Acceleration of RBF-FD meshless phase-field modelling of dendritic solidification by space-time adaptive approach

Tadej Dobravec a, Boštjan Mavrić a,b, Božidar Šarler a,c,*

a Faculty of Mechanical Engineering, University of Ljubljana, Alkerševa cesta 6, Ljubljana, SI-1000, Slovenia
b Division of Scientific Computing, Department of Information Technology, Uppsala University, P.O. Box 337, Uppsala, SE-751 05, Sweden
c Institute of Metals and Technology, Lepi pot 11, Ljubljana, SI-1000, Slovenia

ARTICLE INFO

Keywords:
Dendritic solidification
Phase-field method
Meshless methods
Space-time adaptivity
Scattered nodes

ABSTRACT

A novel adaptive numerical approach is developed for an accurate and computationally efficient phase-field modelling of dendritic solidification. The adaptivity is based on the dynamic quadtree domain decomposition. A quadtree decomposes the computational domain into rectangular sub-domains of different sizes. Each sub-domain is extended to ensure overlap communication between neighbouring sub-domains. In each sub-domain, uniform distribution of computational nodes is generated. The product between the node density and the sub-domain area is fixed to ensure the h-adaptivity. The adaptive approach provides the highest density of computational nodes at the solid-liquid interface and the lowest density in the bulk of the phases. The meshless radial basis function generated finite difference (RBF-FD) method is applied for the spatial discretisation of the partial differential equations which arise from the phase-field model. The RBF-FD method is especially appealing since it allows straightforward spatial discretisation of partial differential equations on scattered node distributions. The use of scattered node distribution reduces the discretisation-induced anisotropy in the phase-field modelling of dendritic growth. The forward Euler scheme is used for temporal discretisation. The adaptive time-stepping is employed to speed up the calculations further. The performance of the novel numerical approach is tested for dendritic solidification of supercooled pure melts and supersaturated dilute binary alloys at arbitrary preferential growth directions. The impact of the numerical parameters on the accuracy and computational efficiency is thoroughly analysed. It is shown that the RBF-FD method, defined on scattered node distribution, together with the space-time adaptive approach, represents an accurate and efficient technique for solving the phase-field models of dendritic solidification.

1. Introduction

The dendritic solidification is one of the most common and consequently studied phenomenon in the solidification of metals [1,2]. It was brought to the attention of both scientists and engineers due to its interesting pattern selection and useful industrial applications. The microstructure evolution during the casting of metals significantly affects the mechanical, physical, and chemical properties of the solidified material [3]. The grain size, interdendritic spacing, and solute distribution in dendritic solidification, for instance, essentially affect the hardness [4] and corrosion resistance [5] of the material. The prediction of microstructure evolution at various casting conditions is, therefore, crucial for the design and production of high-quality castings for scientific, medical, and industrial use. The dendritic solidification can be described by many different formulations [1,2], e.g., by the cellular automaton [6], level set [7–9], and phase-field (PF) [10,11] methods. In this paper, the PF concept is used for the modelling of dendritic solidification. Over the last three decades, this concept has become the preferred one for modelling free boundary problems in materials science [10–14]. The PF method has been successfully applied in many research areas, e.g., for modelling solidification phenomena [10–13], solid-state phase transformations [15–17], coarsening and grain growth [18,19], crack propagation [20–22], and two-phase flow [23,24]. In this paper, the PF models for the solidification of supercooled pure...
melts [25] and supersaturated dilute binary alloys [26] are considered.

The PDEs, which arise from a PF model, are usually spatially discretised by mesh-based methods, e.g., finite difference [27], finite volume [28], or finite element [29] methods. Meshless methods [30–32] represent an alternative to the mesh-based methods in the sense that a predefined mesh is not a prerequisite for the spatial discretisation of the PDEs. Some of the main advantages of the meshless methods compared to the mesh-based methods are [33] the more straightforward development of h-adaptive algorithms, easier treatment of free-boundary problems, no mesh-alignment sensitivity, and higher accuracy. In this paper, the meshless radial basis function generated finite difference (RBF-FD) method [34–36], also named local radial basis function collocation method (LRBFCM) [37], is applied for the spatial discretisation of the PDEs. The method originates from the Kansa method [38,39] for the solution of parabolic, hyperbolic, and elliptic PDEs by using interpolation with radial basis functions (RBF). The interpolation by RBF proves to be very accurate; however, Kansa method is global which results in dense and ill-conditioned interpolation matrices. This problem is overcome by the construction of local interpolation as in the meshless weak-form local radial point interpolation method [40] and in the local strong-form meshless methods [41–43,37], i.e., in the RBF-FD methods.

The RBF-FD method has been successfully applied to many different scientific and engineering problems, e.g., turbulent combined forced and natural convection problems [44], simulation of laminar backward facing step flow under a magnetic field [45], simulation of macrosegregation [46], h-adaptive solution of partial differential equations [47], r-adaptive solution of PF model for dissolution of primary particles in binary aluminium alloys [17], simulation of linear and transient thermo-elasticity [48,49], multi-pass hot-rolling simulation [50], the numerical simulation of the PF crystal models [51], solution of multi-dimensional Cahn-Hilliard, Swift-Hohenberg and PF crystal equations [52], multilevel solution of Poisson equation [53], PF modelling of dendritic solidification [36], and for solution of Hamiltonian PDEs using symplectic and multi-symplectic local RBF collocation methods [54].

In the Kansa method [38] and many RBF-FD method applications to different physical problems [47,44,48,50,56,55], multiquadratics (MQ) are used as shape functions. The use of MQ produces very good results; however, MQ introduce a free shape parameter, which essentially influences the method’s performance. There are several techniques for the optimal shape-parameter search [55,56]. The search has to be performed for each computational node independently, leading to long computational times, especially for large engineering problems. In recent years, shape parameter-free polyharmonic splines (PHS) have gained popularity as shape functions in the RBF-FD methods [34,35]. It has been recently shown [34,35] that neither PHS nor any other shape functions, but the highest-order monomial controls the h-convergence of the RBF-FD method. In contrast, the shape functions control the accuracy. Influenced by these results, an interpolation with PHS as shape functions, augmented with monomials up to the selected order of h-convergence, is used in the RBF-FD method in this paper.

The spacing between the computational nodes must be approximately equal to the PF interface thickness to resolve a PF model correctly. The size of a computational domain is typically at least a few orders of magnitude larger than the PF interface thickness, even in the smallest realistic cases of interest. This makes the numerical solution of a PF model a computationally expensive task. The restriction in selecting the spacing between the computational nodes only applies at and near the interface between phases. In the bulk of phases, the spacing can be increased without the loss of accuracy. Consequently, adaptive approaches have been successfully applied for the solution of PF models [57–62]. In the present paper, a novel 2-D space-time adaptive approach based on the algorithm from [62] is developed. The approach can be seen as an adaptive domain decomposition [63] where the computational domain is dynamically decomposed into sub-domains of different sizes by quadtree algorithm [64]. An h-adaptivity is ensured by the constant ratio between the characteristic size of a sub-domain and the node spacing in it. The algorithm dynamically ensures that the evolving solid-liquid interface always lies in the sub-domains with the highest node density. An adaptive-time stepping further speeds up the calculations. In contrast to the algorithm from [62], the RBF-FD method is used instead of the finite difference method for the spatial discretisation of the PDEs. This selection provides enhanced flexibility regarding the shape of a sub-domain and the type of node distribution in it.

This paper represents the continuation of our previous publication [36] where the RBF-FD method is used for the PF modelling of dendritic solidification of pure supercooled melts [25] by using uniform regular and scattered node distributions. We have shown that using the scattered node distribution speeded up the computational time and introduced the coarsening process in the PF modelling of dendritic solidification, present when regular node distribution is in use [65]. Our previous approach [36], however, uses uniform node distribution with constant node spacing throughout the entire computational domain and is, therefore, computationally inefficient. The main aim of the present paper is to enhance computational efficiency by applying a space-time adaptive approach. The accuracy and speed-up of the adaptive approach in comparison to our previous work are thoroughly analysed. The adaptive approach is based on the quadtree algorithm from [64], where we have developed a cellular automaton method for simulation of dendritic and eutectic growth using an adaptive mesh refinement.

