the neighbourhood $\mathcal{N}(x_p)$ at 0, i.e., we assume $x_p = (0, 0) \in \mathbb{R}^2$ and consider the rounding function

$$R(x, y) = ([g \times (x + 1)/2], [g \times (1 - y)/2]).$$

Additional we define the occupation matrix as follows.

**Definition 2** (Occupation matrix). Let $\mathcal{N}(x_p)$ be a neighbourhood of $x_p$. Then we call the matrix $O = (o_{ij})_{1 \leq i, j \leq n}$ with

$$o_{ij} = \begin{cases} 1, & \text{if there is a particle } q \in \mathcal{N}(x_p) \text{ with } R(x_q) = (i, j) \\ 0, & \text{otherwise} \end{cases}$$

the occupation matrix of $\mathcal{N}(x_p)$.

The occupation matrix is constructed as a flag: If one sub domain is occupied by at least one particle, then there will be a 1 at the corresponding position in $O$, otherwise 0.

3.2.2 Template Detection

Using the occupation matrix in Definition 2 we are now able to extract the spatial distribution of the nodes. In fact, for every unisolvent node set as in Figure 2a, there exists an reordering $\text{sort}(O) = (o_{k,j})_{1 \leq k, l \leq n}$ of
the occupation matrix $O$ such that the lower triangular of $\text{sort}(O)$ is fully occupied.

Following we will use the example domain illustrated in Fig. 3 to further explain our method. In this case the Occupation Matrix after reordering is $\text{sort}(O)$ which includes a lower triangular matrix:

\[
\text{sort}(O) = \begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix} \supseteq \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}.
\]

Thus, a strong template is included in $N(x_p)$ if and only if the condition $\text{sort}(O)_{ij} \geq 1$ for all $1 \leq j \leq i \leq n$ holds. In the case of relaxed template, we only have to consider the sorted vectors counting the numbers of occupied cells with respect to rows and lines, e.g. in the example here we will get

\[l(O) = (5, 5, 4, 3, 2) \quad \text{and} \quad r(O) = (5, 5, 4, 3, 2, 1) \in \mathbb{R}^{1 \times n}.\]

Hence, a relaxed template is included in $N(x_p)$ if and only if the condition

\[l_i(O) \geq i \quad \text{or} \quad r_i(O) \geq i \quad \forall i \quad (7)\]

holds. Thus, the template detection problem is boiled down to two cases. Either to sort the whole occupation matrix $O$ and compare it with the low triangular matrix for detecting strong templates, or to sort the amount of occupied sub domains in one direction and require (7) to be hold for the existence of a relaxed template.

4 Numerical Experiments

In order to illustrate our approach and demonstrate its performance in practice, we implement a prototype MATLAB (R2017b (9.3.0.713 579) version of the Preconditioning Scheme running on an Apple MacBook Pro (Retina, 15-inch, Mid 2012) with a 2.3GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 memory unser macOS High Sierra (version 10.13.6.). The following numerical experiments demonstrate the performance of our solver in comparison with the classical approaches.

4.1 Setup of numerical experiments

4.1.1 Four Approaches

According to our definition of templates, we implemented 2 MATLAB versions for detecting strong and relaxed templates, respectively. The third
<table>
<thead>
<tr>
<th>Nodes</th>
<th>Degree $n$</th>
<th>Method 1</th>
<th>Method 2</th>
<th>Method 3</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$</td>
<td>4</td>
<td>83%</td>
<td>67%</td>
<td>83%</td>
<td>57%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>56%</td>
<td>30%</td>
<td>43%</td>
<td>11%</td>
</tr>
<tr>
<td>$l+1$</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>88%</td>
<td>86%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>57%</td>
<td>33%</td>
</tr>
<tr>
<td>$l+2$</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>92%</td>
<td>91%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>68%</td>
<td>55%</td>
</tr>
<tr>
<td>$l+3$</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>94%</td>
<td>94%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>76%</td>
<td>72%</td>
</tr>
</tbody>
</table>

Table 1: success rate for method 1, 2 and 3

and fourth approach yielded by including the pseudo inverse method using relaxed templates or random selected nodes.