2. Governing equations

2.1. Solidification of pure supercooled melts

As in our previous study [36], solidification of a pure supercooled melt is considered. A simplified case is studied where the density ρ, the specific heat at constant pressure cp, and the thermal conductivity k in the whole computational domain are constant and the convection is neglected. The latent heat of melting and the melting temperature are denoted as Lm and Tm, respectively. The system of two coupled dimensionless governing equations for PF ϕ and the dimensionless temperature θ = (T − Tm)/(Lm/cp), is given as [25]

\[
\frac{\partial \phi}{\partial t} = -\nabla \cdot (\nabla \phi) - \nabla \cdot (\nabla \phi - \nabla \psi) + \sum_{i=1}^{n} \nabla \phi \left( \frac{\partial \phi}{\partial n} \right),
\]

\[
\frac{\partial \theta}{\partial t} = \nabla \cdot \left( \nabla \theta + \frac{\partial \phi}{\partial t} \right) = D \nabla^2 \theta + \frac{\partial \phi}{\partial t},
\]

where \( \alpha(n), \lambda, \) and \( D \) stand for the surface energy anisotropy function, the coupling parameter, and the dimensionless diffusivity, respectively. The values \( \phi = 1 \) and \( \phi = -1 \) denote solid and liquid phases, respectively. The anisotropy function \( \alpha(n) \) is given as

\[
\alpha(n) = (1 - 3\epsilon_4) \left(1 + \frac{4\epsilon_4}{1 - 3\epsilon_4} \left( n_1^4 + n_2^4 + n_3^4 \right) \right),
\]

where \( \epsilon_4 \) stands for the strength of the cubic anisotropy and \( n = (n_1', n_2', n_3') \) is the normal \( n = \nabla \phi / |\nabla \phi| \) in the dendrite coordinate system

\[
n' = R(\theta_0) n,
\]

where

\[
R(\theta_0) = \begin{bmatrix} \cos \theta_0 & -\sin \theta_0 \\ \sin \theta_0 & \cos \theta_0 \end{bmatrix},
\]

is the rotation matrix, determined by the preferential growth direction \( \theta_0 \). The thin-interface limit [25] of the PF model yields relation

\[
D = a_s \lambda \text{ where } a_s = 0.6267.
\]

Parameter \( \lambda \) is therefore the only free parameter of the PF model. The parameter sets the characteristic interface thickness as [25]

\[
W_0 = \frac{1}{a_s} \lambda.
\]
Fig. 1. Illustration of a square computational domain \( \Omega \) with boundary \( \Gamma \). A domain is defined by the south-west coordinate \( r_{nw} \) and the side length of a square \( L \). Solidification from a supercooled melt is initialized by a small nucleus with radius \( R_{nuc} \) and centre \( r_{nuc} \). Zero flux Neumann boundary conditions are proposed for \( \phi \) and \( \theta \).

Table 1
Simulation parameters for solidification of pure supercooled melts [25].

<table>
<thead>
<tr>
<th>Physical problem</th>
<th>Physical problem parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strength of anisotropy (( \alpha_a ))</td>
<td>0.05</td>
</tr>
<tr>
<td>Initial supercooling (( \Delta ))</td>
<td>0.65</td>
</tr>
<tr>
<td>Center of nucleus (( R_{nuc} ))</td>
<td>(0,0)</td>
</tr>
<tr>
<td>Radius of nucleus (( R_{nuc} ))</td>
<td>10</td>
</tr>
<tr>
<td>PF model</td>
<td></td>
</tr>
<tr>
<td>Constant (( a_1 ))</td>
<td>0.8839</td>
</tr>
<tr>
<td>Constant (( a_2 ))</td>
<td>0.6267</td>
</tr>
<tr>
<td>Coupling parameter (( \lambda ))</td>
<td>( \frac{a_1}{a_2} )</td>
</tr>
<tr>
<td>Dimensionless diffusivity (( D ))</td>
<td>1</td>
</tr>
</tbody>
</table>

where \( d_0 \) is the thermal capillary length and \( a_1 = 0.8839 \). The PF model correctly captures the considered physical problem when \( W_0 \) is much smaller than the diffusion length of solidification. The spatial coordinates in Eqs. (1) and (2) are measured in the units of \( W_0 \) while the time is measured in the unit of characteristic attachment time

\[
t_0 = \frac{d_0^2}{D_T a_1^2}. \tag{7}
\]

where \( D_T \) stands for dimensional thermal diffusivity (\( D = D_T t_0/W_0^2 \)).

The computational domain \( \Omega \) with the boundary \( \Gamma \) is a square with the south-west vertex \( r_{nw} \) and the side length \( L \), as shown in Fig. 1. Zero flux Neumann boundary conditions are applied for \( \phi \) and \( \theta \) on \( \Gamma \). The initial condition for \( \theta \) is a constant initial supercooling \( \Delta \). The initial condition for \( \phi \) is a circular nucleus with the origin \( r_{nuc} \) and radius \( R_{nuc} \). In this paper, we are considering a case defined by the parameters from Table 1. This is one of the cases from [25] for which the analytical solution for the steady-state growth velocity, obtained in the framework of the microscopic solvability theory (MST) [66], is also tabulated.

2.2. Solidification of supersaturated dilute binary alloys

In addition to the solidification of pure supercooled melts, we consider also isothermal solidification of a dendrite from a supersaturated dilute binary alloy with negligible diffusivity of solute in the solid phase. The equilibrium concentration in the liquid interface and the partition coefficient are denoted as \( C_j^* \) and \( k_0 \), respectively. The system of two coupled governing equations for PF \( \phi \) and dimensionless supersaturation \( U = (C - C_j^*)/(1 - k_0)/C_j^* \) is given as [67]

\[
\begin{align*}
\frac{1}{2} (1 + k_0 (1 - k_0) U) \phi_t + \nabla \cdot \nabla \phi &= \phi - \phi^3 - (1 - \phi^2)^2 \lambda U \\
+ \nabla \cdot (\phi^2 \nabla U) + \sum_{i=x,y} \Delta \left( \sqrt{V} \phi \right)^2 a(n) \left( \frac{\partial a(n)}{\partial \phi} \phi \right), \tag{8}
\end{align*}
\]

Table 2
Simulation parameters for solidification of dilute binary alloys [26].

<table>
<thead>
<tr>
<th>Physical problem</th>
<th>Physical problem parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition coefficient (( k_0 ))</td>
<td>0.15</td>
</tr>
<tr>
<td>Strength of anisotropy (( \alpha_a ))</td>
<td>0.02</td>
</tr>
<tr>
<td>Initial supersaturation (( U ))</td>
<td>0.55</td>
</tr>
<tr>
<td>Center of nucleus (( R_{nuc} ))</td>
<td>(0,0)</td>
</tr>
<tr>
<td>Radius of nucleus (( R_{nuc} ))</td>
<td>( 2a_1/\lambda )</td>
</tr>
<tr>
<td>PF parameters</td>
<td></td>
</tr>
<tr>
<td>Constant (( a_1 ))</td>
<td>0.8839</td>
</tr>
<tr>
<td>Constant (( a_2 ))</td>
<td>0.6267</td>
</tr>
<tr>
<td>Coupling parameter (( \lambda ))</td>
<td>( 2\alpha_a/a_2 )</td>
</tr>
<tr>
<td>Dimensionless diffusivity (( D ))</td>
<td>2</td>
</tr>
</tbody>
</table>

The cubic anisotropy function \( a(n) \) is given by Eq. (3). The thin-interface limit yields the same relations as for solidification of pure melts, i.e., \( D = a_2/\lambda \). Parameter \( \lambda \) is therefore again the only free parameter of the PF model. The characteristics interface thickness and the characteristics attachment time are in this case set as

\[
W_0 = d_0 \frac{1}{a_1} \lambda, \tag{10}
\]

and

\[
r_0 = \frac{d_0^2}{D_T a_2^2} \lambda^3, \tag{11}
\]

where \( d_0 \) and \( D_T \) stand for the chemical capillary length and the solute diffusivity in the liquid phase, respectively. The dimensionless diffusivity from Eq. (9) is defined as \( D = D_T t_0/W_0^2 \).

The initial condition for the supersaturation is the constant supersaturation \( Y \). The initial condition for the PF is a circular nucleus with the origin \( r_{nuc} \) and the radius \( R_{nuc} \). Zero-flux Neumann boundary conditions are applied for \( \phi \) and \( U \). In this paper, we are considering a case defined by the parameters from Table 2. This is the case from [26] where the PF model for solidification of dilute binary alloys was originally developed.

3. Numerical method

3.1. Forward Euler scheme

We have applied the forward Euler scheme and the RBF-FD method for temporal and spatial discretisation of PDEs from Section 2. The forward Euler scheme is especially suitable for time-stepping in the PF modelling of dendritic solidification due to simplicity, straightforward parallel computing, and easy implementation in the adaptive time-stepping algorithms. The accuracy of the temporal discretisation can be generally increased using higher order single-step methods like modified Euler, midpoint, and higher order Runge-Kutta methods. We performed numerical experiments with fourth-order Runge-Kutta methods in the early stage of the research; however, the difference with the results obtained using the first-order Euler scheme was minimal and not worth the additional computational cost. The higher-order methods should generally allow larger time steps, at least for linear problems. On the other hand, we found that smaller time steps have to be used when using the fourth-order Runge-Kutta method in comparison to the first-order Euler scheme. The reason for smaller stable time steps in the fourth-order Runge-Kutta method is probably the high non-linearity of PDEs for PF modelling of dendritic growth. The main problem with the first-order forward Euler scheme is the von Neumann stability criterion. In the literature [25,60], this problem is successfully overcome using implicit methods, especially for the propagation of diffusion equations...
in the systems of PDEs from Section 2. In some realistic materials, the dimensionless diffusivity can become very large, leading to very small stable time steps.

The RBF-FD method is applied to discretise the PDEs on regular and scattered sets of computational nodes with the characteristic node spacing \( \Delta h \). The time step in the forward Euler scheme is determined as [14]

\[
\Delta t = a_M \frac{1}{4} \frac{\Delta h^2}{\max(D, 1/|a(n)|)},
\]

where \( a_M < 1 \) is the time step stability parameter. The stability of the forward Euler scheme depends on the method for spatial discretisation. We have found out in our previous study [36], that the value \( a_M = 0.1 \) ensures stable and accurate calculations for all tested parameters of the RBF-FD method.

The term \( 1/|a(n)| \) from Eq. (12) varies as a function of the normal \( n \); hence, the minimum of the anisotropy function is used in the stability criterion. The anisotropy function from Eq. (3) has a minimum \( 1 - \epsilon_z \) at \( n = (\cos \pi/4, \sin \pi/4) \). The time step is therefore given as

\[
\Delta t = a_M \frac{1}{4} \frac{\Delta h^2}{\max(D, 1/(1 - \epsilon_z))}.
\]

In the case of supercooled pure melts, the parameters from Table 1 yield

\[
\Delta t = a_M (1 - \epsilon_z) \Delta h^2/4,
\]

since \( D < 1/(1 - \epsilon_z) \). In the case of supersaturated binary alloys, the parameters from Table 2 yield

\[
\Delta t = a_M \Delta h^2 / D/4,
\]

since \( D > 1/(1 - \epsilon_z) \).

3.2. Meshless RBF-FD method

The interpolation with RBFs represents the core of the RBF-FD method. The interpolation procedure is presented first, followed by the procedure for evaluating spatial operators in the computational nodes. For simplicity, the method is presented for the case of scalar fields. The same procedure is also used for the case of vector and tensor fields.

3.2.1. Local interpolation with RBFs

We are considering a computational domain \( \Omega \) with the boundary \( \Gamma \). The domain is represented by \( N_{\text{aug}} \) nodes \( r \in \Omega \cup \Gamma, i = 1, ..., N_{\text{aug}} \). A local sub-domain \( r_i \Omega \) consists of \( N \) nodes \( r_j, j = 1, ..., N \) where \( r_j \neq r \) and \( r_j \in \Omega, i = 2, ..., N \) are the nearest nodes to \( r \). A local sub-domain is schematically shown in Fig. 2. The details of the construction of a local sub-domain are presented in [56].

The characteristic size of \( r_i \Omega \) is defined as

\[
\hat{r} = \sqrt{\frac{1}{N-1} \sum_{j=2}^{N} |r_j - r|^2}.
\]

An arbitrary scalar field \( \eta \) at \( r \in \Omega \cup \Gamma \) is approximated as

\[
\eta(r) \approx \sum_{i=1}^{N} a_i \Phi_i(r),
\]

where \( r \) is the computational node closest to \( r \). Constants \( a_i, i = 1, ..., N \) are the interpolation coefficients and the function \( \Phi_i \) is a RBF centred at \( r_i \). The application of Eq. (17) at each \( r_i \) from \( \Omega \) yields a system of equations for coefficients \( a_i \). The system of equations is well-posed and therefore yields a unique solution if the used RBFs are strictly positive definite functions [68]. If conditionally positive definite functions are used, RBFs have to be augmented with monomials to ensure a well-posed interpolation problem

\[
\eta(r) \approx \sum_{i=1}^{N} a_i \Phi_i(r) + \sum_{i=1}^{N_{\text{aug}}} b_i N_{\text{aug}}(r),
\]

where

\[
|p_1(r) = 1, |p_2(r) = \frac{x - x}{\hat{r}}, |p_3(r) = \frac{y - y}{\hat{r}}, ... .
\]

The number \( N_{\text{aug}} \) as a function of the highest degree of the augmentation monomials \( P \) and the number of dimensions \( N_{\text{dim}} \) is given as

\[
N_{\text{aug}} = \frac{(P + N_{\text{dim}})!}{P! N_{\text{dim}}!}.
\]

The system of equations constructed solely according to Eq. (18) is underdetermined and therefore requires additional relations which are obtained from the condition that the vector of RBF coefficients \( a_i \) is orthogonal to the polynomials evaluated at the nodes from \( \Omega \) [69]

\[
\sum_{i=1}^{N} b_i p_j(r_i) a_i = 0, \sum_{i=1}^{N} b_i p_k(r_i) a_j = 0, \sum_{i=1}^{N} b_i p_N_{\text{aug}}(r_i) a_i = 0.
\]

It is mathematically proven [68] that the interpolation with conditionally positive definite RBFs of order \( m \geq 2 \) is well-posed if the augmentation with monomials at least up to the order \( m - 1 \) is used. While the positions of the nodes in \( \Omega \) are arbitrary in the interpolation solely by RBFs [68], the nodes have to be unisolvent [69] if the interpolation problem is augmented by monomials.

If a node from \( \Omega \) lies on the boundary \( \Gamma \), where the linear boundary condition \( B(r) \eta(r) = b(r) \) applies, \( B \) is simply applied to Eq. (18) at that node in the construction of the system of equations. The system of equations which accounts for the augmentation with \( N_{\text{aug}} \) monomials and the boundary condition applied at the node \( r_j \in \Gamma \) reads as

\[
\mathbf{A} \mathbf{a} = \mathbf{f},
\]

where

\[
\mathbf{A} = \\
\begin{bmatrix}
\Phi_1(1) & \Phi_1(2) & \Phi_1(3) & \cdots & \Phi_1(N) \\
\Phi_2(1) & \Phi_2(2) & \Phi_2(3) & \cdots & \Phi_2(N) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\Phi_{N}(1) & \Phi_{N}(2) & \Phi_{N}(3) & \cdots & \Phi_{N}(N) \\
\end{bmatrix},
\]

\[
\begin{bmatrix}
p_{1}(1) & p_{1}(2) & p_{1}(3) & \cdots & p_{1}(N) \\
p_{2}(1) & p_{2}(2) & p_{2}(3) & \cdots & p_{2}(N) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
p_{N_{\text{aug}}}(1) & p_{N_{\text{aug}}}(2) & p_{N_{\text{aug}}}(3) & \cdots & p_{N_{\text{aug}}}(N) \\
\end{bmatrix},
\]
3.2.2. Evaluation of differential operators

The approximation from Eq. (18) is used for the evaluation of any linear differential operator \( D \) applied to a scalar field \( \eta \) in \( \mathbf{r} \in \Omega \cup \Gamma \), while the boundary condition in \( \mathbf{r} \in \Gamma \) is analytically satisfied in the interpolation problem from Eqs. (22), (23) and (24). Applying \( D \) to Eq. (18) yields

\[
D_{\eta}(\mathbf{r}) \approx \sum_{\mathbf{k} \in \mathcal{N}_{\text{ext}}} j_{\mathbf{k}} A^{-1}_{\mathbf{k}} \mathcal{W}_{\mathbf{k}}(\mathbf{r}),
\]

since \( j_{\mathbf{a}} \) are constants, \( \mathbf{r} \) is the computational node closest to \( \mathbf{r} \) and function \( \mathcal{W}_{\mathbf{k}} \) is either a RBF \( (i \leq N) \) or a monomial \( (i > N) \). By calculating the inverse of the matrix from Eq. (23), Eq. (25) applied at \( \mathbf{r} \) can be rewritten as

\[
D_{\eta}(\mathbf{r}) \approx \sum_{k=1}^{N} j_{\mathbf{k}} A^{-1}_{\mathbf{k}} \mathcal{W}_{\mathbf{k}}(\mathbf{r}),
\]

where \( j_{\mathbf{A}} \) is an element of matrix \( J \). Eq. (26) can now be written in a standard finite-difference-like manner as

\[
D_{\eta}(\mathbf{r}) \approx \sum_{k=1}^{N} j_{\mathbf{k}} u_{\mathbf{k}} A^{-1}_{\mathbf{k}} \mathcal{W}_{\mathbf{k}}(\mathbf{r})).
\]

The summation in Eq. (27) goes from \( k = 1 \) to \( k = N \), since \( j_{\mathbf{k}} = 0 \) for \( k > N \) according to Eq. (24).

### 3.3. Space-time adaptive approach

The space-time adaptive approach is based on the algorithm from [62] with three major distinctions. First, the RBF-FD method is used instead of the finite difference method for the spatial discretisation of the governing equations. Second, along with the regular node distribution, the scattered node distribution is also applied. Third, along with the spacing \( \Delta h \), the time step \( \Delta t \) is also adaptive. A scheme of the space-time adaptive approach is shown in Fig. 4. The approach is based on the quadtree domain decomposition of the computational domain into quadtree sub-domains of different sizes. Node distribution is generated in each quadtree sub-domain. The h-adaptivity is ensured by keeping the ratio between the side length of a quadtree sub-domain and the spacing between the computational nodes fixed. The adaptive time-stepping is employed to speed up the calculations further. The stable time step in the forward Euler scheme depends on \( \Delta h \), hence, different time steps can be used in quadtree sub-domains with varying densities of the nodes. For clarity, the adaptive approach is presented for the case of square computational domains, although the approach for rectangular computational domains is precisely the same. It can be extended to arbitrary shapes of the computational domain; however, this is not the focus of the present paper.

#### 3.3.1. Quadtree data structure

A tree is a data structure consisting of nodes and links without having any cycles [70]. The nodes are hierarchically organized so that every node has at most one parent and an arbitrary number of children. There are three possible types of nodes: root, i.e., a node without a parent, inner node, i.e., a node with a parent and children, and leaf, i.e., a node without children. The level (or depth) \( N \) of the node is defined as

\[
N = N_{\text{parent}} + 1, \quad N_{\text{root}} = 0,
\]

where \( N_{\text{root}} \) and \( N_{\text{parent}} \) stand for the root’s level and the level of the node’s parent, respectively.

The quadtree is a tree data structure in which the root and inner nodes have exactly four children [71]. Each node represents a square
3.3.2. Quadtree domain decomposition

The quadtree algorithm [64] decomposes a square computational domain $\Omega$ with side length $L$ into $N$ computational sub-domains $\Omega_i$, $i = 1, \ldots, N$, represented by the leaves on a tree. The side length $L_i$ of a square $\Omega_i$ is determined according to the level of a leaf as $L_i = L/2^N$. The neighbouring of a $\Omega_i$ is defined as a set of sub-domains $\{\Omega_j\}, i \neq j$ which have a common edge or vertex with $\Omega_i$. The recursive nearest-neighbours search [72] provides a straightforward and computationally efficient way to determine the neighbourhood. The difference in levels between neighbouring sub-domains defines whether a quadtree is balanced or non-balanced. In a balanced quadtree, the difference is at most one.