4.1.2 The supporting neighbour for the kernel

As we are still at a prototype state, instead of using a real particle data set, we uniformly choose random coordinates $x_q = (x, y)$ for the supposed particles $q$ located in the neighbourhood of particle $p$, i.e. $x_q \in \mathcal{N}(x_p)$. Enlarging the neighbourhood can thereby be simulated by increasing the number $|\mathcal{N}(x_p)|$.

4.2 Simulation Results

We ran each approach for 1,000,000 times to detect templates under varying configurations. Main parameters include degree and total amount of Nodes in the domain. The condition number of the Vandermonde Matrix is only calculated if a template has been detected or a set of nodes has been chosen.

The amount of supposed particle for the method 1 is $6l$, for other cases it remains $3l$, where $l$ is the amount of the nodes required to interpolate the DC PSE kernel. Due to the strict constraints Method 1 needs more particles than Method 2. For Method 1 and 2 the success threshold was set by requiring a condition number less than 5000, while we set the success threshold to 2000 for the other approaches.

4.2.1 Method 1 - Detection of Strong Templates

Figure 4 shows the simulation result comparing our first method with random selected nodes for degree $n = 4$ and 5. A clear improvement of the condition number can be observed here. 83% and 56% of the randomly generated instances were within our success bounds for $n = 4, 5$. In contrast, only 67% an 30% of the randomly chosen nodes could meet our success condition for degree $n = 4, 5$, respectively. For lower degrees we observed that the improvement we gain by our approach even increases. However, we decided to restrict a presentation here to the most relevant cases of degrees $n = 4, 5$. 

4.2.2 Method 2 - Detection of Relaxed Templates

Similar performances were shown in simulation of method 2. Here, a success quote of 93% and 71% for $n = 4, 5$ was reached, respectively, while random chosen nodes were well condition only in 75% or 36% of the considered instances.

4.2.3 Method 3 - Regularity of Pseudo Inverses

Method 3 is an extension of method 2, where the relaxed templates are desirable, but instead of calculating the condition number directly, we now take more nodes and compute the condition number of the corresponding pseudo inverse of $V(x_p)$.

As we already discussed in Chapter 1.3, the pseudo inverse approach is a common method to lower the condition number. However, this method usually takes more than twice as much nodes as actually needed. In our approach, we only take $a = \{1, 2, 3\}$ additional nodes.

Fig. 6 shows the improvement of the condition number. For degree
n = 4, the success quote improve from 83% to 88%, 92%, 94% by adding 1, 2, 3 nodes, respectively. Analogously, for n = 5 we obtain improvements from 43% to 57%, 68%, 76%, respectively.

In comparison with random selected node sets Fig. 7, 8 and 9 validate the obtained improvement despite the case n = 5, l + 3. However, by taking more nodes, their performances are getting similar (for instance in case of n=4 and taking 3 additional nodes, the success quote for both were almost 95%, see Fig. 9a). If the computational costs are also taken into account, the latter method would have a big advantage.

5 Conclusions and Outlook

We presented a MATLAB prototype detecting well conditioned nodes in the supporting neighbourhood of DC PSE operators in dimension 2. Thereby, the key element was to use the results of [6] to introduce 2 types of well conditioned templates. The construction of the occupation matrix $O$ enables to efficiently detect the templates by sorting $O$. Our experiments validate
that the thereby chosen nodes are significantly better conditioned than the randomly chosen alternatives.

However, there is no guarantee to find nodes beneath a certain condition number bound, i.e., our method might chooses nodes which are arbitrary ill conditioned. Though the occurrence of such nodes is rare, one of the remaining challenges is to develop an oracle responding faster than the computation of the condition number that rejects ill conditioned node distributions. Increasing the regularity of Vandermonde matrices within DC PSE schemes is supposed to improve the performance in terms of accuracy and runtime. Whatsoever, a benchmark with state of the art implementations is outstanding. A generalisation of the approach to dimension 3 is currently in development.

References