Each $\Omega_i$ is discretised by a uniform node distribution with the density of the computational nodes

$$\rho_i = 1/\Delta h_i^2,$$

where $\Delta h_i$ is the characteristic spacing between the neighbouring computational nodes in $\Omega_i$. The h-adaptivity is ensured by the constant product between the density and the area of $\Omega_i$.

$$L_i/\Delta h_i = L_j/\Delta h_j = \text{const.}$$

The scheme of computational nodes in quadtree sub-domains.

Fig. 3. Linear (left) and cubic (right) PHS centred at the origin $r_i = (x_i, y_i) \in \mathbb{R}^2$.

Fig. 4. Scheme of the space-time adaptive approach. The quadtree domain decomposition of computational domain (left). Computational nodes in the quadtree sub-domains on two successive levels of a quadtree are marked red and blue. Time stepping on different levels on a quadtree (right). Different time steps can be applied in quadtree sub-domains since the stable time step in the forward Euler scheme is a function of the node spacing. Time stepping and communication between neighbouring quadtree sub-domains on different levels are marked by solid arrows and the dotted two-way arrow, respectively.

Fig. 5. Scheme of a quadtree data structure.

(or a rectangle), which is divided by the children into four equal-sized squares (rectangles) as seen in Fig. 5. The quadtree data structure is especially appealing since it allows straightforward dynamical refinement and de-refinement of critical and non-critical areas in the computational domain, i.e., refinement of the regions at the solid-liquid interface and de-refinement of the regions in the bulk of the phases. Another appealing feature of the quadtree is the simple and computationally efficient search of the nearest neighbours [72].
3.3.4. Computational node arrangement

The size of $\Omega^*$ is according to Eq. (34) determined by the level $\mathcal{N}$ of $\Omega$ ($L_i = L_i/2^\mathcal{N}$), the position of $\Omega^*$ in $\Omega$, and the user-defined overlapping parameter $n^*$. The density of computational nodes $\rho_i = 1/\Delta h_i^2$ in $\Omega^*$ is set according to the user-defined ratio

$$m_n = \frac{L_i}{\Delta h_i}.$$  

(35)

Regular node distribution (RND) or scattered node distribution (SND) is generated in each $\Omega^*$. The node distributions are illustrated in Fig. 8.

While the construction of RND with the spacing $\Delta h_i$ is straightforward, the generation of SND [56,73] is more complex. Firstly, $N_s$ computational nodes (marked red in Fig. 9) are positioned on the boundary $\Gamma_{\Omega}^*$ with the spacing $\Delta h_i$. The nodes are not positioned at the corners of $\Omega^*$. Secondly, $N_{ib}$ first inner nodes (marked blue in Fig. 9) are positioned in $\Omega^*$. Each boundary node has one first inner node in the opposite direction to the outward-facing normal at the boundary node. The distance between them is $\Delta h_i$. Thirdly, $N_i = N_{all} - N_{s} - N_{ib}$ inner nodes (marked black in Fig. 9) are randomly positioned into a polygon determined by the first inner nodes, where $N_{all}$ is the total number of computational nodes in $\Omega^*$ determined according to the density $\rho_i$ and the area of $\Omega^*$. Finally, the positions of randomly distributed inner nodes are adjusted in a minimization process, similar to the one in the node repel algorithm from [74] in order to obtain locally isotropic node distribution. The nodes after each of four steps in the generation of uniform node distribution in $\Omega^*$ are shown in Fig. 9. A SND has two parameters: the node density $\rho_i$ and the seed $S$ of pseudo-random numbers used for the generation of randomly positioned nodes. Different seeds $S$ yield different SNDs.

3.3.5. Refinement procedure

The refinement procedure ensures a high density of computational nodes in the critical regions of the computational domain. These regions are defined by the refinement conditions, which are functions of field values and/or their gradients. The refinement conditions are usually fulfilled in the regions with large gradients where the density of computational nodes has to be large enough to ensure stable and accurate numerical evaluation of spatial operators. In this paper, we use the refinement conditions from sub-section 3.3.7. They ensure refinement at and near the solid-liquid interface.

The refinement procedure ensures user-defined minimum node spacing $\Delta h_{\text{min}}$ in all sub-domains $\Omega^*$, in which the refinement condition is fulfilled as presented in Algorithm 1. In the algorithm, the refinement condition is checked in each $\Omega^*$ on the quadtree for which $\Delta h_i > \Delta h_{\text{min}}$. The sub-domains in which the condition is fulfilled are flagged (Step 1 in Algorithm 1) and counted (Step 2 in Algorithm 1). A quadtree is checked again multiple times in order to prevent potential non-balancing from occurring. A sub-domain is flagged if a quadtree would become non-balanced after the refinement of already flagged sub-domains (Step 3 in Algorithm 1) as shown in Fig. 10. In the refinement (Step 4 in Algorithm 1), each flagged $\Omega^*$ is divided into four child sub-domains $\{\Omega^*_i\}$. The whole procedure is repeated (Step 5 in Algorithm 1) until the number of flagged sub-domains is zero.
Algorithm 1: Refinement algorithm.

Result: $\Delta h_i = \Delta h_{\max}$ in those $\Omega^*$ where the refinement condition is fulfilled.

Step 1: Flag each $\Omega^*$ with $\Delta h_i > \Delta h_{\max}$ in which the refinement condition is fulfilled;
Step 2: Store the number of flagged sub-domains to variable $\#flags$;
while $\#flags > 0$ do
Step 3: Flag additional sub-domains to ensure the balancing;
Step 4: Refinement of each flagged $\Omega^*$ by four children sub-domains $\{\Omega^{***}\}$;
Step 5: Perform flagging according to Step 1 and calculate new $\#flags$
according to Step 2;
end

Algorithm 2: De-refinement algorithm.

Result: Maximum possible $\Delta h_i$ in those $\Omega^*$ where the de-refinement condition is fulfilled.

Step 1: Flag each $\Omega^*$ with $\Delta h_i < \Delta h_{\min}$ in which the de-refinement condition is fulfilled if the balancing is sustained after the potential de-refinement;
Step 2: Store the number of flagged sub-domains to variable $\#flags$;
while $\#flags > 0$ do
Step 3: De-refine flagged domains;
Step 4: Perform flagging according to Step 1 and calculate new $\#flags$
according to Step 2;
end

Fig. 9. Nodes after first (far-left), second (centre-left), third (centre-right), and fourth (far-right) step in the generation of a scattered node distribution with the uniform density of the computational nodes.

Fig. 10. Refinement of a quadtree sub-domain (red) and an additional refinement of the neighbouring sub-domains (blue) to ensure a balanced quadtree.

When a flagged $\Omega^*$ is refined by forming four child sub-domains $\{\Omega^{***}\}$, the following steps are performed. First, the computational node distributions are generated in the child sub-domains $\{\Omega^{***}\}$. Secondly, the governing equations are discretised in child sub-domains $\{\Omega^{***}\}$ by the RBF-FD method and the forward Euler scheme. Thirdly, the field values in the child sub-domains $\{\Omega^{**}\}$ are interpolated from $\Omega^*$ and its neighbours. Finally, the stored data for the solution of governing equations and interpolation in $\Omega^*$ is freed.

3.3.6. De-refinement procedure

The de-refinement procedure ensures a low density of computational nodes in the non-critical regions of the computational domain. These regions are determined by the de-refinement conditions. The conditions are usually fulfilled in the regions with low gradients of the field values. In that regions, high accuracy of the numerical solution can be achieved even by using a low density of computational nodes. In this paper, the de-refinement condition from sub-section 3.3.7 is used. It ensures de-refinement in the bulk of phases.

The de-refinement algorithm ensures the maximum allowed node spacing $\Delta h_i \leq \Delta h_{\max}$ in all sub-domains $\Omega^*$ in which the de-refinement condition is fulfilled as presented in Algorithm 2. In the de-refinement algorithm, the de-refinement condition is checked in each $\Omega^*$ on the quadtree for which $\Delta h_i < \Delta h_{\max}$. The sub-domains in which the condition is fulfilled are flagged only if the quadtree remains balanced after the potential de-refinement (Step 1 in Algorithm 2) as shown in Fig. 11. The flagged sub-domains are counted (Step 2 in Algorithm 2). In the de-refinement (Step 3 in Algorithm 2), each flagged $\Omega^*$ is removed from the quadtree if its siblings are flagged for de-refinement too. The whole procedure is repeated (Step 4 in Algorithm 2) until the number of flagged sub-domains is zero.

When the flagged siblings $\{\Omega^{**}\}$ are de-refined, the following steps are performed. Firstly, the computational node distribution is generated in their parent $\Omega^*$. Secondly, the governing equations are discretised in their parent $\Omega^*$ by the RBF-FD method and the forward Euler scheme.

Fig. 11. Sub-domains with allowed refinement (left) and a case of sub-domains with prohibited de-refinement (right) by the constraints imposed by a balanced quadtree.

Thirdly, the field values in the parent $\Omega^*$ are set by the interpolation from the children $\{\Omega^{**}\}$ and their neighbours. Finally, the stored data for the solution of the governing equations and interpolation in siblings $\{\Omega^{**}\}$ is freed.

3.3.7. Refinement/de-refinement conditions

In the PF modelling of solidification, different strategies have been applied to determine the refinement/de-refinement conditions. For instance, the adaptation is triggered when the gradients of the fields exceed a user-defined threshold value [62]. In this paper, very simple refinement and de-refinement conditions are applied. The conditions were obtained from preliminary numerical experiments. The accuracy in comparison to the solution, obtained on the uniform node distribution in the whole computational domain, has been analysed in the derivation of the conditions.

The refinement condition ensures the minimum node spacing $\Delta h_{\min}$ in the areas where the PF rapidly changes from $\phi = -1$ to $\phi = 1$. On the other hand, the de-refinement condition ensures the maximum possible spacing (with the upper limit $\Delta h_{\max}$) in the bulk of the phases. The refinement and de-refinement conditions apply in each $r^*_i \in \Omega^*$ where $r^*_i$ is a node from the computational node distribution generated in
the extended sub-domain $\Omega^*_j$. We distinguish two types of refinement conditions. The first type ensures the refinement in the vicinity of the evolving solid-liquid interface, while the second type ensures the refinement according to the initial condition. A quadtree sub-domain $\Omega_i$ is flagged for refinement during the simulation if

$$|\phi(r_i)| < 0.95,$$  \hfill (36)

for any $r_i \in \Omega_i$. The initial condition for the PF is a solid nucleus with the origin $r_{\text{nuc}}$ and the radius $R_{\text{nuc}}$. The condition from Eq. (36) is at the beginning of the simulation applied to the computational nodes at the coarsest levels with large spacings between the computational nodes. The refinement condition from Eq. (36) fails in the refinement of the initial condition for the PF if all of the computational nodes lie in the bulk of the phases. This typically happens at the coarsest levels where $\Delta h_i > R_{\text{nuc}}$. An additional refinement condition is therefore applied at the beginning of the simulation. A quadtree sub-domain $\Omega_i$ with node spacing $h_i$ is flagged for refinement if

$$|r_i^* - r_{\text{nuc}}| - R_{\text{nuc}} < 2h_i,$$  \hfill (37)

for any $r_i^* \in \Omega_i$. The condition from Eq. (37) ensures refinement in the vicinity of the initial solid-liquid interface. A $\Omega_i$ is flagged for de-refinement if

$$|\phi(r_i^*)| > 0.99,$$  \hfill (38)

for all $r_i^* \in \Omega_i$. The condition from Eq. (38) ensures de-refinement in the bulk of phases.

3.3.8. An example of the refinement/de-refinement algorithm

An example of the refinement/de-refinement during the solidification of a spherical particle from a pure supercooled melt is shown in Fig. 12. The red and blue colours represent the solid ($\phi = 1$) and liquid ($\phi = -1$) phases, respectively, while the white colour denotes the solid-liquid interface ($\phi = 0$). The boundaries of the quadtree sub-domains are denoted by a green colour. The PF and the boundaries of the quadtree sub-domains ($\Omega_i$) are plotted for the initial configuration and the following two changes in the quadtree.

In the initial configuration, the algorithm adapts the area around the nucleus according to Eq. (37). As the nucleus starts to grow, the condition from Eq. (38) is fulfilled in the centre of the solid particle, which results in de-refinement. When the solid particle is large enough, additional sub-domains in the liquid phase are refined according to Eq. (36).

The refinement and de-refinement conditions apply in the nodes $r_i^* \in \Omega_i$ shown in Fig. 13. The conditions do not apply in the nodes $r_i^* \notin \Omega_i$, i.e., the nodes from $\Omega_i^* \backslash \Omega_i$ which lie in the neighbouring sub-domains. An RND or SNR with the constant ratio $m_{\phi} = L_i/\Delta h_i = 10$ is generated in each $\Omega_i^*$ in the example from Fig. 13.

3.3.9. Adaptive time-stepping

The stable time step in the forward Euler scheme depends on the node spacing according to Eq. (12). Fine time steps are, therefore, required on sub-domains with fine node arrangements, while longer time steps are possible on sub-domains with larger node spacings. Ideally, maximum stable time step can be used in each $\Omega_i^*$ according to the value of $\Delta h_i$; however, this is not feasible in practice. The ratio between two stable time steps in the sub-domains with the level difference $\Delta N^*$ is $2^{\Delta N^*}$. For example, during one time step in the sub-domains on the level $N^*$, 1024 time steps are performed in the sub-domains on the level $N^* + 5$. Because the solution procedure is designed in such a way that the time stepping on all the computational sub-domains is performed first, followed by the refinement and de-refinement algorithms, the solidification front may “escape” from the sub-domains with $\Delta h_{\text{min}}$ in 1024 iterations, which yields large errors and most likely causes the collapse of the calculation. To overcome this problem, the maximum allowed time step is imposed on $\Omega_i^*$ as

$$\Delta t_i = \min \left[ \Delta t_i(h_i), \Delta t_i(2^{\Delta N^*} \Delta h_{\text{min}}) \right],$$  \hfill (39)

where the integer $m_{\phi} \geq 2$ stands for the user-defined time-step restriction parameter. The same time step $\Delta t_i = \Delta t_i(h_{\text{min}})$ is used in all the sub-domains if $m_{\phi} = 0$. The maximum allowed value of $m_{\phi}$ depends on the physical problem. In the case which is considered in this paper, value $m_{\phi} = 2$ is used to ensure the stability of the calculations. The maximum time-step is sixteen times larger than the minimum time-step when $m_{\phi} = 2$. Preliminary numerical experiments showed that the values $m_{\phi} > 2$ can be problematic, especially at high growth velocities.

The introduction of parameter $m_{\phi}$ reduces the computational gains of the solution procedure in order to ensure stability. Since only very small values of $m_{\phi}$ are permitted in the examples considered in this paper, the analysis of the impact of $m_{\phi}$ on the computational efficiency is not investigated further.

The adaptive time-stepping is presented using examples from Fig. 14. The time-stepping and communication between different levels are schematically shown on the left of Fig. 14. During the communication, the field values on the boundary $\Gamma_i^*$ of an extended quadtree sub-domain $\Omega_i^*$ are interpolated from the neighbouring extended sub-domains and vice versa. The time-stepping and communication between neighbouring sub-domains on the same and on the different levels is presented on the right of Fig. 14.

The time-stepping with different time steps on different levels is defined by Algorithm 3. In the time-stepping procedure, the extended sub-domains $\{\Omega_i^{n*}\}$ from a quadtree are segmented into sets $\{\Omega_i^{n*}\}_{n}^{1}$ of $N^*$ sub-domains on the level $N^*$. Evaluation of a time step in a quadtree sub-domain $\Omega_i^{n*} \subset \{\Omega_i^{n*}\}_{n}^{1}$ is independent of the evaluation of a time step in all the other sub-domains from $\{\Omega_i^{n*}\}_{n}^{1}$. Time-stepping in domains $\{\Omega_i^{n*}\}_{n}^{1}$ (Time step in Algorithm 3) is therefore performed in parallel. OpenMP [75] application programming interface for shared memory multiprocessor programming is applied for the parallel evaluation of a single time step on level $N^*$. The levels $N^*$ and $N^* + 1$ are synchronised when the solutions on levels $N^*$ and $N^* + 1$ are at the same time. Recursion (Recursion in Algorithm 3) is therefore applied to achieve synchronisation between levels $N^*$ and $N^* + 1$.

During the communication, the interpolation of field values is done in $\Omega_i^{n*}$ in each boundary node $r_k \in \{\Omega_i^{n*}\}_{n}^{1\text{high}}$ which lies in $\Omega_i^{n*}$. The neighbourhood $\{\Omega_i^{n*}\}_{n}^{1\text{high}}$ of each extended sub-domain $\Omega_i^{n*}$ is straightforwardly determined by the nearest neighbours search [72] after each change on a quadtree. The neighbouring quadtree sub-domains of a quadtree sub-domain on level $N^*$ can lie on levels $\{N^* - 1, N^*, N^* + 1\}$.
since a balanced quadtree is used. Interpolation of the field values during the communication (Communication in Algorithm 3) in a quadtree sub-domain \( \Omega^* \in [\Omega^*]_N \) is independent of the interpolation of the field values during the communication in all the other sub-domains from \([\Omega^*]_N\) and is therefore also performed in parallel by OpenMP.

3.3.10. Solution procedure

The RBF-FD method and the forward Euler scheme are used for the spatial and the temporal discretisations of the governing equations in each \( \Omega^* \). The RBF-FD method is also used for the interpolation of the field values to the boundary nodes in the communication between neighbouring sub-domains and in the interpolation during the refinement/de-refinement procedure. The solution procedure is presented in the Algorithm 4. Firstly, the parameters of the numerical method are set. Next, the refinement according to the initial condition is made. Finally, the iteration starts where each iteration consists of three steps: adaptive time-stepping, refinement, and de-refinement.

3.3.11. Numerical implementation and post-processing of the results

The novel numerical approach is implemented in the programming language Fortran 2008 and compiled with the Intel Visual Studio Compiler 19.0. Programming language Python with the libraries Matplotlib and Numpy was used for the post-processing and for graphical presentation of the numerical results.

4. Numerical results and discussion for solidification of pure supercooled melts

The accuracy and computational efficiency of the newly developed adaptive approach is analysed by solving the solidification examples from Section 2. Similar as in our previous paper [36], examples from [25] and [26] are extended to allow an arbitrary preferential growth direction. More precisely, the growth at values \( \theta_0 \in [0, \pi/4] \) is considered due to the cubic symmetry of the anisotropy function. The solid-liquid interface at the end of the simulation for three different values of \( \theta_0 \) is shown in Fig. 15. The accurate simulation of growth for arbitrary preferential directions is especially important during the solidification of multiple, differently oriented dendrites, which is the case in the industrial casting of metals. The solidification of supercooled pure melts is considered in this section, followed by the analysis of the solidification of supersaturated dilute binary alloys in the next section. The considered case is defined in Sub-section 2.1.
The growth velocity as a function of time for three values of $N$ using RND and SND in the case of BSP is shown in Fig. 18. In the case of RND, the best result is observed for $N = 13$. The solution is not converging to the correct value obtained by the MST for $N = 9$. The velocity is converging to a slightly higher value than the one predicted by the MST for $N = 21$. In the case of SND, the solution is converging to the correct value obtained by the MST for all three values of $N$. It is clear that the use of RND is much more sensitive to the selection of $N$ in comparison to the use of SND.

The rescaled growth velocity as a function of time in the case of ASP with $m_3 = 15$ is shown in Fig. 19. In the case of RND, the same behaviour as for BSP from Fig. 18 is observed. In the case of SND, the situation is quite different. The quadtree domain decomposition apparently introduces regularity in the solution obtained using SND, which shifts the velocity away from the solution obtained by BSP. The quadtree domain decomposition has an impact on the generation of SND, as seen in Fig. 20, especially near the boundary of a quadtree sub-domain. As $m_3$ is decreasing, the SND becomes more and more similar to the RND. This explains the shift of the velocities obtained using SND in the ASP away from the velocities obtained using SND in the BSP.

The effect of the regularity on the solution should decay with the increase of $m_3$, since the effect of the square boundary on the generation of SND decreases with the increase of $m_3$ according to Fig. 20. The rescaled growth velocity as a function of time on SND for $m_3 = 30$ and $m_3 = 60$ is shown in Fig. 21. For both values of $m_3$, the influence of regularity is still present, however, the results obtained with $m_3 = 60$ are much more closer to the results obtained by BSP in Fig. 18 in comparison to the results obtained with $m_3 = 15$ and $m_3 = 30$.

4.2. Growth from a supercooled pure melt at $\theta_0 \geq 0^\circ$

The case with $\theta_0 \geq 0^\circ$ is analysed using BSP and ASP. The procedures are tested for the values of $\theta_0$ in the range $\theta_0 \in [0^\circ, 45^\circ]$ with the orientation step of $5^\circ$.

4.2.1. Results of the BSP for $\theta_0 \geq 0^\circ$

A computational domain is defined by parameters $r_{\text{max}} = (-240, -240)$ and $L = 480$. The node spacings $\Delta h = 0.6$ and $\Delta h = 0.8$ are tested. The steady-state growth velocity as a function of $\theta_0$ for $N = 13$ and two values of $\Delta h$ obtained using RND and SND is shown in Fig. 22. The rescaled growth velocity, acquired in the framework of the MST, is also plotted. In the case of RND, $v_{\text{tip}}^{\text{steady}}$ departures from the MST growth velocity as $\theta_0$ increases. The departure decreases as the spacing is lowered from $\Delta h = 0.8$ to $\Delta h = 0.6$. We can also see that the velocity $v_{\text{tip}}^{\text{steady}}$ at $\theta_0 = 0^\circ$ is slightly lowered as $\Delta h$ decreases on RND. This is also observed in [25].

In the case of SND, the calculation has been performed on five different SNDs defined by five different seeds $S$. An error bar with the minimum, the maximum, and the median value of $v_{\text{tip}}^{\text{steady}}$ at each $\theta_0$ is plotted. The median value of $v_{\text{tip}}^{\text{steady}}$ is almost independent of $\theta_0$. As $\Delta h$ is decreased, the median is shifted towards the MST growth velocity and depends even less on $\theta_0$. Also, the error bars decrease as $\Delta h$ is lowered.
4.2.2. Results of the ASP for $\theta_0 \geq 0^\circ$

The side length of square $\Omega$ is in the case of ASP given as

$$L = 2^{\mathcal{N}_{\text{max}} \times m_\Omega} \Delta h_{\text{mix}},$$

(41)

where $\mathcal{N}_{\text{max}}$ is the maximum level of a quadtree. The side length $L = 480$, which is used in the case of BSP, therefore cannot be used simultaneously for values $\Delta h_{\text{mix}} = 0.6$ and $\Delta h_{\text{mix}} = 0.8$. Simulations are performed in a computational domain with parameters $r_{\text{sw}} = (-268.8,-268.8)$ and $L = 537.6$. As discussed at the beginning of section 4.1, the boundary does not have an impact on the growth until $t < 1500$ for $L = 480$ and consequently also for $L = 537.6$. Spacing $\Delta h_{\text{mix}} = 0.8$ is used in combination with ratios $m_\Omega = 21$ and $m_\Omega = 42$. Spacing $\Delta h_{\text{mix}} = 0.6$ is used in combination with ratios $m_\Omega = 28$ and $m_\Omega = 56$. Other two parameters of the APS are set to $n^* = 1$ and $m_{\Delta t} = 2$.

The steady-state growth velocity as a function of $\theta_0$ for $N = 13$ and two values of $\Delta h_{\text{mix}}$ obtained using RND and SND is shown in Fig. 24. In the case of RND, a very similar behaviour as in Fig. 22 for the BSP is observed. Interestingly, the growth velocity is slightly less affected by
Fig. 20. SND in an interior quadtree sub-domain for \( m_\Omega = 5 \) (left), \( m_\Omega = 10 \) (middle) and \( m_\Omega = 20 \) (right). Full and empty dots represent computational and communication nodes, respectively. Green square represents the boundary of a quadtree sub-domain.

Fig. 21. Rescaled growth velocity as a function of time for \( m_\Omega = 30 \) (left) and \( m_\Omega = 60 \) (right) using SND in the case of ASP.

Fig. 22. Steady-state growth velocity as a function of the preferential growth direction for two different spacings \( \Delta h \) and \( N = 13 \) using RND (left) and SND (right) in the case of BSP. Five simulations (five different SNDs) are run, and the minimum, the maximum, and the median velocity at each angle are plotted in the form of an error bar.

The orientation in the case of ASP, especially for \( \theta_3 \geq 35^\circ \). In the case of SND, the behaviour of the median is very similar to the behaviour in the case of BSP; the error bars are, however, larger.

The steady-state growth velocity as a function of \( \theta_3 \) for \( \Delta h = 0.8 \) and \( m_\Omega = 21 \) using RND and SND and three different values of \( N \) is shown in Fig. 25. For both node distributions, similar behaviour to that in Fig. 23 for a BSP is observed. It is, however, evident that the solution obtained by SND with ASP is more sensitive to rotation in comparison to the solution obtained by BSP, especially for \( N = 9 \).

The simulations with SND are additionally performed using \( m_\Omega = 42 \) in combination with \( \Delta h_{\text{min}} = 0.8 \) and \( m_\Omega = 56 \) in combination with \( \Delta h_{\text{min}} = 0.6 \) as shown in Fig. 26. The error bars are reduced as \( m_\Omega \) is increased in the study for \( N = 13 \) and two values of \( \Delta h_{\text{min}} \) in comparison to Fig. 24. The solution is less sensitive to the rotation at different values of \( N \) as \( m_\Omega \) is increased; however, for \( N = 9 \) the solution is still more sensitive to rotation in comparison to the BSP.

4.3. Computational efficiency

In this sub-section, the computational efficiencies of the ASP and BSP are compared to each other to analyse the speed-up of the ASP. In the analysis of the speed-up, the case with \( \theta_3 = 0 \) is considered. The computational efficiency is tested in \( \Omega \) with parameters \( r_{\text{sw}} = (0,0) \) and \( L = 256 \). The analysis is made on an HP ZBook laptop with the hexa-core Intel Core i7-9750H 2.6-4.5 GHz processor. In order to assess the speed-up of the adaptive procedure only, the simulations are performed on a single processor core. We are analysing the speed-up due to reduced number of computational nodes and time iterations. The choice of RND or SND for generation of computational nodes does not
Fig. 23. Steady-state growth velocity as a function of the preferential growth direction for three different sizes of local sub-domain \( N \) and \( \Delta h = 0.8 \) using RND (left) and SND (right) in the case of BSP. Five simulations (five different SNDs) are run at each \( N \), and the median velocity at each angle is plotted.

Fig. 24. Steady-state growth velocity as a function of the preferential growth direction using RND (left) and SND (right) for \( N = 13 \) and two values of \( \Delta h_{\text{min}} \) in the case of ASP. Five simulations (five different SNDs) are run, and the minimum, the maximum, and the median velocity at each angle are plotted in the form of an error bar.

Fig. 25. Steady-state growth velocity as a function of the preferential growth direction using RND (left) and SND (right) for \( \Delta h_{\text{min}} = 0.8 \) and \( m_\Omega = 21 \) in the case of ASP. Five simulations (five different SNDs) are run at each \( N \), and the median velocity at each angle is plotted.
have any impact on this speed-up. Hence, an RND is used in the analysis. The speed-up at different values of the ASP’s parameters is analysed. Ratios $m_Ω = 5, 10, 20, 40$ and $80$ and spacings $\Delta h_{min} = 0.4$ and $0.8$ are tested. The other two ASP parameters are set to $n^* = 1$ and $m_\Delta = 1$.

The ASP reduces the number of computational operations needed for performing a PF simulation. The speed-up has two contributions: the speed-up due to adaptive $\Delta h$ and due to adaptive $\Delta t$. The speed-up due to adaptive $\Delta h$ results from the reduced number of computational nodes for performing a PF simulation. The speed-up due to adaptive $\Delta t$ is the consequence of the reduced number of iterations. The maximum theoretical speed-up of the ASP is reduced due to the computational work for communication between neighbouring quadtree sub-domains and adaptation.

The maximum theoretical speed-up due to adaptive $\Delta t$ is reduced by setting the restriction from Eq. (39) to achieve stable calculations as discussed in sub-section 3.3.9. The maximum time step is set to $\Delta t_{max} = 4\Delta t_{min}$ ($m_\Delta = 1$) in the analysis of computational efficiency. Since only small values of $m_\Delta$ are permitted in the considered case, the speed-up due to adaptive $\Delta t$ is not investigated further.

The maximum theoretical speed-up due to adaptive $\Delta h$ is obtained at the minimum number of computational nodes. This maximum corresponds to the lowest allowed value of $m_\Omega$. However, lower values of $m_\Omega$ provide a larger number of quadtree sub-domains and consequently a larger number of communication nodes which, in turn, reduce the overall speed-up of the ASP as seen in Fig. 27. The Reduction of speed-up as a function of simulation time is also evident from Fig. 27 since the number of computational and communication nodes increase due to evolving solid-liquid interface.

We define the speed-up of the ASP as

$$\text{ASP speed-up} = \frac{\text{ERT}_{\text{ASP}}}{\text{ERT}_{\text{ASP}}}$$

where ERT_{BSP} and ERT_{ASP} stand for the elapsed real time (ERT) in the case of the BSP and ASP, respectively. A simulation by BSP is made on an uniform RND with $\Delta h = \Delta h_{min}$. ERT_{BSP} and ERT_{ASP} for $\Delta h_{min} = 0.8$
and $m_\Omega = 5$ are shown in Fig. 28. The total ERT has three contributions in the case of ASP. The first contribution is iteration ERT, i.e., ERT for time-stepping with the forward Euler scheme while using the RBF-FD method for spatial discretisation. The second contribution is communication ERT, i.e., ERT for interpolating boundary values by the RBF-FD method. Third contribution is adaptation ERT, i.e., ERT for refinement and de-refinement. Naturally, the iteration ERT represents the majority of the total ERT. The main goal of the ASP is to reduce the iteration ERT in comparison to the BSP. However, some extra computational work is introduced due to the communication and adaptation that reduce the total speed-up compared to the BSP.

The percentage of ERT for performing various tasks during the simulation as a function of time for $m_\Omega = 5$ and two values of $\Delta h_{\text{max}}$ is shown in Fig. 29. The iteration of the governing equations represents the majority of the whole ERT. For $\Delta h_{\text{max}} = 0.8$, adaptation takes a larger fraction of time in comparison to $\Delta h_{\text{max}} = 0.4$, while the percentage of communication is approximately the same for both spacings. The percentage of ERT for performing various tasks during the simulation as a function of time for $\Delta h_{\text{max}} = 0.4$ and four values of $m_\Omega$ is shown in Fig. 30. Naturally, the ERT for the adaptation and communication decreases as $m_\Omega$ is increased. The largest percentage of ERT for adaptation at the beginning of the simulation is observed for $m_\Omega = 20$. In the initial configuration for $m_\Omega = 20$, the solid nucleus is refined by only four quadtree sub-domains with $\Delta h_{\text{max}} = 0.4$ since the size of the nucleus is equal to $R_{\text{max}} = 10$ and the size of the smallest quadtree sub-domain is equal to $L_i = \Delta h_{\text{max}} \times m_\Omega = 8$ as seen in Fig. 31. As the solid phase starts to grow, many quadtree sub-domains have to be refined to ensure balancing, which explains the large percentage of the adaptation ERT. That behaviour is for the same reason observed for $\Delta h_{\text{max}} = 0.8$ and $m_\Omega = 10$. For $m_\Omega = 40$ and $m_\Omega = 80$, the adaptation does not occur until the dendrite is large enough, since a large area around the solid nucleus is refined in the initial refinement.

The speed-up as a function of time for different values of $m_\Omega$ and the final speed-up as a function of $m_\Omega$ are shown in Fig. 32. The speed-up is a decreasing function of time since more and more quadtree sub-domains have to be created over time due to the evolving solid-liquid interface. The maximum final speed-up is observed for $m_\Omega = 5$ and $m_\Omega = 10$. A higher speed-up is observed for $\Delta h_{\text{max}} = 0.4$.

At the beginning of the simulation, the RBF-FD coefficients for each spatial operator have to be calculated as described in our previous publication [36]. Preparation time $\text{ERT}_\text{prep}$, the ERT for calculation of the RBF-FD coefficients before the time-stepping begins, has not been analysed yet. In the case of BSP, the coefficients have to be calculated for each computational node in $\Omega$, which is a computationally expensive task, especially for a large number of computational nodes. In the case of ASP, the RBF-FD coefficients are calculated at the beginning of the simulation for nine possible extended sub-domains $\Omega^*_i$ with spacing $\Delta h_{\text{max}}$:

- four corner $\Omega^*_i$ (south-west, north-west, north-east, south-east) with two sides on the boundary of $\Omega$,
- four side $\Omega^*_i$ (west, north, east, south) with one side on the boundary of $\Omega$, and
- inner $\Omega^*_i$ without sides on the boundary of $\Omega$.
When a new \( \Omega^* \) is created in the refinement or de-refinement procedure during a simulation, the calculated RBF-FD coefficients are rescaled according to \( \Delta h_i \) and stored for the calculations in \( \Omega_i^* \). This hugely reduces preparation and adaptation ERT. The ERT for adaptation according to the initial condition is also accounted for in the ERT_{PREP}. The preparation ERTs for both solution procedures are shown in Table 3. Preparation takes much less time in the case of ASP, especially at small values of \( m_{\Omega} \) and \( \Delta h_{\text{min}} \). Naturally, preparation ERT increases with the ratio \( m_{\Omega} \).

5. Numerical results and discussion for solidification of supersaturated binary alloys

In this section, the newly developed adaptive approach is applied for the modelling of solidification of supersaturated dilute binary alloys. The analysis is very similar to the one for pure supercooled melts with two major distinctions. First, only the ASP is used in the analysis, and second, along with the growth velocity, the concentration in the solid phase is also considered in the results of a simulation. The speed-up of the ASP is not analysed in this case. The considered case is defined in Sub-section 2.2.

Fig. 30. The percentage of ERT as a function of simulation time for performing various tasks during the simulation for \( m_{\Omega} = 10 \) (top left), \( m_{\Omega} = 20 \) (top right), \( m_{\Omega} = 40 \) (bottom left), and \( m_{\Omega} = 80 \) (bottom right) at \( \Delta h_{\text{min}} = 0.4 \).

Fig. 31. Initial refinement for \( m_{\Omega} = 5 \) (far-left), \( m_{\Omega} = 10 \) (centre-left), \( m_{\Omega} = 20 \) (centre-right), and \( m_{\Omega} = 40 \) (far-right) at \( \Delta h_{\text{min}} = 0.4 \).

<table>
<thead>
<tr>
<th>( \Delta h_{\text{min}}/m_{\Omega} )</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>80</th>
<th>ASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>4 s</td>
<td>5 s</td>
<td>15 s</td>
<td>44 s</td>
<td>126 s</td>
<td>154 s</td>
</tr>
<tr>
<td>0.4</td>
<td>5 s</td>
<td>8 s</td>
<td>16 s</td>
<td>43 s</td>
<td>135 s</td>
<td>648 s</td>
</tr>
</tbody>
</table>

In the analysis, the dendrite’s growth velocity and the concentration in the solid phase are compared to the results from [26]. The same procedure as for pure melts is used for the calculation of growth velocity. The growth velocity from [26] is calculated as

\[
\dot{v}_{\text{gp}} = \frac{W_0 d_c}{\theta' - D_c} = \frac{\dot{v}_{\text{gp}} x_1}{x_2} \frac{1}{x_2^2}
\]

where \( v_{\text{gp}} \) is the dimensionless growth velocity. In addition to the growth velocity, the concentration in the solid phase \( C_s/C_s' \) along the \( x'-\)axis is verified, where \( x'-\)axis is the \( x\)-axis in the dendrite coordinate system determined by the preferential growth direction angle \( \theta' \). In the comparison of our results with the reference solution from [26], the dimensional time \( \tau = \tau_0 \) and coordinate \( x' = x'W_0 \) are re-scaled as

\[
\tau \rightarrow \tau D_c d_c^2, \quad x' \rightarrow x'/d_c.
\]
The re-scaled $\tilde{t}$ and $\tilde{x}'$ as a function of dimensionless $t$ and $x'$ are given as

$$\tilde{t} = t \frac{a_2}{a_1} \lambda, \quad \tilde{x}' = x' \frac{1}{a_1} \lambda.$$  \hfill (45)

In [26], the growth velocity is plotted in the range $\tilde{t} \in [0, 10000]$ and the concentration in the range $\tilde{x}' = [0, 400]$. With the parameter $\lambda = 2/a_2$, these values approximately correspond to dimensionless coordinates $t \in [0, 380]$ and $x' \in [0, 112]$. In order to compare the results, the size of the computational domain is set to $L = 268.8$ and the simulation is run until $t = 850$. This choice ensures a pseudo-infinite supercooled binary alloy, for which the impact of boundary on the solidification at the end of the simulation is negligible. The growth velocity and the re-scaled concentration in the solid phase from [26] are set as the reference solutions to which we compare our simulation results.

5.1. Growth from a supersaturated binary alloy at $\theta_0 = 0$

Re-scaled growth velocity $\tilde{v}_{tip}$ as a function of re-scaled time $\tilde{t}$ and the normalised concentration in the solid phase $C_s/C_L$ as a function of the re-scaled coordinate $\tilde{x}'$ are compared to the reference solutions for $\alpha_{at} = 0.3$, $\Delta h_{min} = 0.6$, and three values of $N$ on RND and SND as shown in Figs. 33 and 34.

The behaviour of the growth velocity on RND and SND is very similar to the behaviour in the case of pure materials; however, the growth velocity converges towards the steady-state growth velocity slower in comparison to the reference velocity. This observation is addressed at the end of this sub-section. On both node distributions, the best result is observed for the choice $N = 13$. For both RND and SND, the values $N = 9$ and $N = 21$ yield too low and too high growth velocities, respectively. However, the velocities at different values of $N$ are closer to each other in the case of SND.
Normalised concentration $C_j/C_j^\text{ref}$ as a function of the re-scaled co-ordinate $x'$ is much less sensitive to the selection of $N$ for both node distributions, as seen in Fig. 34. The concentration is slightly above the reference solution for both node distributions. In the case of RND, the concentration is approaching the reference solution as $N$ is increased. In the case of SND, the concentration is almost identical for all values of $N$. The influence of the scattered computational nodes on the performance is further analysed by increasing the size of a quadtree sub-domain $m_3$, as shown in Fig. 35. As in the case of pure materials, the velocities at $N = 9$ and $N = 21$ are closer to velocity at $N = 13$ for $m_3 = 56$ in comparison to $m_3 = 28$.

The reference velocity converges to the steady-state faster in comparison to our results. The reason is the fact that the PF model from [26] is not exactly identical to the PF model [67] used in this paper. The difference between the models is term $1 + (1 - k_0)U$ which multiplies the left-hand side of the governing equation for $\phi$ from Eq. (8). This term is equal to one in [26]. The growth velocity as a function of time for the case where we use exactly the same formulation as in [26] is shown in Fig. 36. For both node distributions with $N = 13$, the growth velocity is almost identical to the reference velocity in the interval $t \in [0, 10000]$. For $t > 10000$, the velocity is slightly reduced. For $N = 9$ and $N = 13$, the same behaviour in comparison to $N = 13$ as when using term $1 + (1 - k_0)U$ is observed.

In [26], the finite-difference method with spacing $\Delta h = 0.4$ and forward Euler scheme with time step $\Delta t = 0.008$ are used for spatial and temporal discretisation of the governing equations. The configuration that resembles the used finite-difference method the most is $N = 13$ on RND, as evidenced in Fig. 36. To compare the results with the same
spacing, the analysis with different values of $N$ on RND and SND is also performed for $\Delta h_{min} = 0.4$, as seen in Fig. 37. While the velocity at $N = 13$ for $\Delta h_{min} = 0.4$ differs very little from the velocity at $\Delta h_{min} = 0.6$, a huge improvement is observed for $N = 9$ and $N = 21$ on both node distributions.

5.1.1. Growth from a supersaturated binary alloy at $\theta_0 \geq 0$

For $\theta_0 \geq 0$, a two-times larger computational domain is used in comparison to the analysis at $\theta_0 = 0$, while the final time remains the same. The discretisation parameters are set to $\alpha_{AL} = 0.3$ and $\Delta h_{min} = 0.6$. The performance for only one SND is analysed, since the performance at multiple SNDS has already been assessed in the previous section in the example of pure materials. Steady-state growth velocity $v_{steady}$ as a function of preferential angle $\theta_0$ is shown in Fig. 38. Normalised steady-state concentration in the solid-phase $C_{steady}^s / C^s_{\infty}$ as a function of $\theta_0$ is shown in Fig. 39.

The behaviour of the method is very similar to the behaviour in the case of pure materials. Again, the use of SND is much less prone to the selection of $N$. The best result is again observed for $N = 13$ on both node distributions. In the case of RND, the concentration is independent of $\theta_0$ for $N = 13$. For $N = 21$, the concentration is firstly increased and then decreased as $\theta_0$ is increased with the maximum at $\theta_0 = 25^\circ$ and $\theta_0 = 20^\circ$ for $N = 9$ and $N = 21$, respectively. In the case of SND, the concentration is almost independent of $\theta_0$ for all three values of $N$. The concentration is slightly increased at $\theta_0 = 40^\circ$ and $\theta_0 = 45^\circ$ for $N = 9$. The influence of the scattered computational nodes on the performance for different preferential growth directions is analysed by running simulations with $m_{13} = 56$, as shown in Fig. 40. The steady-state growth velocity and the concentration are both less prone to rotation, especially for $N \neq 13$. The increase of the concentration for $\theta_0 = 40^\circ$ and $\theta_0 = 45^\circ$ for $N = 9$ at $m_{13} = 28$ is no longer present at $m_{13} = 56$.

6. Summary and conclusion

This paper aims to develop a novel numerical approach for the accurate and computationally efficient modelling of dendritic solidification. In the framework of this study, the PF formulation is applied to model the dendritic solidification of pure materials and binary alloys. The meshless RBF-FD method and the forward Euler scheme are used for the spatial and temporal discretisation of the PDEs. The space-time adaptive procedure, based on dynamic quadtree domain decomposition, is developed to increase the numerical approach’s computational efficiency. The procedure dynamically ensures the highest density of computational nodes and the finest time-stepping at the solid-liquid interface. Regular or scattered computational nodes are generated in each quadtree sub-domain.

The use of the scattered node distribution provides higher accuracy for otherwise same numerical parameters when the growth for an arbitrary preferential growth direction is considered. Furthermore, the solution is much more sensitive to the size of the local RBF-FD sub-domain using the regular node distribution. The system of PF equations is very sensitive to the strength of anisotropy of the surface energy. Even a small amount of the discretisation-induced anisotropy, therefore, also plays an important role in the evolution of a dendrite. The underlying regular node distribution apparently yields the directions which are more favourable for growth since the local RBF-FD sub-domains in the
whole computational domain have the same shape. However, in the case of scattered node distribution, the shape of the local sub-domain differs from node to node, which apparently breaks the favourable directions and reduces the overall discretisation-induced anisotropy.

The RBF-FD method proves to be very suitable for the PF modelling of dendritic solidification since the implementation of the RBF-FD method using scattered node distribution is simple and straightforward. In a lot of the PF studies of dendritic solidification \([25,26,67,62,60,76]\) etc., the mesh-based finite difference method is used for spatial discretisation. The finite difference method is used exclusively in combination with the regular node distribution and is therefore much more sensitive to the mesh-induced anisotropy effects \([65]\). A straightforward application of the scattered computational nodes represents one of the distinctive advantages of the present numerical approach. The advantages of the use of the scattered node distribution are to some extent compromised due to rectangular quadtree domain decomposition. The increase of the quadtree sub-domain size/node spacing ratio mitigates the effect of rectangular domain decomposition on the solution obtained using scattered nodes.

The speed-up of the adaptive solution procedure is an increasing function of the area of the computational domain and a decreasing function of the length of the solid-liquid interface and the node spacing. Adaptive time-stepping speeds up the calculations further. The solid-liquid interface can be optimally resolved by decreasing the quadtree sub-domain size/node spacing ratio, which results in a small number of all computational nodes and consequently short iteration times. Decrease of the quadtree sub-domain size/node spacing ratio, however, increases the number of quadtree sub-domains and, therefore, the communication and adaptation times. The optimal value of the quadtree sub-domain size/node spacing ratio with the highest speed-up is a consequence of the interplay of these phenomena. The compromise has to be made in the selection of the quadtree sub-domain size/node spacing ratio since the accuracy and the computational efficiency are an increasing and decreasing function of the ratio.

The present work demonstrates that the developed space-time adaptive approach provides an efficient computational tool. The paper shows that the use of RBF-FDs, defined on scattered node distributions, reduces discretisation-induced anisotropy in the solution of the PF model. The adaptive approach can be straightforwardly extended to 3-D using an octree \([62]\) instead of the quadtree data structure. Adaptive discretisation-order algorithm can be easily implemented, e.g., higher-order interpolation can be used in quadtree sub-domains at the solid-liquid interface. We can also easily change the number of nodes in a local RBF-FD sub-domain on different quadtree sub-domains. For instance, we can use a higher number of nodes in the quadtree sub-domains at the solid-liquid interface and a lower number in the bulk of phases. That way, we can increase computational efficiency while sustaining high accuracy near the solid-liquid interface. Furthermore, a physics-adaptive algorithm can be straightforwardly considered. For example, only the heat diffusion equation can be solved in the quadtree sub-domains in the bulk of phases, while the heat diffusion and PF equations can be solved in the quadtree sub-domains at the interface.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by the Slovenian Research Agency (grant numbers J2-1718 (basic project), P2-0162 (programme group)); and the Slovenian Research Agency and Store Steel company (grant number L2-09246 (applied project)).

References


